LEARN TO SCALE REASONING AT TEST TIME VIA REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling test-time computation has emerged as a powerful paradigm for Large Language Models (LLMs). The effectiveness of sequential test-time scaling depends on whether models can effectively utilize additional inference budgets for further reasoning. However, we show that RLVR-trained reasoning models struggle to benefit from additional inference computation, even when larger budgets are available. In this work, we investigate whether RL training can teach LLMs to benefit from additional inference computation. To analyze how training changes the use of additional computation, we introduce state coverage as a proxy for in-context exploration and find that incentivizing additional computation during training can broaden coverage but also induce repetition. These findings motivate Learning to scale INference with lEss repetition (LINE), a reward-shaping approach that encourages continued reasoning on unsuccessful attempts while controlling redundancy. Experiments show that LINE enables Qwen3-4B-Base to translate additional inference computation into accuracy gains, while also improving average mathematical reasoning accuracy by 4.4 percentage points over GSPO and strengthening out-of-domain generalization. Broader evaluations show that LINE improves reasoning performance across multiple models and benchmarks.

1 INTRODUCTION

Scaling test-time computation, often conceptualized as enabling models to "think harder" before answering, has emerged as a powerful paradigm for breaking the performance ceiling of Large Language Models (LLMs) (Snell et al., 2024; Wu et al., 2024; Liu et al., 2025a). Broadly, Test-Time Scaling (TTS) strategies fall into two primary regimes: Parallel Scaling (Lightman et al., 2023; Snell et al., 2024; Liu et al., 2025a; Brown et al., 2024; Wang et al., 2022), which aggregates outputs from multiple independent samples, and Sequential Scaling (Guo et al., 2025; Jaech et al., 2024; Kumar et al., 2024), which prioritizes extended reasoning chains or iterative refinement. Unlike parallel scaling, sequential scaling requires a model to effectively utilize additional computation within a single reasoning process.

However, reasoning models trained with Reinforcement Learning with Verifiable Rewards (RLVR) struggle to benefit from additional inference computation. Many unsuccessful responses terminate with substantial budget remaining (Figure 1(a)), and increasing the available budget beyond the training limit yields little further accuracy improvement (Figure 1(b)). Further, forcing models to keep reasoning with "Wait" cues, as in s1 (Muennighoff et al., 2025), does not reliably improve accuracy and can even hurt performance, despite increasing the computation they use (Figure 1(c)). These observations raise a natural question:

Can RL training teach models to benefit from additional inference computation within a single reasoning trajectory?

We begin with a simple intervention that rewards additional computation on incorrect responses (Section 3.2). This intervention increases compute use and improves both training reward and reasoning performance (Figure 2), suggesting that RL can indeed teach models to benefit from extra computation. However, it leaves open *how* the additional computation is spent. To characterize this, we analyze reasoning trajectories through the lens of in-context exploration (Gandhi et al., 2024; Setlur et al., 2025). Specifically, we map each trajectory onto abstract reasoning states at lexical, mathematical, and behavioral levels, and measure *state coverage*, defined as the number of distinct states visited within a trajectory (Section 3.3). We find that additional computation translates into

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.

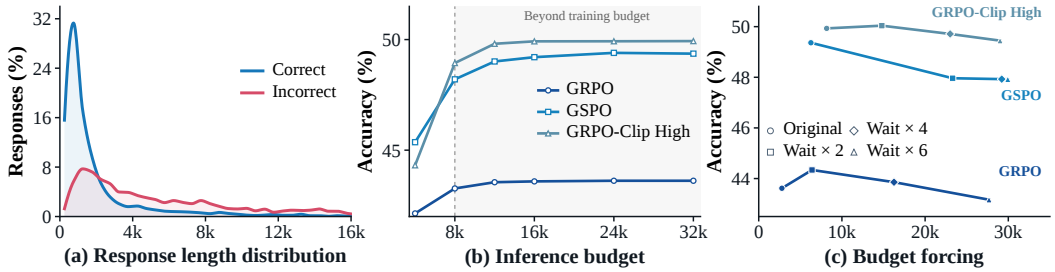


Figure 1: **Limited budget utilization within a single reasoning trajectory.** **(a)** Computation allocated to correct and incorrect responses. **(b)** Larger-budget generation. **(c)** Budget forcing using s1-style "Wait" cues (Muennighoff et al., 2025). Increasing the available budget or forcing its use yields limited or inconsistent accuracy gains for these RL baselines. All accuracy values are averaged over five in-domain test sets.

broader state coverage across all three abstractions (Figure 4). However, incentivizing computation alone does not ensure its productive use: models can also satisfy the compute objective through repetitive, low-information continuations that revisit previously covered states (Figure 5). This suggests that the benefit of additional computation depends not only on how much is spent, but also on whether it reaches new states.

Based on these observations, we propose *Learning to scale INference with lEss repetition* (LINE), a simple reward-shaping approach that encourages unsuccessful attempts to continue reasoning while controlling repetitive degeneration. LINE combines a computation incentive with a repetition penalty and can be directly integrated into standard RLVR training without additional process-level supervision or inference-time controllers.

Experiments across Qwen3 (Yang et al., 2025a), LLaMA (Grattafiori et al., 2024), and diverse reasoning benchmarks show that LINE helps models use additional inference computation more effectively. On Qwen3-4B-Base, applying LINE to GSPO improves average in-domain and out-of-domain accuracy by 4.4 and 1.5 percentage points, respectively (Table 1), and improves matched-budget accuracy beyond the 8k training horizon (Figure 6). Our key contributions are as follows:

- **Measuring in-context exploration within a trajectory.** We represent within-trajectory exploration through distinct abstract reasoning states and use state coverage to measure how additional computation changes the states visited by a model.
- **A simple recipe for training models to use additional computation.** We propose LINE, a simple recipe that encourages incorrect trajectories below a reference-relative compute target to use additional computation while mitigating repetitive degeneration.
- **Improved sequential test-time scaling.** On Qwen3-4B-Base, LINE improves matched-budget accuracy, including at budgets beyond the 8k training horizon. Additional evaluations show reasoning accuracy gains across Qwen3 and LLaMA models.

2 Related Work

Sequential Test-Time Computation. Recent works have shown remarkable potential in scaling test-time computation (Snell et al., 2024) by training models to devote more computation to chains of thought (CoT) that exhibit reasoning behaviors, including verification and self-correction (Kumar et al., 2024). Scaling test-time computation significantly enhances the performance of Large Language Models (Guo et al., 2025; Team et al., 2025; Luo et al., 2025; Jaech et al., 2024) on various domains. Reinforcement Learning (RL), in particular, facilitates model-adaptive scaling of test-time computation through intrinsic adaptive exploration and strategy application (Guo et al., 2025; Team et al., 2025; Gandhi et al., 2025; Setlur et al., 2025; Yeo et al., 2025). These processes are mechanistically driven by negative gradients (Setlur et al., 2025; Zhu et al., 2025) or high entropy (Cui et al., 2025; Wang et al., 2025a; Cheng et al., 2025). Our focus is on learning within-trajectory reasoning behavior during RL: we encourage greater use of the available computation while controlling repetition.

Budget-Aware Reasoning. Recent studies have investigated budget-aware reasoning, specifically aiming to mitigate overthinking and improve both efficiency and controllability. Several works (Jiang et al., 2025; Huang et al., 2025; Zhang et al., 2025a; Kang et al., 2025; Zhang et al., 2025c; Yu et al., 2025b; Yang et al., 2025c; Liu et al., 2025b) propose learning to adapt Chain-of-Thought (CoT)

computation to problem difficulty. Others (Zhang et al., 2025b; Ma et al., 2025) focus on controlling inference compute, while some studies address reasoning within a specific budget (Wen et al., 2025; Aggarwal & Welleck, 2025; Xu et al., 2025). Our focus is complementary: we study how to encourage additional computation on unsuccessful trajectories without inducing repetitive degeneration.

3 Training Models to Use Additional Test-Time Computation

We investigate whether RL training can help models benefit from additional computation within a single reasoning trajectory. Starting from the limited gains of RLVR baselines under larger inference budgets, we first test a computation-only intervention that encourages unsuccessful attempts to continue. We then use in-context exploration to analyze the resulting trajectories and identify the repetition problem.

3.1 Background

Autoregressive reasoning as an MDP. We model autoregressive generation as a finite-horizon Markov Decision Process (MDP). Given an input query x, the state $s_t = [x, y_{<t}]$ is the current context, and the action $a_t = y_t \in \mathcal{V}$ is the next token selected by the policy $\pi_\theta(a_t \mid s_t)$. The transition is deterministic: $s_{t+1} = s_t \oplus a_t$, where $\oplus$ denotes concatenation. Generation terminates at an end-of-sequence (EOS) token or the maximum response budget. In RLVR, a verifier assigns a terminal correctness reward $R_{\text{acc}}(x, \boldsymbol{\tau})$ to the completed trajectory $\boldsymbol{\tau}$.

Reinforcement Learning for LLM Reasoning. The policy is trained to maximize the expected trajectory reward $J(\pi_\theta) = \mathbb{E}_{x,\boldsymbol{\tau}\sim\pi_\theta(\cdot|x)}[R(x, \boldsymbol{\tau})]$ using policy gradient methods (Sutton et al., 1999):

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{x,\boldsymbol{\tau}\sim\pi_\theta(\cdot|x)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, A^{\pi_\theta}(s_t, a_t)\right],$$

where $A^{\pi_\theta}(s_t, a_t)$ is the advantage of taking action a_t. GRPO (Shao et al., 2024) and GSPO (Zheng et al., 2025) are both instances of policy gradient methods, with GRPO using token-level importance weights and GSPO using sequence-level importance weights. Details of these algorithms are provided in Appendix C.2.

3.2 Can RL Teach Models to Use Additional Test-Time Computation?

To characterize how RLVR-trained models use additional test-time computation, we evaluate GRPO, GRPO with Clip-Higher (GRPO-CH), and GSPO with larger inference budgets and under forced continuation. We follow the experimental setup in Section 5.1. Increasing the available budget from 16k to 32k yields only marginal accuracy gains of 0.03, 0.16, and 0.01 percentage points for GRPO, GSPO, and GRPO-CH, respectively (Figure 1(b)). Similarly, forcing continued reasoning with s1-style "Wait" cues (Muennighoff et al., 2025) increases computation use but does not consistently improve accuracy (Figure 1(c)): from natural generation to "Wait" $\times 6$, accuracy changes from 43.6% to 43.2% for GRPO, from 49.4% to 47.9% for GSPO, and from 49.9% to 49.5% for GRPO-CH. These observations suggest that RLVR-trained models struggle to convert additional computation into consistent accuracy gains, motivating our intervention during training to encourage more effective use of additional inference computation.

