LEARNING TO SCALE AT TEST TIME: RL FROM LLM FEEDBACK LOOPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time scaling improves large language model reasoning by allowing models to explore, revise, and aggregate multiple attempts. Reinforcement learning (RL) offers a way to develop these capabilities, but training on extensive interaction trajectories can be costly, and the value of different training data remains poorly understood. In this work, we investigate how to select useful RL contexts and learn from offline solution attempts and feedback. We construct a dataset of 200K offline trajectories by iteratively generating solutions and feedback for 6K mathematics problems. Through extensive post-training experiments on this dataset, we identify important factors such as solution diversity, loop count, and surprisingly irrelevant factors such as orchestration topology. To put these findings into practice, we select trajectories from the latest revision round that satisfy accuracy and diversity constraints for each problem. Using these selected data, we train Qwen3.5-4B without additional supervised fine-tuning, achieving 82.5% accuracy on IMO-AnswerBench with test-time scaling, surpassing DeepSeek-Math-V2 (75.8%) and Gemini Deep Think (IMO Gold, 80.0%). Our trained Qwen3-4B model also matches QED-Nano's accuracy (Qu et al., 2026) with approximately half the inference compute.

1 INTRODUCTION

Test-time scaling improves reasoning by allowing a language model to revisit its work: it can sample multiple attempts, critique previous solutions, revise sequentially over loops, or aggregate complementary reasoning paths. These strategies can substantially improve accuracy without changing the underlying model weights (Muennighoff et al., 2025; Zhu et al., 2025; Bilal et al., 2026). Recent systems such as RSA (Venkatraman et al., 2025), reasoning cache (Wu et al., 2026a), and QED-Nano (Qu et al., 2026) further show that small models can benefit from long-horizon inference scaffolds that repeatedly summarize, revise, and combine intermediate work.

However, test-time scaling greatly increases rollout cost and length. QED-Nano (Qu et al., 2026) reports an increase in average token usage on IMO-ProofBench from 93.7K for single-turn inference to 2.05M with the RSA scaffold, an approximately $22\times$ increase. Therefore, training on these contexts requires careful design to improve efficiency. One approach is to carefully assign and shape reward across loops and branches of collected trajectories (Wei et al., 2025; He et al., 2026; Liao et al., 2026). This provides multiple targeted learning signals from each collected trajectory, helping the policy reinforce useful revisions and reasoning choices.

Another line of work studies sampling strategies to reuse offline data for higher sample efficiency. LUFFY (Yan et al., 2025) combines off-policy demonstrations with on-policy rollouts through a mixed policy objective. ExGRPO (Zhan et al., 2026) retains and prioritizes past rollouts according to correctness and entropy. PrefixRL (Setlur et al., 2026) shows that offline prefixes of successful reasoning traces can make difficult problems learnable. These results suggest that previously generated reasoning trajectories from orchestrated agents can be reused to support RL.

In this work, we take a step forward to understand when and how offline data with feedback loops can help LLMs via reinforcement learning. To this end, we first construct a dataset that can simulate various orchestration topologies. We collect approximately 200K rollout trajectories on 6,000 verifiable mathematics problems using Qwen3-4B-Thinking-2507 as the solver and Qwen3-235B-A22B-Thinking-2507 as the feedback provider. In each loop, the solver generates candidate solutions, and

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.

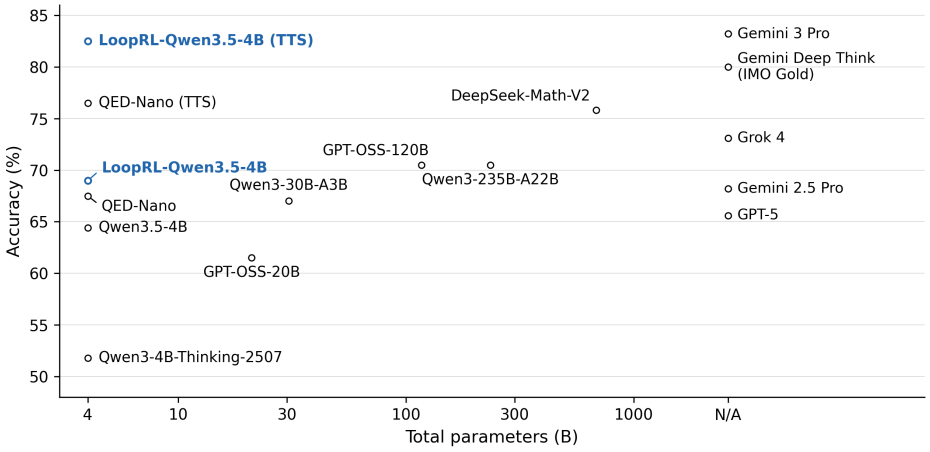


Figure 1: **IMO-AnswerBench accuracy versus model size.** LoopRL-Qwen3.5-4B achieves higher accuracy than other 4B models and several substantially larger models; "TTS" denotes its performance with test-time scaffold.

the feedback provider critiques them to guide the next loop. We use the recorded solution attempts and feedback as RL contexts. Therefore, we can rearrange them into contexts for revision and aggregation, without collecting a corresponding supervised response for every arrangement. This enables simulating multiple orchestration topologies, and allowing us to systematically study the principles for efficient RL training over looped data with different connections.

Our main contribution is the systematic empirical study in Section 3. Through over 100 RL runs, we ablate over how trajectory accuracy, diversity, topology, and loop count can change post-training performance. Our experiments with Qwen3-4B-Thinking-2507 establish three findings:

- **Answer diversity helps identify useful RL contexts.** Among all-incorrect prompts, ones that generate multiple answers are better than ones that have a single wrong answer.
- **Later-loop trajectories provide more useful training data.** After matching the rollout accuracy distribution and answer diversity distribution, training on later-loop trajectories improves accuracy.
- **Training on single-loop data can transfer to aggregation inference.** Training on one later-loop solution–feedback pair outperforms the aggregated contexts, so matching the inference topology is not necessary.

We put these findings into practice in Section 4 with a simple data selection method called LoopRL: choose the latest loop trajectory for each problem subject to bounds on accuracy and answer diversity. With test-time scaling, LoopRL-Qwen3.5-4B achieves 82.5% on IMO-AnswerBench (Figure 1), exceeding the reported scores of much larger models, including Gemini Deep Think (IMO Gold) (80.0%) and the 685B DeepSeek-Math-V2 (75.8%), and approaches Gemini 3 Pro (83.2%). Moreover, LoopRL-Qwen3.5-4B achieves 67.0% on Apex Shortlist (Dekoninck et al., 2026), compared with Gemini 3 Pro's 66.5% (Appendix B.1).