Encouraging continued reasoning during training. Many incorrect responses terminate with substantial budget remaining (Figure 1(a)). In RLVR, a correctness-only reward assigns the same terminal reward to an early incorrect answer and an extended but still unsuccessful attempt, providing no preference for additional computation among these failures. We therefore test whether encouraging incorrect trajectories that underuse their available computation to continue during training can teach the model to use additional inference computation productively. We augment GSPO with the computation-only intervention R_{comp}; its exact definition is given in Section 4. Figure 2 shows that the intervention can improve reasoning performance as it increases computation use. To understand how the additional computation is used, we next analyze the trajectories through the lens of *in-context exploration*.

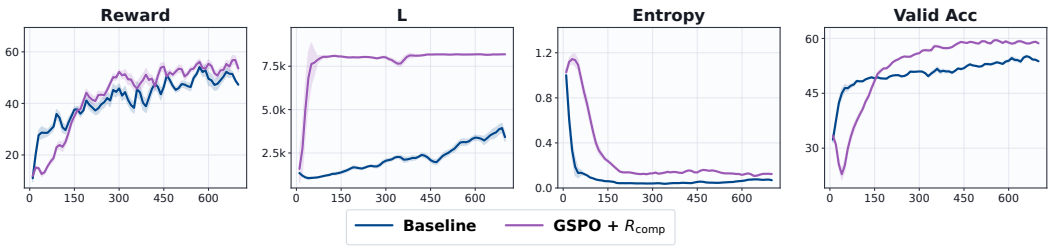


Figure 2: **Training dynamics for GSPO and the computation-only intervention (R_{comp}) on Qwen3-4B-Base.** The intervention increases computation use and improves training reward and validation accuracy.

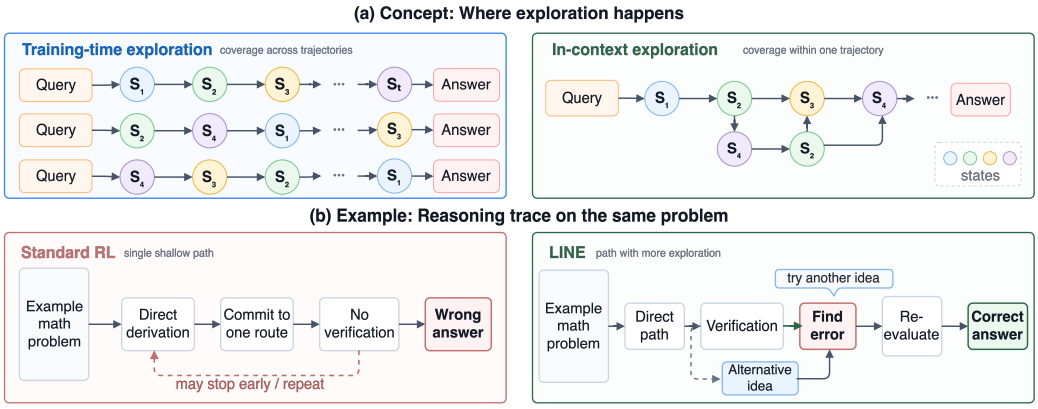


Figure 3: **The difference between in-context exploration and training exploration.** While training-time exploration covers states across sampled trajectories, in-context exploration visits diverse reasoning states within a single trajectory, as illustrated by the reasoning example below.

3.3 Analyzing In-Context Exploration

In-context exploration concerns the reasoning performed within one trajectory (Gandhi et al., 2024; Setlur et al., 2025). It differs from exploration across sampled training trajectories, as illustrated in Figure 3. We introduce *state coverage* as a proxy for this behavior.

Defining in-context states. Given an input query x and a reasoning trajectory $\boldsymbol{\tau} = (y_1, y_2, \ldots, y_T)$ of T generated tokens, we define the in-context state sequence as $S_{\text{IC}}(\boldsymbol{\tau}) = (s_1, s_2, \ldots, s_T)$. Directly counting raw autoregressive states is not informative because each state contains a different history prefix. Their count therefore reduces to the number of generation steps and cannot distinguish different reasoning content from repetitive continuation.

State abstraction. We introduce a state abstraction function ϕ that maps raw states to abstract states: $\tilde{s}_t = \phi(s_t) = \phi([x, y_{<t}])$. The abstraction ϕ can operate at different levels of granularity, such as lexical patterns, equations, or reasoning behaviors.

In-context state coverage. We define the ***In-Context Distinct State Count*** $C_{\text{context}}(\boldsymbol{\tau}; \phi)$ as

$$C_{\text{context}}(\boldsymbol{\tau}; \phi) = \left|\{\phi(s_t) \mid t = 1, \ldots, T\}\right|. \tag{1}$$

This quantity counts the distinct abstract states visited within a trajectory. It provides a proxy for in-context exploration under the chosen abstraction; visiting more states does not by itself establish reasoning progress.

Coverage metrics. We examine lexical, mathematical, and behavioral diversity using the following coverage proxies:

- **Distinct n-gram count (DNC):** measures lexical diversity through distinct n-grams.
- **Textual diversity (TD):** measures dissimilarity across trajectory segments using BLEU-based metrics (Papineni et al., 2002; Hu et al., 2025).
- **Distinct equation count (DEC):** measures diversity of formulations (Hu et al., 2025).

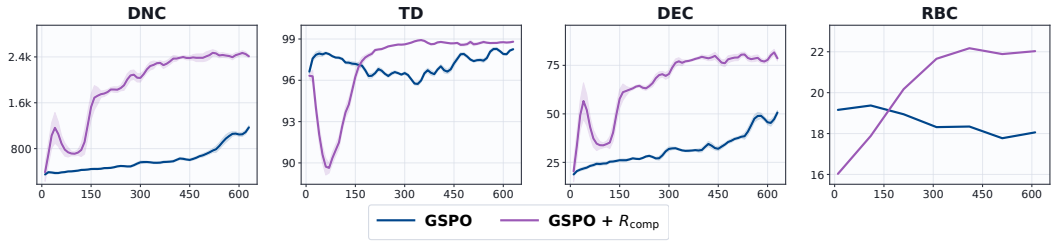


Figure 4: **In-context state coverage dynamics under GSPO and the computation-only intervention (R_{comp}) on Qwen3-4B-Base.** Adding R_{comp} increases DNC, TD, DEC, and RBC, indicating broader measured lexical, mathematical, and behavioral state coverage. These trends do not control for the increase in generated tokens.

- **Distinct reasoning behavior type count (RBC):** captures diversity of reasoning behaviors. Following Gandhi et al. (2025), we use Gemini-2.5-Flash-Thinking (Comanici et al., 2025) to annotate behavioral patterns and cluster them into distinct types. The resulting taxonomy is provided in Appendix B.

These metrics describe different aspects of the generated trajectories. In particular, TD measures dissimilarity rather than directly counting states. Because coverage counts also depend on compute use, we interpret them jointly with accuracy and repetition, rather than treating higher counts as evidence of more efficient reasoning.

The intervention broadens measured coverage. Figure 4 shows that the computation-only condition achieves higher DNC, TD, DEC, and RBC than GSPO alone. Thus, the increased compute use is accompanied by more distinct lexical patterns and equations, greater textual dissimilarity, and more reasoning behavior types within a trajectory. Together with the validation gains, these observations suggest that the intervention can encourage broader in-context exploration under our measurements.

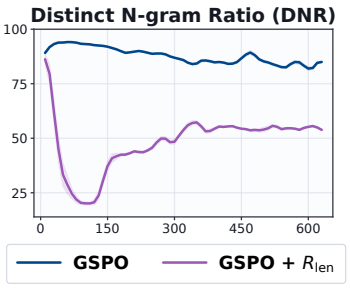


Figure 5: **Distinct n-gram ratio under computation-only training.**

Additional computation can degenerate into repetition. However, the additional computation is not necessarily productive. In the computation-only condition, GSPO tends to exhaust the available budget, while its distinct n-gram ratio (DNR), the fraction of n-gram occurrences that are distinct, drops substantially (Figure 5). The model therefore repeats local patterns more frequently even as the absolute number of distinct patterns grows. This exposes the limitation of computation-only shaping: it encourages continuation but does not distinguish further reasoning from repetition. These findings motivate combining the continuation intervention with explicit redundancy control, as formalized in Section 4.

4 Learning to scale INference with lEss repetition

Motivated by the observation that computation incentives can improve reasoning performance but may also introduce redundant continuation, we propose *Learning to scale INference with lEss repetition* (LINE), a simple reward-shaping recipe that encourages additional computation while controlling the repetition that can arise from the computation incentive alone.

Computation incentive. For each sample i, we define

$$B_{\text{target},i} = B_{\text{ref},i} + \Delta B, \tag{2}$$

where $B_{\text{ref},i}$ is the compute allocated by the reference policy and ΔB is the target compute increment. The computation incentive is applied only to incorrect trajectories whose allocated compute B falls below the target:

$$R_{\text{comp}} = \begin{cases} 0 & B \geq B_{\text{target}},\ R_{\text{acc}} = 0, \\ -(B_{\text{target}} - B) & B < B_{\text{target}},\ R_{\text{acc}} = 0, \\ 0 & R_{\text{acc}} = 1. \end{cases} \tag{3}$$

Thus, R_{comp} encourages additional computation on unresolved problems without incentivizing additional computation on correct responses.

Redundancy reward. As shown in Section 3.3, training with the computation incentive alone can generate low-information and repetitive continuations. We therefore introduce a redundancy reward R_{red}, which is zero within the redundancy tolerance and negative when a trajectory exceeds it. Let $\mathcal{D}(\boldsymbol{\tau}) \geq 0$ denote a redundancy score for trajectory $\boldsymbol{\tau}$, and let Θ be a tolerance threshold. We define

$$R_{\text{red}} = -\mathbb{I}\left[\mathcal{D}(\boldsymbol{\tau}) > \Theta\right], \tag{4}$$

This thresholded form leaves the reward unchanged when redundancy remains within tolerance and assigns a negative reward once excessive redundancy is detected.

Final objective. The final reward is $R = R_{\text{acc}} + \eta R_{\text{comp}} + \beta R_{\text{red}}$. The computation term encourages additional reasoning on incorrect trajectories, while the redundancy reward discourages spending that computation on repetition.

Connection to count-based exploration. LINE is conceptually related to count-based exploration (Bellemare et al., 2016): additional computation provides opportunities to visit reasoning states, while redundancy control discourages repeatedly visiting the same measured local patterns. We use this connection to interpret the resulting reasoning trajectories rather than directly optimize state coverage.

5 Experiments

In this section, we empirically examine whether the proposed reward-shaping recipe improves reasoning and test-time scaling. Specifically, we study the following research questions:

- RQ1: Does LINE improve reasoning accuracy over standard RL and reward baselines?
- RQ2: Does LINE convert additional computation into performance gains during test?
- RQ3: How is the additional computation used, and how does redundancy control affect it?

5.1 Experimental Setup

Evaluation. We assess performance across eight reasoning benchmarks, categorized into in-domain mathematical tasks and out-of-distribution (OOD) general reasoning. The in-domain suite comprises AIME 2024/2025, AMC (Li et al., 2024), MATH-500 (Hendrycks et al., 2021), and OlympiadBench (He et al., 2024). The OOD evaluation includes ARC-c (Clark et al., 2018), GPQA-Diamond (Rein et al., 2024), and MMLU-Pro (Wang et al., 2024). Regarding metrics, for benchmarks with limited sample sizes (AIME and AMC), we report the average accuracy over 32 independent runs (Avg@32). For all other benchmarks, we report Pass@1. All evaluations use a sampling temperature of 0.6 and top-p=1.0. The maximum inference budget is 32k unless otherwise specified; budget-scaling experiments and validation analyses use the budgets indicated in their figures and tables.

RLVR setups. Training is performed with a prompt batch size of 128, generating 8 rollouts per prompt. We update the policy using a mini-batch size of 32 and a learning rate of 1e-6. During the training phase, the sampling temperature is set to 1.0, with a maximum compute budget of 8,192 unless otherwise specified. Experiments are implemented using the verl framework[1] on nodes equipped with 4×H100 GPUs, employing Math-Verify[2] for outcome-based reward calculation. Unless otherwise stated, we use the default RLVR and reward-shaping configuration described in Appendix D. The computation-incentive coefficient is normalized over the full target range so that the computation incentive covers responses up to the 8k compute training limit.

Models and baselines. We use Qwen3-4B-Base as the primary testbed and additionally evaluate Qwen3-4B (non-thinking) and Llama-OctoThinker-3B (Wang et al., 2025b), using DAPO-Math-17k (Yu et al., 2025a), Polaris (An et al., 2025), and DeepMath-5k (He et al., 2025; Tan et al., 2025), respectively. We primarily study GSPO; on Qwen3-4B-Base, we compare GRPO, GRPO with Clip-Higher (GRPO-CH), GSPO, reward baselines (cosine reward (Yang et al., 2025b), DIVER-MIX (Hu et al., 2025), and Overlong reward (Yu et al., 2025a)), and LINE. The Overlong + Top20 variant updates only the top 20% highest-entropy tokens (Wang et al., 2025a); group filtering is disabled for both Overlong variants to keep training-data selection consistent. Unless stated otherwise, LINE uses the default n-gram instantiation of the generic redundancy score $\mathcal{D}(\boldsymbol{\tau})$; the semantic instantiation is evaluated only as a controlled GSPO alternative (Appendix D.2). Additional model-size results are reported in Appendix E.4.

[1] https://github.com/volcengine/verl

[2] https://github.com/huggingface/Math-Verify

Table 1: **Reasoning performance on Qwen3-4B-Base (%). Bold**/underlined: best/second-best, including ties. Blue: gains over the corresponding RL baseline. Parentheses identify the redundancy implementation.

Method		In-Domain Performance					Out-of-Domain Performance			
Algorithm	**Reward / redundancy**	**MATH**	**Olympiad**	**AMC**	**AIME 24/25**	**Avg.**	**ARC-c**	**GPQA**	**MMLU-Pro**	**Avg.**
Qwen3-4B-Base (before RL)		66.0	33.2	36.6	8.5/6.9	30.2	66.9	26.3	30.9	41.4
RL baselines										
GRPO	–	80.4	47.1	55.2	16.8/18.7	43.6	84.6	44.4	60.1	63.0
GRPO-CH	–	86.4	54.1	61.8	25.2/22.2	49.9	89.6	46.0	62.8	66.1
GSPO	–	85.2	51.7	62.7	26.7/20.5	49.4	88.4	**48.5**	61.5	66.1
Reward baselines										
GSPO	Cosine	75.2	38.1	51.8	10.3/9.4	36.9	81.7	38.4	54.2	58.1
GSPO	DIVER-MIX	87.8	54.4	66.3	28.1/23.5	52.0	90.0	42.4	63.4	65.3
GRPO-CH	Overlong	84.2	46.8	56.3	19.1/14.1	44.1	90.6	44.4	60.2	65.1
GRPO-CH	Overlong + Top20	83.0	45.6	59.6	22.1/17.8	45.6	89.7	42.4	62.1	64.7
Our recipes										
GRPO	LINE (n-gram)	85.0	49.9	60.5	22.9/16.4	46.9 (+3.3)	90.3	46.5	60.4	65.7 (+2.7)
GRPO-CH	LINE (n-gram)	**88.8**	54.1	65.5	30.4/24.5	52.6 (+2.7)	**91.5**	43.9	63.0	66.1 (+0.0)
GSPO	R_{comp} only	88.4	54.5	**66.8**	29.5/21.8	52.2 (+2.8)	91.0	40.9	**63.9**	65.2 (-0.9)
GSPO	LINE (semantic)	**88.8**	54.5	61.9	28.6/23.2	51.4 (+2.0)	90.6	44.9	63.1	66.2 (+0.1)
GSPO	LINE (n-gram)	**88.8**	**57.2**	66.2	**30.5/26.7**	**53.8** (+4.4)	91.4	47.5	63.8	**67.6** (+1.5)

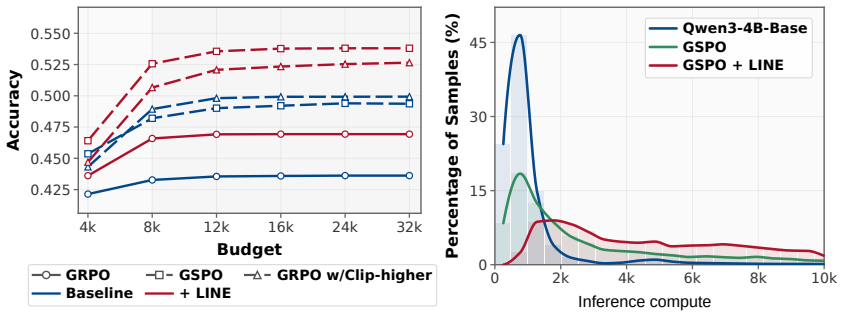


Figure 6: **Test-time scaling performance under controlled inference budgets on Qwen3-4B-Base.** Accuracies are calculated over five in-domain mathematical benchmarks as shown in Table 1.

5.2 Results

RQ1: LINE improves reasoning accuracy over strong baselines. As shown in Table 1, LINE improves average in-domain performance and OOD performance in most benchmarks. When applied to GSPO, LINE increases the in-domain average from 49.4% to 53.8%, with particularly large gains on AIME25 (6.2 percentage points). The OOD average also improves from 66.1% to 67.6%, suggesting that the benefit is not limited to the training distribution. The semantic GSPO alternative also improves over GSPO, but is less consistent across benchmarks; its controlled comparison is reported in Appendix D.2. We further verify that the improvements are robust across independent runs (Appendix E.5). Similar in-domain improvements (2–3 percentage points) are observed on Qwen3-4B and Llama-OctoThinker-3B (Wang et al., 2025b) in Table 8, as detailed in Appendix E.3. We further verify that the improvement consistently holds across different model scales varying from 1.7B to 8B (Appendix E.4). These results indicate that LINE is effective across the tested settings, especially for in-domain mathematical reasoning.

RQ2: LINE helps models benefit from additional inference computation. Figure 6 evaluates models under shared maximum budgets. The left panel shows that LINE maintains higher accuracy than the standard RL baselines at matched budgets. At the 8k budget, LINE reaches 52.6%, compared with 48.2% for GSPO, and at 32k it reaches 53.8%, compared with 49.4% for GSPO. Across the full range from 4k to 32k, LINE improves by 7.4 percentage points, whereas GSPO improves by 4.0 points. Complete results are provided in Appendix E.2. The right panel shows that LINE makes greater use of the available inference budget. Figure 10 shows that LINE can more

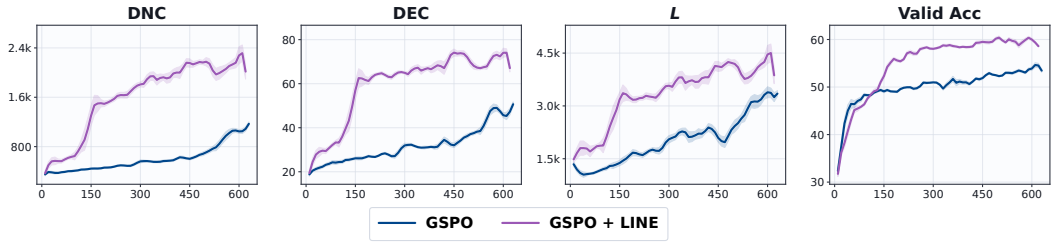


Figure 7: **Training dynamics comparing GSPO and GSPO + LINE on Qwen3-4B-Base.** We track compute use, validation accuracy (8k), distinct n-gram count (DNC), and distinct equation count (DEC) on the validation set. Compared to GSPO, LINE achieves higher state coverage (DNC, DEC) and better accuracy.

stably convert computation into performance under budget forcing than the baselines. These results suggest that LINE helps models benefit from additional inference computation.

RQ3: LINE changes how additional computation is used. To analyze how LINE changes in-context state coverage, we examine the training dynamics in Figure 7. Compared to the GSPO baseline, LINE uses more inference computation and achieves a rapid increase in both distinct n-gram counts and distinct equation counts, indicating broader measured state coverage. Together with the repetition observed without repetition control in Figure 5 and the accuracy comparison in Table 1, these results suggest that incentivizing computation alone is insufficient to account for the gains. Appendix E.1 further reports coverage at matched maximum budgets and per normalized compute. These results show that LINE changes how the available computation is used, yielding broader measured reasoning patterns.