LoopRL improves both training and inference efficiency by reaching comparable performance with less training data and fewer inference tokens. We apply GRPO directly without an additional SFT stage; the RL stage of LoopRL-Qwen3-4B takes 1,632 H200 GPU-hours, compared with QED-Nano's reported 9,216 H100 GPU-hours (Qu et al., 2026). At inference, LoopRL-Qwen3-4B matches QED-Nano's impressive accuracy on IMO-AnswerBench using nearly half the compute.

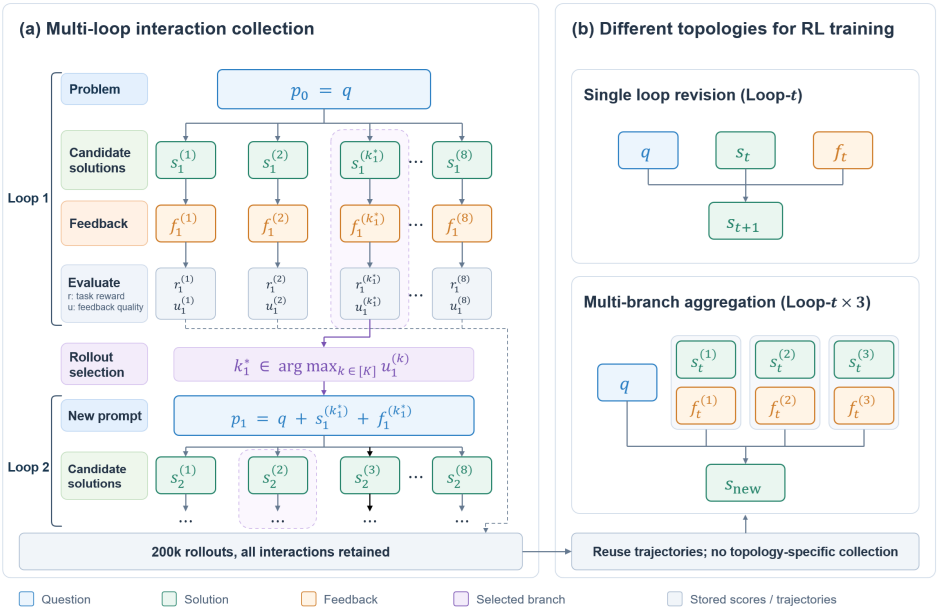


Figure 2: **Multi-loop interaction collection and construction of different training topologies.** (a) Starting from a problem q, each loop samples $K = 8$ candidate solutions s, generates feedback f for each solution, and records its task reward r and feedback-quality score u. The solution–feedback pair with the highest quality, highlighted in purple, is combined with the problem to form the next-loop prompt. Only the selected branch is expanded, while all candidate interactions are retained. (b) The stored records are reused to construct different topologies of RL contexts: the top figure uses a single solution–feedback pair, whereas the bottom one combines three pairs.

2 Loop Dataset Construction

2.1 Multi-loop Interaction Generation

At inference time, a solver can refine its solutions through multiple feedback loops, producing a range of intermediate contexts. To identify which of these contexts are most beneficial for RL training, we construct a dataset from a shared collection of offline interactions to systematically study various data selection choices.

We start by selecting 6K verifiable problems in Nemotron-Math-v2. For every problem, we generate a branching interaction trajectory (Figure 2a). Qwen3-4B-Thinking-2507 samples $K = 8$ candidate solutions at each loop, starting with only the original problem as input. Qwen3-235B-A22B-Thinking-2507 then provides feedback on each candidate given the problem and that solution. We score each feedback using a feedback-quality evaluator $\mathcal{J}$ and select the branch with the highest feedback-quality score for further expansion:

$$u_t^{(k)} = \mathcal{J}\big(q, s_t^{(k)}, f_t^{(k)}\big), \qquad k_t^\star \in \arg\max_{k \in [K]} u_t^{(k)}. \tag{1}$$

where $s_t^{(k)}$ and $f_t^{(k)}$ denote the k-th solution and feedback at loop t. We combine the original problem with the selected solution $s_t^{(k_t^\star)}$ and its feedback $f_t^{(k_t^\star)}$ to form the prompt for the next loop.

Since we perform rollout selection at each loop, choosing exactly one candidate solution and its associated feedback to continue the rollout (purple branch in Figure 2(a)), the resulting structure can be viewed as a sparse tree: each expanded state has K stored children, but only one child is recursively expanded. Consequently, the amount of stored data grows linearly with loop count.

2.2 CONSTRUCTING RL CONTEXTS

We use the problem together with recorded solution attempts and feedback as RL contexts. The stored interactions serve as conditioning information, not supervised targets to imitate: the context is collected offline, while the response used for optimization is sampled on-policy. This separation allows us to provide contexts to the model under different orchestration topologies.

Following our experimental convention, LOOP-0 uses only the problem; LOOP-1 through LOOP-4 each add one complete stored solution and its associated feedback, using the pair selected by feedback quality at the indicated collection loop, as shown in Figure 2b. For example, LOOP-2 contains the solution and feedback generated at collection loop 2.

We also construct inputs containing multiple pairs. LOOP-$t\times3$ concatenates three distinct candidate pairs from collection loop t of the same problem, giving the model parallel alternatives to combine (Figure 2b). Furthermore, trajectories can also be combined across depths, as LOOP-1+2+3 concatenates one pair from each of loops 1, 2, and 3 as RL context.

During RL, the stored context is fixed and the current policy samples fresh solutions from it. A final-answer verifier checks each new solution against the reference answer and assigns reward 1 if correct and 0 otherwise; policy optimization uses these newly generated solutions and rewards. The stored attempts and feedback serve as conditioning information. Consequently, we can train on rearranged contexts without first collecting a supervised target for each arrangement.

3 WHAT MAKES GOOD POST-TRAINING DATA FOR TEST-TIME SCALING?

3.1 EXPERIMENTAL SETUP

Training. We initialize every policy from Qwen3-4B-Thinking-2507 and optimize it with GRPO (Shao et al., 2024). Each prompt generates a group of 8 rollouts, and each rollout contains only one solution. We use a learning rate of 3×10^{-6}, a rollout temperature of 1.0, and a maximum response length of 40,000 tokens. The compared runs use the same optimization settings and differ in the offline contexts supplied to the policy. The eight rollouts in a GRPO group are generated during training; they are separate from the stored candidates used to assemble the input.

Two evaluation topologies. We use *sequential refinement* on HMMT Nov 2025, HMMT Feb 2026, and AIME 2026, with a maximum response length of 60,000 tokens, and *multi-branch aggregation* on the 400 problems in IMO-AnswerBench (Luong et al., 2025), with a maximum response length of 100,000 tokens. In both, the evaluated model generates solutions and its own feedback: each feedback call receives only the problem and the solution being critiqued. The reference answer is used only for scoring and is not supplied to solution or feedback calls.

Sequential refinement: one preceding pair per new solution. Each sampled chain begins with a solution generated from the problem alone. The model then critiques that solution and receives the original problem, its latest solution, and the corresponding feedback as input for a new solution. Each subsequent round repeats this process using only the immediately preceding pair, and separate chains advance independently. At inference loop 3, a chain has produced an initial solution followed by three revisions.

Multi-branch aggregation: three preceding pairs per new solution. At loop 0, the model independently samples eight solutions from the problem and critiques each one, producing a pool of eight solution–feedback pairs. To generate the next loop, we construct eight new inputs, each containing the original problem and three pairs randomly sampled from the preceding pool. The three pairs within an input are distinct, and the subset is sampled independently for each new input, so different inputs can share pairs. The model generates one new solution from each input, then critiques the eight new solutions to form the next pool. Thus the pool retains eight branches at every loop, and the inputs for a new loop are assembled entirely from the preceding loop. This feedback-based aggregation procedure follows the recursive aggregation structure of RSA (Venkatraman et al., 2025).

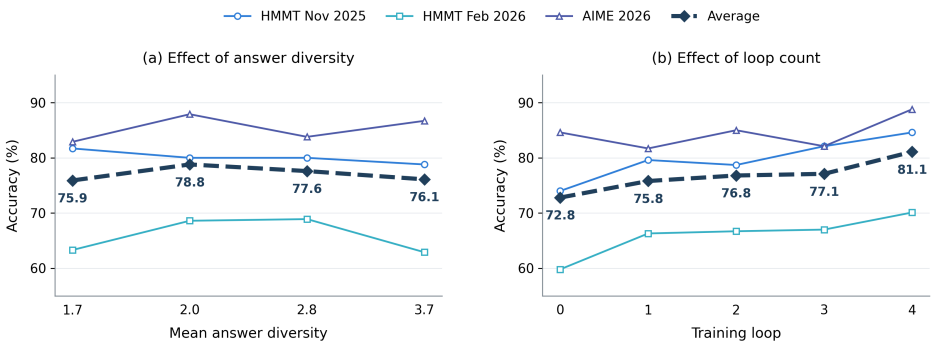


Figure 3: **Effects of answer diversity and loop count on RL performance.** Answer diversity counts distinct canonicalized final answers among eight stored solutions per prompt. (a) Training on prompts with the same accuracy distributions and different mean answer diversities. (b) Training on LOOP-0 through LOOP-4 with matched offline rollout accuracy and answer diversity distributions. LOOP-0 uses problem-only prompts for training, while LOOP-1 through LOOP-4 concatenate each problem with a solution–feedback pair from the indicated collection loop, following Section 2.2. Each curve reports accuracy (%) for HMMT Nov 2025, HMMT Feb 2026, AIME 2026, and their average, at inference loop 3.

Scores and reporting conventions. Accuracy is the percentage of evaluated solution candidates whose final answers are correct at the reported inference loop. For IMO-AnswerBench, this averages candidate correctness across eight solutions per problem; it does not require choosing one solution from the pool. In this section, all controlled comparisons report inference loop 3.

Avoid data leakage. For our dataset construction, the source dataset was released no later than December 2025. Our evaluation benchmarks date from late 2025 or 2026; in particular, HMMT Feb 2026 and AIME 2026 postdate these training resources. This temporal separation avoids the risk of data leakage for these two benchmarks.

3.2 ANSWER DIVERSITY IDENTIFIES USEFUL FAILURE STATES

Since prompts with zero rollout accuracy may still become useful for training as the policy improves (Zheng et al., 2026; Wu et al., 2026b), we conduct a fine-grained study of their answer diversity. We measure diversity as the number of distinct final answers among eight sampled solutions. For a prompt p_t, let $\{s_{t+1}^{(k)}\}_{k=1}^{K}$ be its candidate solutions sampled by the offline rollout policy. We define its accuracy as

$$A(p_t) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{V}\left(q, s_{t+1}^{(k)}\right), \tag{2}$$

where $\mathcal{V}$ is a final-answer verifier, returning 1 if the answer is correct and 0 otherwise, and its answer diversity as

$$D(p_t) = \left|\left\{\text{canon}\left(\text{Ans}(s_{t+1}^{(k)})\right) : k \in [K]\right\}\right|. \tag{3}$$

Both statistics are computed from the eight offline continuations generated conditioned on the given context. Here, diversity is computed over canonicalized final answers rather than raw response strings, so that superficial formatting or wording differences are not counted as distinct answers. For example, 0.5 and $1/2$ are treated as the same answer.

Among prompts with zero accuracy, a diversity of one means that all eight sampled solutions repeat the same wrong answer, suggesting collapse to a single erroneous answer mode and limited exploration of alternative answers. We hypothesize that prompts with multiple answers (i.e., $D > 1$), even when all are incorrect, provide more useful contexts for RL.

Answer diversity identifies useful failure trajectories. To test this hypothesis, we construct two subsets from the stored interaction records with zero accuracy and answer diversities of 1 and 2.

Training on the first obtains 75.0% mean accuracy across HMMT Nov 2025, HMMT Feb 2026 and AIME 2026. Training on the second reaches 79.8%, a gain of 4.8 percentage points. Thus, diversity distinguishes the subsequent training value of prompts that accuracy alone cannot separate, supporting the retention of some zero accuracy prompts. These results suggest keeping all incorrect prompts with multiple answers and removing those that repeat a single wrong answer.

The effect of diversity at the same accuracy distribution. Having established that diversity matters among all incorrect prompts, we next examine its effect when the training subsets contain a range of prompt accuracies. We match their prompt accuracy distributions and vary their mean answer diversity (details can be found in Appendix D.5). The values 1.7, 2.0, 2.8, and 3.7 denote the average of diversity across prompts in each subset. As shown in Figure 3a, the four subsets obtain mean accuracies of 75.9%, 78.8%, 77.6%, and 76.1%, respectively. Mean accuracy across the three benchmarks improves as diversity rises from 1.7 to 2.0, but declines with further increases. Higher diversity therefore does not consistently yield better training data, even after accuracy is controlled.

Implications for data selection. These results motivate retaining prompts with multiple answers. In the following experiments, we remove prompts p whose sampled solutions are all correct ($A(p) = 1$) or all yield the same final answer ($D(p) = 1$). Our primary dataset therefore is

$$\mathcal{P}_{\text{train}} = \{p : A(p) < 1 \text{ and } D(p) > 1\}. \tag{4}$$

Excluding prompts with $A(p) = 1$ is motivated by the lack of reward contrast in uniformly correct rollout groups (Yu et al., 2025), and our experiments support excluding those whose samples all repeat the same wrong answer ($A(p) = 0$ and $D(p) = 1$).