5.3 Discussion

Global diversity. We further analyze global exploration dynamics using Global Distinct N-gram Count and policy entropy. As shown in Figure 8, standard GSPO has lower global distinct n-gram counts and a faster entropy decline than GSPO + LINE. Both methods show an overall increase in global distinct n-gram counts, while LINE maintains higher counts and entropy. This pattern is consistent with reduced concentration on a small set of response patterns and may help the model discover rare, high-reward states.

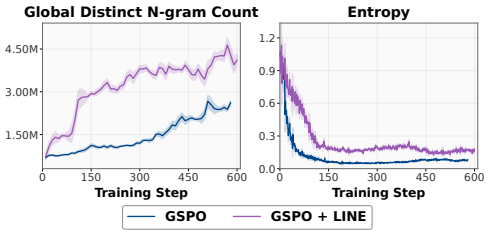


Figure 8: **Global exploration dynamics under GSPO and GSPO + LINE on Qwen3-4B-Base.**

The effect of LINE across difficulties. Following Sun et al. (2025), we evaluate LINE on AIME 2024 across difficulty levels. As shown in Table 3, LINE consistently improves accuracy on easy and medium problems, with the largest gains on the easier subsets, and also yields a modest improvement on hard problems. However, this benefit does not extend to the extremely hard subset, where the accuracy remains 0.0 across all three runs. These results show gains on the easy, medium, and hard subsets, but no observed improvement on the extremely hard subset in this evaluation. We further explore this question by studying the interaction with SFT (Appendix E.7).

The effect of ΔB. In this setting, increasing ΔB systematically increases allocated computation and improves validation accuracy from 56.9% to 63.4%, with a plateau at 61.6% for intermediate increments (Figure 9). At the same time, state coverage metrics (DNC and DEC) increase steadily, indicating that trajectories using more computation contain a broader range of measured reasoning states. In this setting, increasing the computation incentive is accompanied by further accuracy gains. While a slight drop in the distinct n-gram ratio (DNR) is observed at larger ΔB, this effect is minor and does not offset the overall gains. These results suggest that increasing the target compute increment during training can further help models benefit from additional test-time computation.

Continual scaling via curriculum training. We further extend the compute budget through a second training stage with a larger computation budget. We also evaluate an e3-style (Setlur et al., 2025) difficulty curriculum, using easy samples identified by Qwen3-30B-A3B-Thinking in Stage 1 and medium/hard samples in Stage 2, with and without LINE. As shown in Table 2, extending

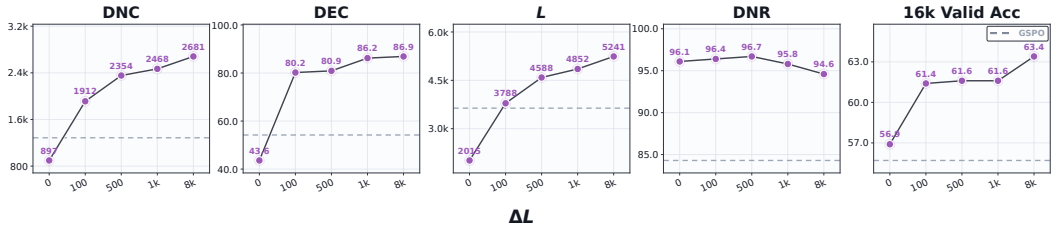


Figure 9: **The effect of ΔB on Qwen3-4B-Base.** As ΔB grows, the model uses more inference compute and achieves higher state coverage (DNC, DEC) and improved accuracy.

Table 2: **Continual scaling via curriculum training on Qwen3-4B-Base.** Stage 1 trains with an 8k budget, and Stage 2 extends the corresponding models to 12k. We compare GSPO, our reproduction of the e3-style difficulty curriculum (Setlur et al., 2025), and their combinations with LINE. **Bold** denotes the best result within each stage. Stage-2 averages include the gain over the corresponding Stage-1 average in parentheses.

Method	MATH-500	OlympiadBench	AMC	AIME 2024	AIME 2025	Average
Qwen3-4B-Base	66.0	33.2	36.6	8.5	6.9	30.2
Stage 1: 8k training budget						
GSPO	85.2	51.7	62.7	26.7	20.5	49.4
e3	85.4	52.4	60.3	22.8	14.5	47.1
e3 + LINE	85.0	51.6	61.5	25.2	17.0	48.1
GSPO + LINE	**88.8**	**57.2**	**66.2**	**30.5**	**26.7**	**53.8**
Stage 2: 12k training budget						
GSPO	87.8	52.1	64.9	26.8	22.7	50.9 (+1.5)
e3	87.4	55.0	66.0	29.7	22.0	52.0 (+4.9)
e3 + LINE	88.6	55.6	68.1	29.9	26.9	53.8 (+5.7)
GSPO + LINE	**89.4**	**59.0**	**71.7**	**33.4**	**27.9**	**56.3** (+2.5)

the training budget from 8k to 12k improves GSPO + LINE from 53.8% to 56.3%, compared with 50.9% for GSPO and 52.0% for the e3-style curriculum at Stage 2. Adding LINE to the e3-style curriculum improves its Stage-2 average from 52.0% to 53.8%, suggesting that reward shaping can complement curriculum-based training. GSPO + LINE has the highest absolute accuracy at both stages, while the e3-based variants show larger gains between stages.

Table 3: Performance across difficulty levels on AIME 2024. We report Avg@32 accuracy (%) and average compute (k units). All responses are generated up to a maximum compute budget of 32k.

	Accuracy (%)					Compute (k units)				
Method	Easy	Medium	Hard	Ext. Hard	Overall	Easy	Medium	Hard	Ext. Hard	Overall
GSPO	84.4	37.5	7.3	0.0	26.7	1.9	8.3	13.2	7.6	9.3
GSPO + LINE	**95.8** (↑11.4)	**44.2** (↑6.7)	**10.2** (↑2.9)	0.0 (–)	**31.6** (↑4.9)	3.7	9.1	13.3	11.7	10.4

6 Conclusion

In this work, we investigate how to improve test-time scaling in large language models, using in-context exploration as a perspective for analyzing reasoning trajectories. We show that state coverage provides a useful proxy for understanding these trajectories, and that additional computation is associated with improved performance. Building on this insight, we propose LINE, a simple yet effective approach that combines computation incentives with redundancy control to encourage additional computation while limiting repetitive continuation. Across multiple models and benchmarks, our method improves reasoning performance; in the primary setting, it also improves sequential test-time scaling behavior. Overall, our findings highlight that effective test-time scaling is not merely about increasing computation, but about training models to use that computation more effectively, as reflected in budget-conditioned accuracy and measured state coverage.

Ethics Statement

This work does not involve human participants, personal data, or interventions in real-world decision-making. The experiments use language models and publicly available reasoning benchmarks, with outcome-based evaluation and no collection of sensitive information. Improving reasoning capabilities may also increase the potential for misuse, overreliance on generated solutions, and unequal access to computation. In addition, the proposed use of extra inference computation can increase energy consumption and computational cost. We therefore view efficiency, safety evaluation, and responsible deployment as important considerations when applying these methods beyond the benchmark settings studied here.

Reproducibility Statement

The computation incentive, thresholded redundancy reward, and combined reward objective are defined in Section 4. The model, data, prompt, training, and evaluation settings are described in Section 5.1 and Appendix D. Appendix D also reports the RL hyperparameters, compute budgets, GPU resources, and prompt templates. We use the public DAPO-Math-17k training data and public reasoning benchmarks, implement training with the verl framework, and use Math-Verify for outcome-based reward calculation. Additional model-scale, random-seed, matched-budget, ablation, and redundancy analyses are provided in Appendix E.1–F. Training scripts and configuration files will be released to support independent reproduction of the reported results.

AI Use Statement

AI-based writing tools were used to assist with language editing, proofreading, and improving the clarity of the manuscript. The authors were responsible for the scientific ideas, methodology, experiments, analysis, and conclusions, and reviewed and approved the final text.

References

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2915–2923, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/abel16.html.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of machine learning research*, 3(Nov):397–422, 2002.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

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.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, and etc Ice Pasupat. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

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.

Yunzhen Feng, Julia Kempe, Cheng Zhang, Parag Jain, and Anthony Hartshorn. What characterizes effective reasoning? revisiting length, review, and structure of cot. *arXiv preprint arXiv:2509.19284*, 2025.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen 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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

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

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.

Shijue Huang, Hongru Wang, Wanjun Zhong, Zhaochen Su, Jiazhan Feng, Bowen Cao, and Yi R Fung. Adactrl: Towards adaptive and controllable reasoning via difficulty-aware budgeting. *arXiv preprint arXiv:2505.18822*, 2025.

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

Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang, Tengchao Lv, Lei Cui, and Furu Wei. Think only when you need with large hybrid-reasoning models. *arXiv preprint arXiv:2505.14631*, 2025.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24312–24320, 2025.

J Zico Kolter and Andrew Y Ng. Near-Bayesian exploration in polynomial time. In *Proceedings of International Conference on Machine Learning*, pp. 513–520, 2009.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

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

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

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025a.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping, 2025b. URL https://arxiv.org/abs/2505.15612.

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 01-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-01-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8, 2025.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

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.

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

Amrith Setlur, Matthew YR Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms. *arXiv preprint arXiv:2506.09026*, 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.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Yiyou Sun, Georgia Zhou, Haoyue Bai, Hao Wang, Dacheng Li, Nouha Dziri, and Dawn Song. Climbing the ladder of reasoning: What LLMs can—and still can't—solve after SFT? In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=vrzApLEOdH.

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

Yuqiao Tan, Minzheng Wang, Shizhu He, Huanxuan Liao, Chengfeng Zhao, Qiunan Lu, Tian Liang, Jun Zhao, and Kang Liu. Bottom-up policy optimization: Your language model policy secretly contains internal policies. *arXiv preprint arXiv:2512.19673*, 2025.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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

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*, 2025a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025b.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Budgetthinker: Empowering budget-aware llm reasoning with control tokens. *arXiv preprint arXiv:2508.17196*, 2025.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Junnan Li, and Caiming Xiong. Scalable chain of thoughts via elastic reasoning. *arXiv preprint arXiv:2505.05315*, 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.

Shiming Yang, Yuxuan Tong, Xinyao Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=OLodUbcWjB.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for llm reasoning. *arXiv preprint arXiv:2502.18080*, 2025c.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025a.

Zishun Yu, Tengyu Xu, Di Jin, Karthik Abinav Sankararaman, Yun He, Wenxuan Zhou, Zhouhao Zeng, Eryk Helenowski, Chen Zhu, Sinong Wang, et al. Think smarter not harder: Adaptive reasoning with inference aware optimization. *arXiv preprint arXiv:2501.17974*, 2025b.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. *arXiv preprint arXiv:2505.13417*, 2025a.