3.3 Later-Loop Contexts Consistently Improve RL Performance

Beyond accuracy and answer diversity, feedback loops offer a natural ladder of training contexts, from early solution attempts to later revisions informed by repeated feedback. This ordering allows us to study whether later loops are better contexts for RL. PrefixRL (Setlur et al., 2026) and QuestA (Li et al., 2026) show that conditioning on prefixes of successful traces can adjust problem difficulty and strengthen the learning signal in RL. Since data from different loops may also differ in accuracy and diversity, we match the loop-specific training subsets on rollout accuracy distribution and answer diversity distribution before comparing loop count. Detailed accuracy and diversity distributions can be found in Appendix D.6.

Figure 3b shows a consistent improvement as training data are drawn from later loops. Problem-only training (Loop-0) achieves 72.8% mean accuracy across the three main benchmarks. Accuracy rises to 75.8%, 76.8%, 77.1%, and 81.1% for Loop-1 through Loop-4, respectively. The best result, obtained with Loop-4, exceeds problem-only training by 8.3 percentage points.

The advantage of later-loop trajectories persists after matching rollout accuracy and answer diversity, suggesting that these aggregate statistics do not fully capture a state's training value. One possible explanation is that later-loop trajectories contain more instances of reflection on earlier reasoning, providing useful context for learning further revisions. These results support considering loop count alongside accuracy and diversity when selecting offline training data.

3.4 Single-Pair Training Supports Multi-Branch Aggregation Inference

The preceding experiments show that later-loop trajectories provide more useful training signals. We next study how to organize these contexts into training prompts. RSA includes aggregation prompts during training to match the form of its inference inputs (Venkatraman et al., 2025). We examine whether such topology alignment is necessary.

We group the four configurations in Table 1 into two categories, following Section 2.2. *Single-pair contexts* combine the problem with one solution–feedback pair from collection loop 2 (Loop-2) or loop 4 (Loop-4). *Aggregation contexts* combine the problem with three pairs: parallel candidates from loop 2 (Loop-2$\times$3), or successive pairs from loops 1, 2, and 3 (Loop-1+2+3). We evaluate every trained policy under two inference topologies (Section 3.1): *sequential refinement* on the three HMMT/AIME benchmarks and *multi-branch aggregation* on IMO-AnswerBench.

Table 1: **Training context and performance across different inference topologies.** All training contexts include the original problem. Single-pair contexts (LOOP-2 and LOOP-4) add one solution–feedback pair from the indicated collection loop; Aggregation contexts add three pairs, drawn either from loop 2 (LOOP-2×3) or one each from loops 1, 2, and 3 of the same problem (LOOP-1+2+3). Scores are accuracies (%) at inference loop 3. At inference, HMMT Nov 2025, HMMT Feb 2026, and AIME 2026 apply sequential refinement, where each new solution is revised from one preceding pair per loop; IMO-AnswerBench applies aggregation, which uses eight candidate solutions per loop and three preceding pairs to generate each new candidate. Average is the mean of all four benchmark scores.

	Sequential refinement			Aggregation	
Training context	HMMT25Nov	HMMT26Feb	AIME26	IMO-AnswerBench	Avg.
Single-pair contexts					
LOOP-2	78.7	66.7	85.0	71.1	75.4
LOOP-4	**84.6**	**70.1**	**88.8**	**72.0**	**78.9**
Aggregation contexts					
LOOP-2×3	74.2	65.2	85.4	70.5	73.8
LOOP-1+2+3	78.3	65.9	84.2	71.9	75.1

Table 1 shows that aggregating more pairs does not consistently improve performance on either evaluation. At a fixed loop count, adding two parallel pairs reduces the four-benchmark average from 75.4% to 73.8%. On IMO-AnswerBench, whose inference inputs contain three pairs, LOOP-2×3 reaches 70.5%, compared with 71.1% for single-pair LOOP-2. Using successive pairs in LOOP-1+2+3 yields 71.9% on IMO-AnswerBench and a 75.1% four-benchmark average. Single-pair LOOP-4 achieves the highest reported score on each benchmark and the highest average, 78.9%; its IMO-AnswerBench score of 72.0% is close to that of LOOP-1+2+3.

These comparisons show that matching the topologies in training and inference is not necessary for effective aggregation. A single pair from a later collection loop supports both sequential refinement and multi-branch inference, motivating its use in our final training dataset.

4 PUTTING INTO PRACTICE: TRAINING FOR TEST-TIME SCALING

Building on the controlled study in Section 3, we now combine its findings into a practical training recipe and demonstrate its performance with extended test-time scaling. We favor later-loop trajectories with moderate answer diversity and train on single solution–feedback pairs. We apply this recipe to Qwen3-4B-Thinking-2507 and Qwen3.5-4B using the same selected data, and evaluate accuracy and inference efficiency over additional inference loops.

4.1 DATASET CONSTRUCTION

Our controlled experiments show that later-loop trajectories provide more useful RL contexts. We combine this preference with selection based on both statistics: a context is eligible if its rollout accuracy satisfies $A \leq 0.5$ motivated by previous works (Li et al., 2026; Toshniwal et al., 2026) and its answer diversity satisfies $1 < D \leq 4$, further restricting the prompt pool in Equation 4. Applying these criteria only at the latest loop would exclude an entire problem whenever its final-loop context fails the filters, even if an earlier context meets them. Our goal is to retain as many distinct problems as these criteria allow, while preferring later-loop contexts.

Therefore, we first retain prompts that satisfy these criteria from loop 4, then consider loop 3 only for problems not yet retained, and repeat this process for loops 2 and 1. Each problem contributes at most one prompt, drawn from its latest eligible loop; problems with no eligible context in any of the four loops are excluded.