Junyu Zhang, Runpei Dong, Han Wang, Xuying Ning, Haoran Geng, Peihao Li, Xialin He, Yutong Bai, Jitendra Malik, Saurabh Gupta, et al. Alphaone: Reasoning models thinking slow and fast at test time. *arXiv preprint arXiv:2505.24863*, 2025b.

Shengjia Zhang, Junjie Wu, Jiawei Chen, Changwang Zhang, Xingyu Lou, Wangchunshu Zhou, Sheng Zhou, Can Wang, and Jun Wang. Othink-r1: Intrinsic fast/slow thinking mode switching for over-reasoning mitigation. *arXiv preprint arXiv:2506.02397*, 2025c.

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

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*, 2025.

APPENDIX

A Limitations **15**

B Reasoning Behavior Taxonomy **15**

C Background **17**
C.1 MDP Formulation of LLM Reasoning 17
C.2 Reinforcement Learning for LLM Reasoning 17
C.3 Count-Based Exploration Background 18

D Experimental Details **18**
D.1 Template Prompt 18
D.2 Implementation of RL 19
D.3 Compute resources 19

E Additional Results **20**
E.1 State coverage at matched budgets with compute normalization 20
E.2 Test-time scaling across inference budgets 20
E.3 Extension to more models 20
E.4 Scalability Across Model Scales 21
E.5 Random Robustness 22
E.6 Pitfalls of Direct State Coverage Maximization 22
E.7 SFT as a Complement 23
E.8 Hyperparameter Sensitivity Analysis 24

F Case Study **25**

A LIMITATIONS

Our work has several limitations. First, our primary RL experiments use an 8k compute budget, while a separate continual-scaling experiment extends training to 12k; the models tested have up to 8B parameters. It is interesting to extend LINE to larger scales. Second, while LINE improves reasoning trajectories during RL, it does not directly address the acquisition of new reasoning capabilities. Preliminary experiments with supervised fine-tuning (SFT) (Appendix E.7) suggest potential complementary effects, but a systematic integration remains open. Third, our redundancy control relies on n-gram-based design, which cannot fully capture semantic repetition. Although we explore embedding-based alternatives (Appendix D.2), they are less stable and sensitive to implementation details. Finally, our understanding of how LINE affects reasoning processes is still limited; preliminary qualitative analyses (Appendix F) suggest possible structural differences, but are not yet conclusive. Due to limited academic resources, our experiments are constrained in scale, and further investigation at larger scales is an important direction for future work.

B REASONING BEHAVIOR TAXONOMY

Table 4 summarizes the reasoning behavior categories extracted from reasoning trajectories using Gemini-2.5-Flash-Thinking. Each category consolidates semantically similar raw annotations into a compact and interpretable behavior label.

Table 4: Reasoning behavior taxonomy. Each category is defined by a concise operational description and illustrated with example raw tags.

Behavior Category	Operational Definition	Example Tags
Backtracking	Revising an approach after detecting an error, contradiction, or dead end.	Backtracking; Strategy Shift; Revising Approach.
Verification	Checking results, assumptions, constraints, or answer consistency.	Verification; Constraint Check; Plausibility Check.
Subgoal Setting	Breaking a complex problem into manageable intermediate steps.	Subgoal Setting; Problem Decomposition; Structured Analysis.
Enumeration	Systematically considering multiple cases, possibilities, or hypotheses.	Enumeration; Reasoning by Cases; Exploring Alternatives.
Abstraction and Generalization	Extracting general principles or structures from specific instances.	Abstraction; Generalization; Pattern Generalization.
Simplification and Reduction	Reducing complexity through substitution, smaller cases, or removing details.	Simplification; Variable Substitution; Simpler Cases.
Problem Reframing and Transformation	Changing the representation or interpretation of the problem.	Problem Reframing; Representation Change; Transformation.
Constraint and Boundary Analysis	Reasoning about constraints, domains, edge cases, or feasible ranges.	Constraint Analysis; Boundary Analysis; Range Analysis.
Logical Deduction and Derivation	Deriving conclusions through step-by-step logical or algebraic reasoning.	Logical Deduction; Algebraic Manipulation; Derivation.
Strategic Planning and Method Selection	Choosing or planning an appropriate solution strategy.	Strategic Planning; Method Selection; Strategic Choice.
Hypothesis Generation and Heuristic Exploration	Proposing tentative ideas or heuristics under uncertainty.	Hypothesis Generation; Heuristic Search; Trial and Error.
Pattern Recognition and Analogy	Identifying recurring structures, symmetries, or analogies.	Pattern Recognition; Analogy; Symmetry Recognition.
Mathematical Knowledge Application	Applying known formulas, theorems, properties, or domain knowledge.	Knowledge Application; Formula Retrieval; Theorem Recall.
Tool Use and Computational Integration	Using code, calculators, algorithms, or external tools to support reasoning.	Tool Use; Computational Modeling; Algorithmic Thinking.
Definition Clarification and Formalization	Clarifying terms, variables, conditions, or formal notation.	Definition Clarification; Formalization; Variable Definition.
Self-Correction and Metacognitive Monitoring	Monitoring uncertainty, errors, assumptions, or reasoning progress.	Self-Correction; Metacognition; Error Detection.

Continued on next page

Table 4: Reasoning behavior taxonomy continued.

Behavior Category	Operational Definition	Example Tags
Unit Precision and Calculation Structuring	Managing units, precision, intermediate results, or computation structure.	Unit Consistency; Precision Awareness; Structured Calculation.

C Background

C.1 MDP Formulation of LLM Reasoning

LLM Reasoning as an MDP. We model the autoregressive generation process of a Large Language Model (LLM) as a deterministic MDP tuple $(\mathcal{S}, \mathcal{A}, \boldsymbol{\pi}, T)$, where

- **State Space ($\mathcal{S}$):** The state space $\mathcal{S}$ consists of all possible sequences of tokens from the vocabulary $\mathcal{V}$. Crucially, a specific state $s_t \in \mathcal{S}$ at time step t is the concatenation of the input query (prompt) x and the thought chain generated so far $y_{<t}$. Formally, $s_t = [x, y_1, \ldots, y_{t-1}]$.
- **Action Space ($\mathcal{A}$):** The action space is discrete and equivalent to the model's vocabulary $\mathcal{V}$, where an action $a_t \in \mathcal{V}$ corresponds to selecting the next token to append to the current sequence.
- **Transition Dynamics (T):** The transition is deterministic. Given the current state s_t and action a_t, the next state is uniquely determined by appending the token to the history: $s_{t+1} = s_t \oplus a_t$. The process terminates when a special end-of-sequence (EOS) token is generated.
- **Policy ($\boldsymbol{\pi}_\theta$):** The LLM functions as a parameterized policy $\boldsymbol{\pi}_\theta(a_t|s_t)$, which maps the current context s_t to a probability distribution over the vocabulary $\mathcal{V}$.

C.2 Reinforcement Learning for LLM Reasoning

Reinforcement Learning for LLM Reasoning. To optimize the policy $\boldsymbol{\pi}_\theta$ to maximize the expected return $J(\pi_\theta) = \mathbb{E}[\sum r_t]$, the optimization is typically performed via **Policy Gradient** (Sutton et al., 1999):

$$\nabla_\theta J(\boldsymbol{\pi}_\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log \boldsymbol{\pi}_\theta(a_t \mid s_t)\, A^{\boldsymbol{\pi}}(s_t, a_t) \right],$$

where $A^{\boldsymbol{\pi}}$ represents the advantage of taking action a_t. GRPO and GSPO are instantiations of this framework.

Group Relative Policy Optimization (GRPO). GRPO (Shao et al., 2024) performs optimization at the token level without a value function. It utilizes the standard per-token probability ratio:

$$\rho_{i,t}(\theta) = \frac{\boldsymbol{\pi}_\theta(y_{i,t}|x, y_{i,<t})}{\boldsymbol{\pi}_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}, \tag{5}$$

where $y_{i,t}$ is the t-th token of the i-th sequence. The advantage is computed via group normalization:

$$\hat{A}_i = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G)}. \tag{6}$$

The objective averages the PPO-clip loss (Schulman et al., 2017) over all generation steps in the sampled trajectories:

$$J^{\text{GRPO}}(\theta) = \mathbb{E}_{x,\{y_i\}\sim\boldsymbol{\pi}_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min \Big(\rho_{i,t}(\theta)\hat{A}_i, \right.$$
$$\left. \text{clip}(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i \Big) \right]. \tag{7}$$

Group Sequence Policy Optimization (GSPO). GSPO (Zheng et al., 2025) elevates optimization to the sequence level using importance ratios normalized by the computation allocated to each trajectory:

$$\rho_i(\theta) = \left(\prod_{t=1}^{|y_i|} \frac{\boldsymbol{\pi}_\theta(y_{i,t}|x, y_{i,<t})}{\boldsymbol{\pi}_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})} \right)^{1/|y_i|}, \tag{8}$$

where the $1/|y_i|$ exponent normalizes for differences in allocated computation. Using the same group-based advantage $\hat{A}_i$ as GRPO, the objective is computed once per sequence.

C.3 Count-Based Exploration Background

In this section, we provide classical background motivating the coverage perspective in Section 3, rather than a theoretical derivation of LINE. The exploration-exploitation tradeoff is a long-standing challenge in RL literature. This becomes more pronounced in LLM reasoning tasks due to the vast state-action space with reward sparsity, where existing methods often suffer from deficient exploration and poor sample efficiency. A classic, theoretically justified exploration principle is *optimism in the face of uncertainty*, which augments the reward estimates of less-visited states/actions with an exploration bonus proportional to their uncertainty. In the minimal multi-armed bandit (MAB) setting, the well-known upper-confidence bound (UCB) algorithm (Auer, 2002) chooses the action a_t according to:

$$a_t = \arg\max_{a \in \mathcal{A}} \hat{R}_t(a) + \sqrt{\frac{2 \log t}{n_t(a)}}, \tag{9}$$

where $\hat{R}_t(a)$ and $n_t(a)$ are the reward estimate and visitation count for action a at time t. The UCB algorithm in Eq. 9 has logarithmic regret in the stochastic multi-armed bandit setting. Let $L(T)$ denote cumulative regret. Specifically, for rewards in $[0, 1]$ and a unique optimal action a^*, let $\Delta_a = R(a^*) - R(a) > 0$ denote the gap of a suboptimal action. The standard UCB1 bound (Auer, 2002) is

$$\mathbb{E}[L(T)] \leq \sum_{a:\, \Delta_a > 0} \left(\frac{8 \log T}{\Delta_a} + \left(1 + \frac{\pi^2}{3}\right) \Delta_a \right), \tag{10}$$

and therefore grows logarithmically in T for fixed gaps. Subsequent studies (Bellemare et al., 2016; Tang et al., 2017) extend this principle to MDPs by counting state-action pairs $n(s, a)$ and adding a bonus reward to encourage exploring less-visited pairs as

$$R'(s, a) = R(s, a) + \frac{\beta}{\sqrt{n(s, a)}}, \tag{11}$$

which is shown to be formally near-optimal within the probably approximately correct (PAC-MDP) framework (Strehl & Littman, 2008; Kolter & Ng, 2009).

D Experimental Details

D.1 Template Prompt

We adopt the following template for all experiments involving Qwen models, building upon the Qwen3 template for Qwen3-4B-Base and the Qwen-Nothinking template for Qwen3.

Qwen3 Template

<|im_start|>user
{problem} Let's think step by step and output the final answer within \boxed{}.
<|im_end|>
<|im_start|>assistant

Qwen3-NoThinking Template

<|im_start|>user
{problem} Let's think step by step and output the final answer within \boxed{}.
<|im_end|>
<|im_start|>assistant
<think>

</think>

For training the Llama-OctoThinker models, we adopt the original prompt in Wang et al. (2025b) to ensure performance.

OctoThinker Template

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. User: You must put your answer inside \boxed{} and Your final answer will be extracted automatically by the \boxed{} tag.
{problem}
Assistant:

D.2 Implementation of RL

In this section, we describe the RL training setup in detail. We implement all RL methods using the Verl framework. GRPO uses the standard group-normalized token-level objective, while GRPO-CH uses the same objective with a higher upper clipping bound. GSPO uses the sequence-level importance ratio described in Appendix C.2. For the reward baselines, the cosine baseline applies the correctness-conditioned compute shaping from Yang et al. (2025b); DIVER-MIX combines the outcome reward with its response-diversity signal; and Overlong adds a length-dependent reward to the GRPO-CH objective. The Overlong+Top20 variant applies that reward only to the top 20% highest-entropy tokens. We disable group filtering for both Overlong variants. Our method adds the computation incentive and redundancy penalty to the corresponding RL objective; the default redundancy signal is a repeated 10-gram count. For the cosine baseline, we use $\mathrm{CosFn}(t, T, a, b) = a + (b-a)(1+\cos(\pi t/T))/2$ with $(r_0^c, r_B^c, r_0^w, r_B^w) = (0.6, 1.0, 0, -0.5)$. For a controlled semantic alternative, we replace the default surface signal with semantic similarity: each trajectory is split into chunks, represented with Qwen3-Embedding-0.6B, and penalized when chunks are too similar. We evaluate this semantic score only with GSPO to isolate the redundancy implementation. It improves over GSPO on the in-domain average (51.4 versus 49.4), but is more sensitive to chunking and similarity thresholds and is less stable during RL. We therefore use the n-gram score in the main experiments; the semantic result is reported in Table 1. For the main RL runs, we use the shared optimization settings in Table 5; baseline-specific reward definitions are given above. We do not employ entropy regularization or KL-based losses. The computation-incentive coefficient

Table 5: Shared optimization settings for the main reinforcement learning runs.

Hyperparameter	**Value**
Optimizer	AdamW
Policy learning rate	$1e^{-6}$
Training batch size	128 prompts
Samples per prompt	8
Mini-batch size	32 prompts
Policy updates per rollout	16
Prompt context budget	1024
Generation compute budget	8192
Rollout temperature	1.0
Computation-incentive coefficient (η)	$0.3/9000$
Repetition threshold (Θ)	10
Redundancy penalty weight (β)	0.6

$\eta = 0.3/9000$ normalizes the computation incentive over the target range induced by the 8k compute response cap and the reference-relative extension. We use $\Theta = 10$ and $\beta = 0.6$ as the default redundancy control settings in the main experiments.

D.3 Compute resources

To support reproducibility, each main training run used 4 × H100 GPUs with 140 GB GPU memory per GPU. The GSPO baseline required approximately 72 hours, corresponding to 288 GPU-hours.

Table 6: **State coverage at matched maximum generation budgets on Qwen3-4B-Base.** Each entry reports LINE/GSPO. DNC and DEC denote distinct 10-gram and distinct equation counts, respectively.

Budget	DNC	DNC (normalized)	DEC	DEC (normalized)
4k	1757.1 / 1161.2	531.71 / 482.80	72.6 / 52.0	22.45 / 24.07
8k	2534.8 / 1510.4	519.31 / 460.29	100.4 / 64.6	21.56 / 22.90
16k	2856.9 / 1709.0	515.69 / 443.96	110.6 / 72.0	21.27 / 22.25
32k	2894.5 / 1795.0	514.22 / 437.05	111.5 / 76.1	21.22 / 22.00

Table 7: **Test-time scaling accuracy across inference budgets on Qwen3-4B-Base (%).** Accuracies are averaged over five in-domain test sets.

Method	4k	8k	12k	16k	24k	32k
GRPO	42.1	43.3	43.6	43.6	43.6	43.6
GSPO	45.4	48.2	49.0	49.2	49.4	49.4
GRPO-CH	44.3	48.9	49.8	49.9	49.9	49.9
GRPO + LINE	43.6	46.6	46.9	46.9	46.9	46.9
GSPO + LINE	46.4	52.6	53.6	53.8	53.8	53.8
GRPO-CH + LINE	44.7	50.7	52.1	52.3	52.5	52.6

GSPO + computation incentive required approximately 240 hours, corresponding to 960 GPU-hours, while GSPO + LINE required approximately 100 hours, corresponding to 400 GPU-hours. In these runs, LINE used fewer GPU-hours than training with the computation incentive alone while still improving performance.

E Additional Results

E.1 State coverage at matched budgets with compute normalization

To assess whether additional computation corresponds to higher measured coverage, we evaluate the best GSPO and GSPO+LINE checkpoints under the same maximum generation budgets. Table 6 reports both absolute counts and compute-normalized counts.

LINE has higher absolute DNC and DEC at every matched budget. Its normalized DNC is also consistently higher, while normalized DEC remains close to the GSPO baseline. These results provide budget-matched support for broader measured coverage, especially at the lexical level, while keeping the coverage metrics as analysis proxies rather than direct semantic objectives.

E.2 Test-time scaling across inference budgets

Table 7 reports the complete budget-conditioned accuracy results underlying Figure 6. Figure 10 provides a supplementary budget-forcing comparison for the baselines and GSPO + LINE. At 32k, LINE improves over the corresponding GRPO, GSPO, and GRPO-CH baselines by 3.3, 4.4, and 2.7 percentage points, respectively.

E.3 Extension to more models

To examine whether LINE generalizes beyond the primary Qwen3-4B-Base setting, we further evaluate it on two additional model settings: Qwen3-4B and Llama-OctoThinker-3B. These models differ from the main testbed in both model family and training stage. Qwen3-4B represents a post-trained non-thinking model, while Llama-OctoThinker-3B is a reasoning-oriented model with a different pretraining and mid-training trajectory. This allows us to test whether LINE is tied to a specific initialization or can serve as a more general recipe for encouraging additional computation.

As shown in Table 8, LINE consistently improves in-domain mathematical reasoning. On Qwen3-4B, applying LINE on top of GSPO improves the in-domain average from 68.2 to 70.2, with particularly clear gains on AMC and AIME 2025. The out-of-domain average also increases substantially from 65.4 to 71.7, mainly driven by improvements on ARC-c and MMLU-Pro. On Llama-OctoThinker-3B,

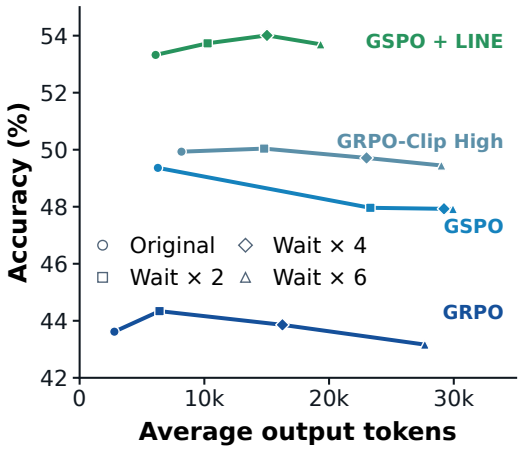


Figure 10: **Budget-forcing comparison with GSPO + LINE.** The figure compares accuracy as a function of average output tokens for the RL baselines and GSPO + LINE.

Table 8: **Evaluation on Qwen3-4B and Llama-OctoThinker-3B.** AIME 24/25 denotes AIME 2024 / AIME 2025, reported as x/y. **Bold** indicates the best result within each model block. Gains of our methods over the corresponding GSPO baselines are marked in blue.

Model	In-Domain Performance					Out-of-Domain Performance			
	MATH	Olympiad	AMC	AIME 24/25	Avg.	ARC-c	GPQA	MMLU-Pro	Avg.
Qwen3-4B									
Base	82.8	51.9	60.4	24.2/19.4	47.7	74.7	37.9	59.4	57.3
GSPO	94.0	**68.1**	82.0	54.2/42.5	68.2	87.5	**52.5**	56.2	65.4
GSPO + LINE (n-gram)	**94.4**	67.0	**85.3**	**57.7/46.4**	**70.2** (+2.0)	**92.1**	52.0	**70.9**	**71.7** (+6.3)
Llama-OctoThinker-3B									
Base	23.4	9.0	10.4	1.1/0.6	8.9	37.6	20.7	11.5	23.3
GSPO	55.8	23.1	28.2	3.8/2.3	22.6	**66.2**	24.7	27.8	**39.6**
GSPO + LINE	**60.8**	**28.1**	**30.3**	**4.5/4.4**	**25.6** (+3.0)	52.0	**27.8**	**38.0**	39.3 (-0.3)

LINE improves the in-domain average from 22.6 to 25.6, suggesting that the recipe remains effective even when the base model has a different reasoning prior.

The OOD behavior on Llama-OctoThinker-3B is more mixed: although GPQA and MMLU-Pro improve, ARC-c decreases, leading to a slight drop in the OOD average. This suggests that LINE reliably improves mathematical reasoning across model families, while its effect on general-domain transfer can depend on the base model and data distribution. Overall, these results support the applicability of LINE across the tested model families and training settings, while also showing that its transfer to OOD tasks can depend on the model and data distribution.