Formally, let $p_t(q)$ denote the LOOP-t context for problem q. Its eligible loop indices are

$$\mathcal{T}(q) = \{t \in \{1, 2, 3, 4\} : A(p_t(q)) \leq 0.5, 1 < D(p_t(q)) \leq 4\}\,. \tag{5}$$

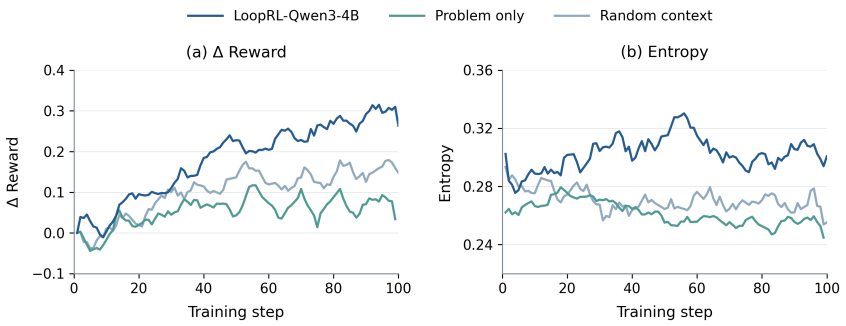


Figure 4: **Training dynamics of selected offline data, problem only prompts, and random context.** Starting from Qwen3-4B-Thinking-2507, we compare GRPO on selected contexts (LOOPRL-Qwen3-4B, dark blue), problem only prompts (problem only, teal), and random contexts (random context, blue gray), using a batch size of 64. All three setups use the same problem set, and random context augments each problem with a solution–feedback pair from a randomly chosen collection loop. Panel (a) presents Δ reward, defined as the mean reward at each step minus the initial reward. Panel (b) presents policy entropy. These curves are smoothed by a centered five-point moving average.

For every problem with at least one eligible state, we select

$$t^\star(q) = \max \mathcal{T}(q), \qquad \mathcal{P}_{\text{selected}} = \left\{ p_{t^\star(q)}(q) : \mathcal{T}(q) \neq \varnothing \right\}. \tag{6}$$

This procedure favors later interaction contexts while retaining earlier ones when the later ones fail the selection criteria. The resulting dataset contains only 5.2K prompts, and combines the benefits of loop count with remaining accuracy headroom and a bounded range of answer diversity.

4.2 TRAINING DYNAMICS

To ablate over the effect of curated dataset, we run three parallel training starting from Qwen3-4B-Thinking-2507 with exactly the same underlying problem set (Figure 4). "Problem only" uses the problem statement alone. "Random context" adds a stored solution–feedback pair by randomly choosing one collection loop for each problem. LOOPRL-Qwen3-4B is trained on P_{selected}, with the context for each problem chosen according to the selection rule above.

The training curves show a progression from problem only to random context and then to LOOPRL. In Figure 4a, problem-only training yields modest improvements over its initial reward. Adding a randomly chosen solution–feedback pair increases these reward gains, while selecting the pair with LOOPRL yields the largest overall gains. This progression suggests that providing offline interaction context helps learning, and that choosing which context to provide brings a further benefit.

Figure 4b shows an overall increase in policy entropy under LOOPRL-Qwen3-4B, while entropy under problem only and random context training trends downward. Higher policy entropy reflects a less concentrated distribution over next-token choices, leaving more room to sample alternative continuations during optimization. At step 100, LOOPRL-Qwen3-4B also achieves a higher mean evaluation accuracy of 81.4% across HMMT Nov 2025, HMMT Feb 2026 and AIME 2026, while problem only and random context yield 78.1% and 79.9%, respectively.

We then continue our training run until step 160. Our RL stage takes 68 hours on 24 H200 GPUs, totaling 1,632 GPU hours. For comparison, QED-Nano reports four days of RL training on 96 H100 GPUs, totaling 9,216 GPU hours. Full training curves of LOOPRL-Qwen3-4B can be found in Appendix B.3.

4.3 INFERENCE PERFORMANCE

We evaluate our model on the IMO-AnswerBench using the multi-branch inference procedure described in Section 3.1, while QED-Nano uses its summary-based test-time scaling configuration.

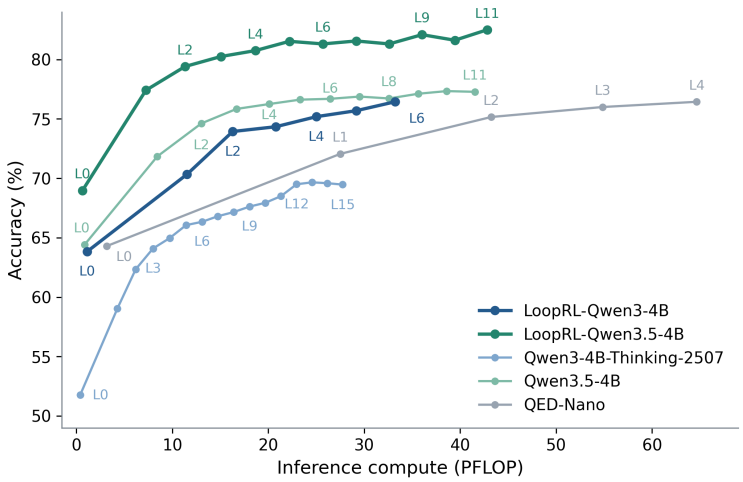


Figure 5: **Inference accuracy versus compute on IMO-AnswerBench.** Each point represents an evaluated inference loop; labels L0, L1, etc. denote loops 0, 1, and so on. LoopRL-Qwen3-4B and LoopRL-Qwen3.5-4B are trained from Qwen3-4B-Thinking-2507 and Qwen3.5-4B, respectively, with GRPO on the selected offline data. LoopRL-Qwen3.5-4B achieves the highest accuracy of 82.5% with a relatively low total computational cost of 42.8 PFLOP.

We extend inference beyond the three loops used in the controlled comparisons and plot accuracy against cumulative inference compute (Figure 5).

Efficient RL on Qwen3-4B-Thinking-2507. Both our trained model LoopRL-Qwen3-4B and the base model improve with additional inference loops, with our model achieving higher accuracy at every inference loop. At inference loop 3, LoopRL-Qwen3-4B reaches 74.3%, compared with 64.1% for the base model. Extending inference to loop 6 raises our model's accuracy to 76.4% at 33.2 PFLOP. QED-Nano reaches the same accuracy at 64.6 PFLOP, so our model uses 48.6% less compute. Furthermore, these gains are achieved by applying GRPO directly to Qwen3-4B-Thinking-2507, without an additional SFT stage.

Transfer to Qwen3.5-4B. To test whether the selected data can benefit a different model, we also train Qwen3.5-4B with GRPO on the same $\mathcal{P}_{\text{selected}}$. These data were constructed from Qwen3-4B-Thinking-2507 solution attempts and Qwen3-235B-A22B-Thinking-2507 feedback, and are reused for Qwen3.5-4B training without regenerating the offline interactions. The trained model LoopRL-Qwen3.5-4B outperforms its base checkpoint at every inference loop while maintaining a comparable compute budget. It reaches 82.5% accuracy at 42.8 PFLOP in loop 11, compared with 77.3% for the base model at the same loop. These results show that the selected offline data remain useful for training a model different from the one that generated the solution attempts.