E.4 Scalability Across Model Scales

To verify the scalability of LINE, we evaluate the recipe across different model sizes: Qwen3-1.7B-Base, 4B-Base, and 8B-Base. As summarized in Table 9, LINE consistently delivers performance gains regardless of the base model's capacity.

Specifically, on the in-domain reasoning average, LINE improves the 1.7B model by 7.6%, the 4B model by 4.4%, and the 8B model by 2.5% compared to the GSPO baseline. Notably, on out-of-distribution (OOD) tasks, the 8B model with LINE achieves a 73.4% average accuracy, a 2.9% improvement over the baseline.

Table 9: **LINE on different model sizes. Bold** indicates the best result within each model-size block. Gains over the corresponding GSPO baseline are marked in blue.

Model	In-Domain Performance					Out-of-Domain Performance			
	MATH	Olympiad	AMC	AIME 24/25	Avg.	ARC-c	GPQA	MMLU-Pro	Avg.
Qwen3-1.7B-Base									
Base	51.2	20.7	25.8	3.4/1.7	20.6	54.1	20.2	27.5	33.9
GSPO	70.6	32.3	38.4	8.1/3.8	30.6	**79.4**	28.2	41.0	49.5
+ LINE	**77.0**	**39.6**	**45.2**	**17.5/11.5**	**38.2** (+7.6)	79.1	**33.8**	**45.6**	**52.8** (+3.3)
Qwen3-4B-Base									
Base	66.0	33.2	36.6	8.5/6.9	30.2	66.9	26.3	30.9	41.4
GSPO	85.2	51.7	62.7	26.7/20.5	49.4	88.4	**48.5**	61.5	66.1
+ LINE	**88.8**	**57.2**	**66.2**	**30.5/26.7**	**53.8** (+4.4)	**91.4**	47.5	**63.8**	**67.6** (+1.5)
Qwen3-8B-Base									
Base	67.8	35.3	38.9	10.3/8.5	32.2	58.5	32.3	51.2	47.3
GSPO	89.8	58.8	72.9	34.4/25.8	56.3	93.2	50.0	68.3	70.5
+ LINE	**91.4**	**60.4**	**73.4**	**37.2/31.6**	**58.8** (+2.5)	**94.5**	**55.1**	**70.7**	**73.4** (+2.9)

Table 10: **Evaluation across three independent runs on Qwen3-4B-Base. Bold** indicates the best result. Gains over the GSPO baseline are marked in blue. Mean ± Std is computed over the three independent runs.

Model	In-Domain Performance					Out-of-Domain Performance			
	MATH	Olympiad	AMC	AIME 24/25	Avg.	ARC-c	GPQA	MMLU-Pro	Avg.
Qwen3-4B-Base	66.0	33.2	36.6	8.5/6.9	30.2	66.9	26.3	30.9	41.4
GSPO	85.2	51.7	62.7	26.7/20.5	49.4	88.4	48.5	61.5	66.1
GSPO + LINE *with different random seeds*									
Run 1	**88.8**	57.2	66.2	30.5/**26.7**	53.8 (+4.4)	91.4	47.5	63.8	67.6 (+1.5)
Run 2	87.2	57.0	66.0	30.9/24.1	53.0 (+3.6)	91.0	**51.0**	**65.2**	**69.1** (+3.0)
Run 3	87.2	**59.0**	**68.7**	**33.3**/25.2	**54.7** (+5.3)	**91.8**	43.4	63.7	66.3 (+0.2)
Mean ± Std	$87.7_{\pm 0.9}$	$57.7_{\pm 1.1}$	$67.0_{\pm 1.5}$	$31.6_{\pm 1.5}/25.3_{\pm 1.3}$	$53.8_{\pm 0.9}$ (+4.4)	$91.4_{\pm 0.4}$	$47.3_{\pm 3.8}$	$64.2_{\pm 0.8}$	$67.7_{\pm 1.4}$ (+1.6)

E.5 Random Robustness

To evaluate the stability of LINE, we repeat the GSPO+LINE experiment on Qwen3-4B-Base. This setting tests whether the observed improvements are robust to training stochasticity rather than being driven by a single favorable run.

As shown in Table 10, all three runs improve over the GSPO baseline on the in-domain average. The gains range from +3.6 to +5.3 points, with the best run reaching 54.7 compared to 49.4 for GSPO. The improvements are also consistent across the most challenging mathematical benchmarks: all three runs improve AIME 2024 over the GSPO baseline, and all runs improve AIME 2025 by a large margin.

The OOD results exhibit larger variance, which is expected because LINE is primarily designed to improve mathematical reasoning by encouraging additional computation while controlling repetition. Nevertheless, all three runs improve the OOD average over GSPO, with gains ranging from 0.2 to 3.0 percentage points and the best run reaching 69.1. These results indicate that the main in-domain improvement of LINE is stable across independent runs, while OOD transfer is more sensitive.

E.6 Pitfalls of Direct State Coverage Maximization

As discussed in Section 3.3, one possible approach is to directly optimize an empirical state-coverage metric, such as the In-Context Distinct State Count $C_{\text{context}}(\tau)$. In principle, this objective is closely aligned with the count-based exploration perspective: if a trajectory visits more distinct abstract states, it should cover a broader measured state space under the chosen abstraction. However, in practice, directly using such a metric as an RL reward is fragile because it depends on a hand-designed state abstraction function ϕ, for which no perfect choice exists. We instantiate this direct-optimization baseline using distinct n-gram count as a concrete example of ϕ.

As shown in Figure 11, the model initially increases the measured state coverage rapidly. However, this improvement is not accompanied by stable reasoning gains. Instead, training soon collapses: validation accuracy drops, the distinct n-gram ratio deteriorates, and the generated responses become increasingly less aligned with the mathematical task. The model spends computation on superficially varied text without obtaining a valid mathematical solution.

This behavior reflects a reward-hacking problem. Because the state abstraction is based on surface-level textual diversity, the policy can increase the reward by generating unusual or weakly related surface patterns rather than by developing useful solution trajectories. In other words, maximizing the coverage metric itself can decouple state coverage from reasoning quality.

In contrast, LINE avoids directly optimizing a brittle coverage metric. Instead, it encourages additional computation through a compute-based incentive and combines it with a redundancy penalty to suppress degenerate loops. This design does not guarantee semantic coverage, but it provides a more stable training signal by discouraging the most common low-information solution to the computation incentive.

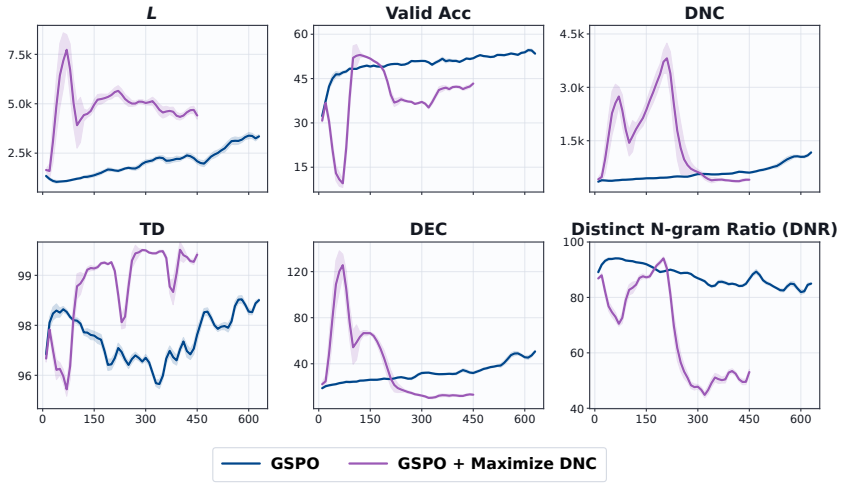


Figure 11: Directly optimizing state coverage leads to reward hacking and training collapse. Maximizing distinct n-gram count increases superficial diversity but degrades reasoning quality, resulting in reduced accuracy and unstable training dynamics.

E.7 SFT as a Complement

We further study how supervised fine-tuning (SFT) interacts with LINE. We aim to understand whether SFT provides signals that are difficult to acquire through RL alone.

Setup. We construct an SFT dataset by randomly sampling 4k problems from the training set and generating Chain-of-Thought (CoT) solutions using GPT-OSS-120B (Agarwal et al., 2025). We fine-tune Qwen3-4B-Base for 3 epochs with a context budget of 4,096. We then apply RL training with and without LINE on top of the SFT-initialized model.

Results. The training dynamics in Figure 12 show the following trends:

1. **SFT improves initialization.** Models initialized with SFT start with higher accuracy and state coverage compared to the base model, indicating that SFT provides a stronger starting point for RL training.
2. **LINE enables continued growth.** While SFT+GSPO improves over the base model, compute use and state coverage tend to plateau early. Adding LINE leads to more sustained growth in both metrics, suggesting that it helps the model develop reasoning trajectories beyond the behaviors induced by supervised data.
3. **A plausible complementary effect.** The combination of SFT and LINE achieves the best performance in our setting. One possible interpretation is that SFT provides useful reasoning patterns that are not easily discovered through RL alone, while LINE helps utilize them through extended reasoning. However, we leave a more detailed investigation of this interaction to future work.

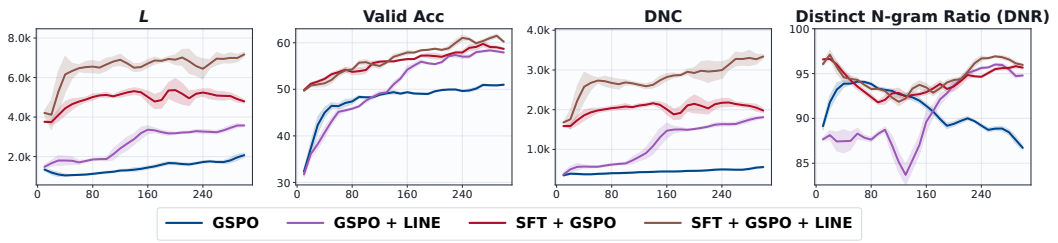


Figure 12: Interaction between supervised fine-tuning (SFT) and LINE on Qwen3-4B-Base. We compare GSPO, GSPO + LINE, SFT + GSPO, and SFT + GSPO + LINE in terms of compute use, validation accuracy, state coverage (DNC), and diversity (DNR). SFT improves initial performance and state coverage, while LINE encourages additional computation and broader state coverage. Their combination achieves better performance.