5 Conclusion

We studied how to select offline data for RL while sampling the completions used for optimization from the current policy. Our interaction dataset makes it possible to compare training contexts. The experiments show that answer diversity distinguishes useful failure states without needing to be maximized, that later-loop trajectories remain more effective after controlling for accuracy and diversity, and that training on a single solution–feedback pair can transfer to multi-branch inference.

To demonstrate these findings in practice, we select the latest eligible state for each problem under accuracy and diversity constraints and train models for extended test-time scaling. The resulting Qwen3.5-4B model reaches 82.5% accuracy on IMO-AnswerBench; the trained Qwen3-4B-Thinking-2507 model matches QED-Nano's accuracy with 48.6% less inference compute. Extending the analysis to other model scales and tasks, and adapting data selection as the policy changes, are directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools for enhancing the clarity and readability of the manuscript, detecting potential typographical errors, assisting with literature retrieval and discovering of related work, generating synthetic data sets, cleaning and reformatting dataset, and implementing methods. We have not used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, supporting qualitative and thematic data analysis, and interpreting results. Helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work. All outputs produced by the LLMs, including manuscript text, code implementations, and synthetic data, underwent thorough human review, verification, and revision by the authors. No AI-generated content was directly adopted without careful inspection and modification. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the LOOPRL dataset and the LOOPRL-Qwen3-4B and LOOPRL-Qwen3.5-4B model checkpoints on Hugging Face. Problem selection and feedback-quality evaluation are detailed in Appendices D.1 and D.2, respectively. Appendix D.3 specifies the training objective, rewards, and hyperparameters, while Appendix D.4 describes the evaluation procedures and inference hyperparameters. Prompt templates are provided in Appendix E.

REFERENCES

Ahsan Bilal, Muhammad Ahmed Mohsin, Muhammad Umer, Lena Trigg, Ali Subhan, Muhammad Ali, and Dean F Hougen. Refining over resampling: Test-time self-correction for llm reasoning. *arXiv preprint arXiv:2608.05643*, 2026.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. *arXiv preprint arXiv:2605.00674*, 2026.

Wei Du, Shubham Toshniwal, Branislav Kisacanin, Sadegh Mahdavi, Ivan Moshkov, George Armstrong, Stephen Ge, Edgar Minasyan, Feng Chen, and Igor Gitman. Nemotron-math: Efficient long-context distillation of mathematical reasoning from multi-mode supervision. *arXiv preprint arXiv:2512.15489*, 2025.

Bowei He, Yankai Chen, Xiaokun Zhang, and Xue Liu. Branching policy optimization: Sandbox-native language agent reinforcement learning. *arXiv preprint arXiv:2607.14171*, 2026.

Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in llms via question augmentation. In *International Conference on Learning Representations*, volume 2026, pp. 144191–144215, 2026.

Sicong Liao, Zhi Chen, and Yaohua Tang. Tcpo: Turn-level credit policy optimization. *arXiv preprint arXiv:2608.01667*, 2026.

Minh-Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, et al. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35418–35442, 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. 20275–20321, 2025.

Yuxiao Qu, Amrith Setlur, Jasper Dekoninck, Edward Beeching, Jia Li, Ian Wu, Lewis Tunstall, Aviral Kumar, et al. Qed-nano: Teaching a tiny model to prove hard theorems. *arXiv preprint arXiv:2604.04898*, 2026.

Amrith Setlur, Zijian Wang, Andrew Cohen, Paria Rashidinejad, and Sang Michael Xie. Reuse your flops: Scaling rl on hard problems by conditioning on very off-policy prefixes. *arXiv preprint arXiv:2601.18795*, 2026.

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.

Shubham Toshniwal, Aleksander Ficek, Siddhartha Jain, Wei Du, Vahid Noroozi, Sadegh Mahdavi, Somshubra Majumdar, and Igor Gitman. Learning generative selection for best-of-n. *arXiv preprint arXiv:2602.02143*, 2026.

Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, et al. Recursive self-aggregation unlocks deep thinking in large language models. *arXiv preprint arXiv:2509.26626*, 2025.

Quan Wei, Siliang Zeng, Chenliang Li, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, et al. Reinforcing multi-turn reasoning in llm agents via turn-level reward design. *arXiv preprint arXiv:2505.11821*, 2025.

Ian Wu, Yuxiao Qu, Amrith Setlur, and Aviral Kumar. Reasoning cache: Continual improvement over long horizons via short-horizon rl. *arXiv preprint arXiv:2602.03773*, 2026a.

Jiahao Wu, Ning Lu, Shengcai Liu, Kun Wang, Yanting Yang, Bailong Lin, Chen Jason Zhang, Li Qing, and Ke Tang. Train at moving edge: Online-verified prompt selection for efficient rl training of large reasoning model. *arXiv preprint arXiv:2603.25184*, 2026b.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance, 2025. *URL https://arxiv. org/abs/2504.14945*, 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, 2025. *URL https://arxiv. org/abs/2503.14476*, 1(2), 2025.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. In *International Conference on Learning Representations*, volume 2026, pp. 12810–12846, 2026.

Haizhong Zheng, Yang Zhou, Brian Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *Advances in Neural Information Processing Systems*, 38:124321–124346, 2026.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, et al. Scaling test-time compute for llm agents. *arXiv preprint arXiv:2506.12928*, 2025.

A Detailed Results

A.1 Results on IMO-AnswerBench

Table 2 shows the comparison of our models and other leading models on IMO-AnswerBench. Our Qwen3.5-4B model trained on carefully selected contexts achieves 82.5% accuracy with test-time scaling, surpassing much larger models such as DeepSeek-Math-V2 and Gemini Deep Think (IMO Gold) and approaching the reported performance of leading closed-source models such as Gemini 3 Pro (83.2%).

Table 2: **Comparison of our models with open- and closed-source models on IMO-AnswerBench.** The upper block reports our own evaluations, while the lower block shows results reported in prior work (Qu et al., 2026; Luong et al., 2025). LoopRL-Qwen3-4B and LoopRL-Qwen3.5-4B are trained from Qwen3-4B-Thinking-2507 and Qwen3.5-4B, respectively, on the selected contexts $\mathcal{P}_{\text{selected}}$. Parameter counts are total billions; compute is the mean inference cost per problem in PFLOP. TTS denotes test-time scaling: our models use multi-branch aggregation (Section 3.1), and QED-Nano uses its summary-based scaling Qu et al. (2026). Dashes denote unavailable values.

Model	Params. (B)	Compute (PFLOP)	Accuracy (%)
Qwen3-4B-Thinking-2507	4	0.41	51.8
Qwen3.5-4B	4	0.86	64.4
QED-Nano	4	3.19	64.3
LoopRL-Qwen3-4B	4	1.14	63.8
LoopRL-Qwen3.5-4B	4	0.64	69.0
QED-Nano + TTS	4	64.64	76.4
LoopRL-Qwen3-4B + TTS	4	33.21	76.4
LoopRL-Qwen3.5-4B + TTS	4	42.81	82.5
GPT-OSS-20B	21	—	61.5
Qwen3-30B-A3B	30.5	—	67.0
GPT-OSS-120B	117	—	70.5
Qwen3-235B-A22B	235	—	70.5
DeepSeek-Math-V2	685	—	75.8
GPT-5	—	—	65.6
Gemini 2.5 Pro	—	—	68.2
Grok 4	—	—	73.1
Gemini Deep Think (IMO Gold)	—	—	80.0
Gemini 3 Pro	—	—	83.2

A.2 Results on HMMT and AIME

Table 3 reports results of the base and RL-trained models on HMMT Nov 2025, AIME 2026, and HMMT Feb 2026 at inference loop 3; RL improves the three-benchmark average from 69.65% to 81.11% for Qwen3-4B-Thinking-2507 and from 81.88% to 85.62% for Qwen3.5-4B.

Table 3: **Accuracy at inference loop 3 on HMMT and AIME.** All accuracies are percentages. Average is the unweighted mean over the three benchmarks.

Model	HMMT25Nov	AIME26	HMMT26Feb	Average
Qwen3-4B-Thinking-2507	67.7	82.9	58.3	69.7
Qwen3.5-4B	82.9	90.0	72.7	81.9
LoopRL-Qwen3-4B	84.6	87.9	70.8	81.1
LoopRL-Qwen3.5-4B	86.7	92.9	77.3	85.6

B Additional Experimental Results

B.1 Results on Apex Shortlist

We additionally evaluate LoopRL-Qwen3.5-4B on Apex Shortlist (Dekoninck et al., 2026), using the multi-branch aggregation inference similar to IMO-AnswerBench. Figure 6 shows our model reaches 67.0% accuracy, outperforming Gemini 3 Pro[1] (66.5%).

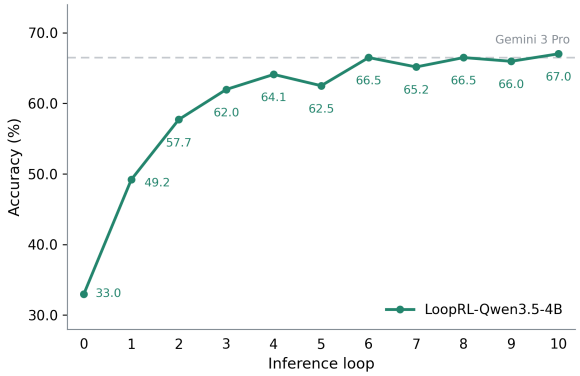


Figure 6: **Accuracy on Apex Shortlist.** LoopRL-Qwen3.5-4B is evaluated at inference loops 0–10. The light gray dashed line marks the 66.5% accuracy of Gemini 3 Pro reported by the MathArena leaderboard.

B.2 Improving Pass@8 on IMO-AnswerBench

To assess how often the candidate set contains a correct solution, we compare our trained Qwen3-4B-Thinking-2507 model with the base model using Pass@8 (Figure 7). We measure this metric separately within each loop: a problem counts as solved if at least one of the eight candidates generated in that loop is correct.

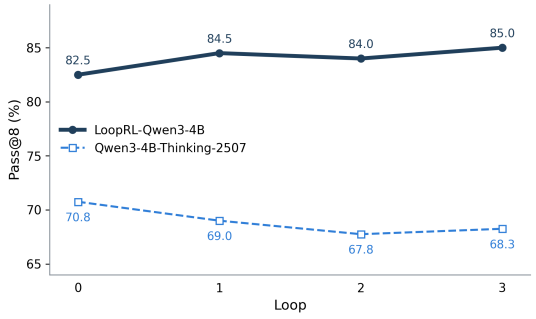


Figure 7: **Pass@8 across inference loops on IMO-AnswerBench.** LoopRL-Qwen3-4B is trained from Qwen3-4B-Thinking-2507, with GRPO on the selected offline data. The pass@8 score is the percentage of problems with at least one correct candidate among the eight generated solutions.

Our model improves Pass@8 at every inference loop. At loop 0, when inference starts from the problem alone, it achieves 82.5%, exceeding the base model's 70.8% by 11.8 percentage points. It maintains Pass@8 between 84.0% and 85.0% over loops 1–3, reaching 85.0% at loop 3. The base model reaches 68.3% at loop 3, leaving a gap of 16.8 percentage points.

[1] https://matharena.ai/?comp=apex--shortlist_2025&view=problem

B.3 Training Dynamics for LoopRL-Qwen3-4B

Figure 8 presents the full logged training trajectory of LoopRL-Qwen3-4B, extending the first 100 steps shown in Figure 4 through step 160. Reward increases over training, and policy entropy is higher toward the end of the run. Response length grows early in training and then fluctuates at a higher level.

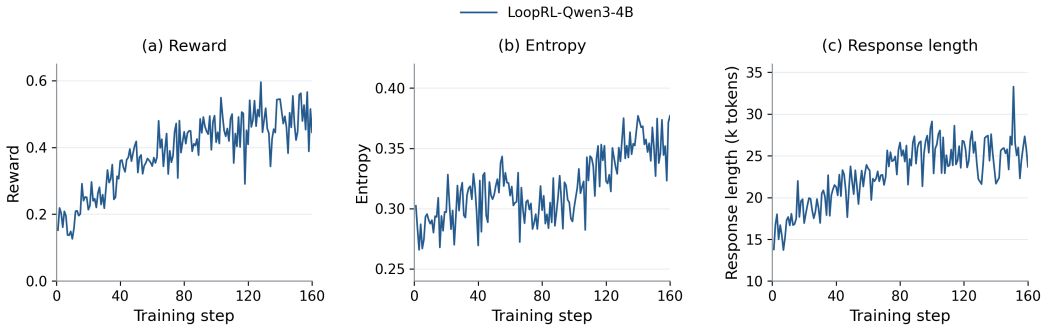


Figure 8: **Full training dynamics of LoopRL-Qwen3-4B.** Starting from Qwen3-4B-Thinking-2507, we apply GRPO to selected contexts $\mathcal{P}_{\text{selected}}$. Panels show (a) mean rollout reward, (b) mean policy entropy, and (c) mean response length in thousands of tokens.

B.4 Training Dynamics for LoopRL-Qwen3.5-4B

Figure 9 presents the full logged training trajectory of LoopRL-Qwen3.5-4B, using GRPO on the same selected offline contexts as LoopRL-Qwen3-4B.