E.8 Hyperparameter Sensitivity Analysis

We conduct an ablation study to verify that our method's improvements are robust to hyperparameter choices and to understand the impact of constraints.

Impact of Repetition Threshold (Θ). The threshold Θ controls the tolerance for repeated local patterns. A smaller threshold imposes a stricter constraint, while a larger threshold allows more repetition before the penalty is triggered. As shown in Table 11, $\Theta = 10$ achieves the best average performance, outperforming both the stricter setting $\Theta = 6$ and the looser setting $\Theta = 15$.

When Θ is too small, the model can be over-penalized. Mathematical reasoning often contains necessary local repetition, such as restating variables, reusing equations, or applying the same transformation across multiple cases. A strict threshold may incorrectly penalize these valid reasoning patterns, disrupting the logical flow needed for hard problems. This is reflected by the weaker AIME 2025 performance under $\Theta = 6$.

When Θ is too large, the constraint becomes less effective. The model can satisfy the computation incentive by allocating additional compute to repeated local patterns with little new information. This weakens the role of the redundancy penalty in discouraging unproductive computation. The lower average performance under $\Theta = 15$ suggests that some repetition control is necessary for converting additional computation into useful reasoning.

Table 11: The Ablation study of Θ.

	MATH	Olympiad	AIME	AMC	AIME25	Avg.
GRPO w/Clip-Higher	86.4	54.1	25.2	61.8	22.2	49.9
+LINE ($\Theta = 6$)	86.0	55.4	29.2	65.4	21.5	51.5
+LINE ($\Theta = 10$)	**88.8**	54.1	**30.4**	**65.5**	**24.5**	**52.6**
+LINE ($\Theta = 15$)	86.8	**56.0**	28.5	63.9	22.4	51.5

Impact of Repetition Magnitude β We further examine the effect of the penalty magnitude β in the final reward defined in Section 4. Figure 13 compares $\beta = 0.3$ and $\beta = 0.6$. The final accuracy is relatively robust to this choice: both settings converge to similar validation performance. This suggests that the effectiveness of LINE does not rely on a narrowly tuned penalty weight.

The main difference lies in the efficiency of the resulting trajectories. A larger penalty weight reduces compute use and is accompanied by a faster entropy decrease. In our main experiments, we use $\beta = 0.6$ as a default because it preserves the performance gains while reducing unnecessary repetition.

Impact of redundancy n-gram The redundancy penalty in LINE can be viewed as operating on a state abstraction ϕ, where repeated n-grams provide a practical signal for repeated local patterns. Different choices of n-gram size therefore correspond to different granularities of this abstraction.

As shown in Table 12, all n-gram settings consistently improve over the GSPO baseline on in-domain performance. While larger n-grams (e.g., 100 or 512) achieve slightly higher accuracy than smaller ones, the overall performance remains stable across a wide range of choices.

This result suggests that the effectiveness of LINE does not depend on a precise or carefully tuned abstraction. Instead, it is sufficient to employ a coarse signal that can detect and suppress degenerate

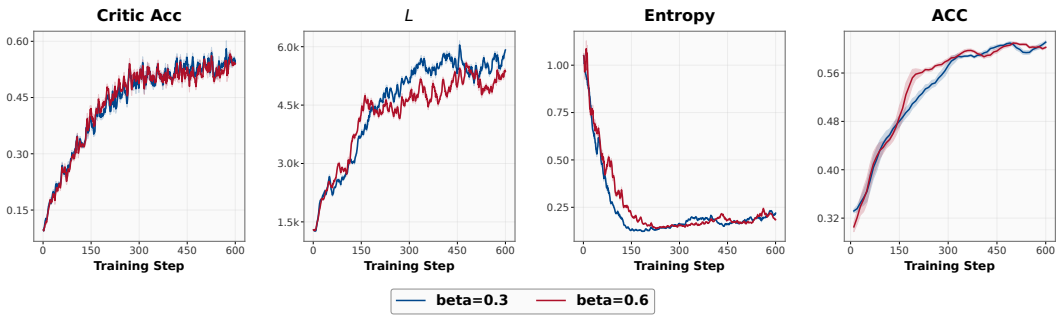


Figure 13: Training dynamics of different β.

Table 12: **Ablation of redundancy n-gram size on Qwen3-4B-Base. Bold** indicates the best result. Gains over the GSPO baseline are marked in blue.

Model	In-Domain Performance					Out-of-Domain Performance			
	MATH	Olympiad	AMC	AIME 24/25	Avg.	ARC-c	GPQA	MMLU-Pro	Avg.
Qwen3-4B-Base	66.0	33.2	36.6	8.5/6.9	30.2	66.9	26.3	30.9	41.4
GSPO	85.2	51.7	62.7	26.7/20.5	49.4	88.4	48.5	61.5	66.1
GSPO + LINE *with different redundancy n-gram sizes*									
10-gram	87.2	**59.0**	68.7	33.3/25.2	54.7 (+5.3)	91.8	43.4	63.7	66.3 (+0.2)
100-gram	**88.8**	58.7	**70.6**	30.4/**26.3**	**55.0** (+5.6)	89.6	**50.0**	65.3	**68.3** (+2.2)
512-gram	88.6	57.3	70.2	**35.8**/22.8	54.9 (+5.5)	**92.0**	46.5	**66.2**	68.2 (+2.1)
GSPO + R_{comp}	88.4	54.5	66.8	29.5/21.8	52.2 (+2.8)	91.0	40.9	63.9	65.2 (-0.9)

repetition patterns. In other words, the abstraction ϕ need not be lossless: approximate representations that capture task-relevant redundancy are sufficient to support improved reasoning, consistent with prior findings (Abel et al., 2016).

Different n-gram sizes illustrate a trade-off between sensitivity and permissiveness. Smaller n-grams more aggressively penalize local repetition, while larger n-grams preserve more flexibility but may miss short loops. However, this trade-off has only a minor impact on overall performance, further reinforcing that the key factor is the presence of an explicit redundancy constraint rather than the exact form of ϕ.

F Case Study

To analyze the internal reasoning dynamics, we follow the principle of Feng et al. (2025) and employ Claude-3.7-Thinking to extract the underlying reasoning graph structure from the model's generated responses. This allows us to visualize and quantify trajectory structure beyond surface sequences or behavior labels.

Qualitative Analysis. We select a representative problem from the AIME benchmark where the baseline fails while our method succeeds. As visualized in Figure 14a, the **GSPO baseline** (Left) follows a linear and shallow reasoning path. It follows a direct derivation and quickly converges to an incorrect answer (324).

In contrast, the model trained with our LINE recipe (Figure 14b) exhibits a significantly richer reasoning topology. Crucially, the trajectory contains a broader set of graph-extracted segments, including an alternative-approach segment ("Alternative Approach: Using Polar Form") and a calculation-error segment ("Calculation Error"). This broader trajectory reaches the correct solution (540).

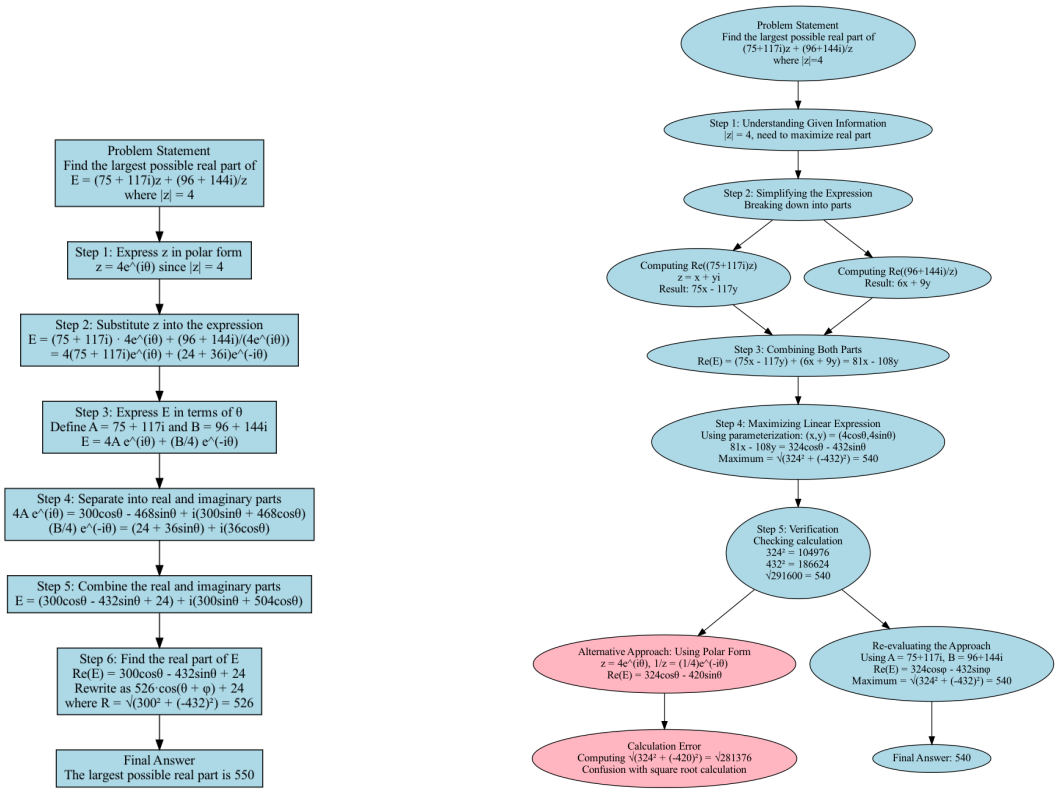
Quantitative Structural Metrics To examine whether this structural difference appears across samples, we compute the average **Depth** (maximum number of edges along a path in the reasoning graph) and **Width** (average branching factor) across the 40 samples from AIME. As shown in Table 13, our method consistently expands the reasoning structure:

- **Increased Depth** ($13.8 \rightarrow 14.75$)**:** The extracted reasoning graph contains more sequential steps along its deepest path.
- **Increased Width** ($2.15 \rightarrow 2.30$)**:** The increased width is consistent with a broader extracted trajectory structure rather than purely linear continuation.

These structural metrics indicate that the trajectories generated with additional computation have broader measured graph structure under this analysis.

Table 13: Exploration width and depth metrics.

Model	Depth	Width
GSPO	13.8	2.15
GSPO + LINE	14.75	2.3



(a) GSPO (b) GSPO + LINE

Figure 14: A case from AIME on GSPO and our recipe models.