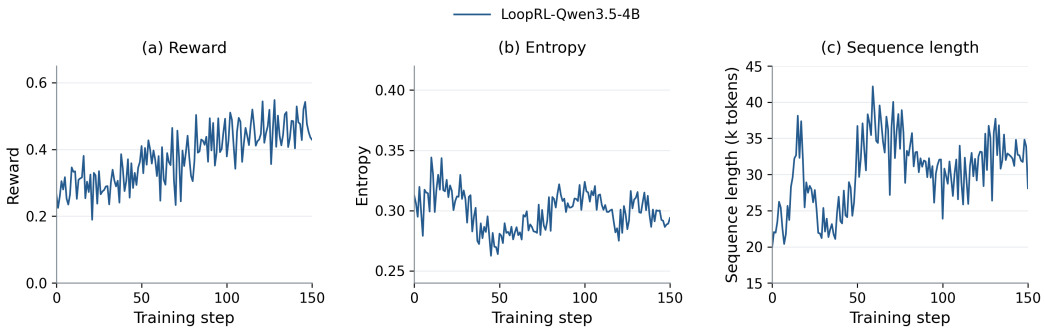


Figure 9: **Training dynamics of LoopRL-Qwen3.5-4B.** Starting from Qwen3.5-4B, we apply GRPO to the selected contexts $\mathcal{P}_{\text{selected}}$. Panels show (a) mean rollout reward, (b) mean policy entropy, and (c) mean actor sequence length in thousands of tokens.

B.5 Training Dynamics for All-Incorrect Contexts

We further examine the comparison between diversities of 1 and 2 among all-incorrect contexts through the training dynamics of Qwen3-4B-Thinking-2507.

Both runs show substantial reward improvements in Figure 10. These results show that all-incorrect contexts can still provide learning signals during GRPO training, and contexts with $D = 2$ provide stronger signals than $D = 1$. Discarding zero-accuracy context would therefore remove potentially useful training data.

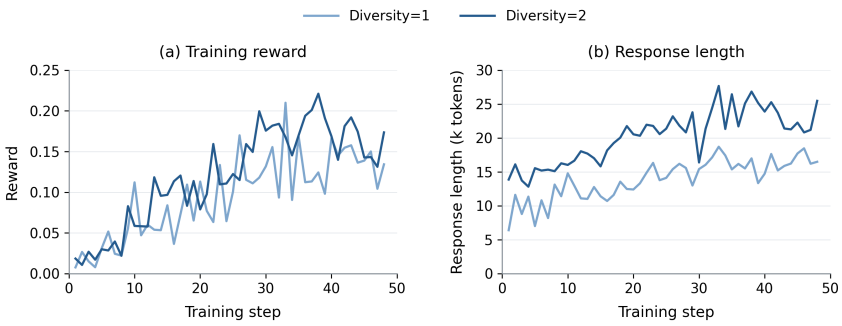


Figure 10: **Training dynamics on all-incorrect offline contexts.** Qwen3-4B-Thinking-2507 is trained on the two zero-accuracy subsets with mean answer diversities of 1 and 2 measured from eight offline continuations per context. (a) Mean training reward. (b) Mean response length in thousands of tokens. Curves show the unsmoothed values at training steps 1–48.

B.6 Loop Count Comparison with Accuracy Control Only

Besides the studies in Section 3.3, we additionally compare training contexts from Loop-0 through Loop-4 while matching their offline accuracy distributions and leaving answer diversity uncontrolled.

Table 4 shows that accuracy does not increase monotonically with training loop count under this accuracy-only control. This comparison motivates controlling both accuracy and diversity when assessing the value of later-loop contexts.

Table 4: **Loop count comparison with accuracy control only.** Rows indicate training context, following Section 2.2. The second and third columns report mean offline accuracy A (in percent) and mean answer diversity D. The final three columns report accuracy (%) on HMMT Nov 2025, AIME 2026, and HMMT Feb 2026 at inference loop 3. Each subset contains 2,730 contexts; offline accuracy distributions are approximately matched, while diversity is not controlled.

Training context	Mean A (%)	Mean D	HMMT25Nov	AIME26	HMMT26Feb
Loop-0	26.22	3.32	68.1	78.8	56.1
Loop-1	26.42	2.55	74.8	78.3	60.6
Loop-2	26.20	2.29	75.4	80.8	60.6
Loop-3	26.34	2.11	79.2	83.8	64.4
Loop-4	26.19	2.05	73.1	79.6	59.5

B.7 Prompt NLL and Length Across Loops

Table 5: **Prompt NLL and length across loops.** We report the mean NLL and the mean prompt length for each loop.

Loop	Mean prompt NLL	Mean prompt length
Loop-1	1.1473	1565.5
Loop-2	1.0463	1837.5
Loop-3	0.9892	1787.9
Loop-4	0.9855	1808.5

We compute the mean negative log-likelihood (NLL) of prompts from loops 1–4 and compare it with mean prompt length. Table 5 shows that mean prompt NLL decreases monotonically from Loop-1 to Loop-4, whereas mean prompt length is not monotonic. This pattern may provide a clue to why contexts drawn from different loops lead to different RL outcomes (Section 3.3). We leave a detailed

investigation of the underlying mechanism and any causal relationship between prompt NLL and RL performance to future work.

C Inference Compute Estimation

We estimate the principal arithmetic operations for text generation from the model architecture. Architecture dimensions follow the official Qwen3-4B and Qwen3.5-4B configurations.

Consider a generation call with $I \geq 1$ input tokens and $O \geq 1$ output tokens. Prefill processes I positions and produces the first output; the following $O-1$ decode steps each process one position. The backbone therefore processes $S = I + O - 1$ positions. We assume an empty cache at the start of each call and reuse the KV cache or recurrent state within that call. No prefix-cache sharing is credited across candidates or requests. Both model families have the form to compute FLOPs:

$$\widehat{F}(I, O) = AS + BS(S+1) + CO, \qquad S = I + O - 1. \tag{7}$$

The three terms count fixed backbone work per processed position (linear term), causal full-attention work (quadratic term), and output-vocabulary projections. Table 6 shows model configurations and the coefficients A, B and C. Separate calls are estimated separately and then summed.

Table 6: Architecture dimensions and FLOPs coefficients for inference. A, B and C are three coefficients for computing FLOPs.

Quantity	Qwen3-4B	Qwen3.5-4B
Hidden size d	2,560	2,560
MLP intermediate size m	9,728	9,216
Total text layers L	36	32
Full-attention layers L_f	36	8
Gated DeltaNet layers L_Δ	0	24
Full-attention query heads h_q / KV heads h_{kv}	32 / 8	16 / 4
Full-attention head dimension d_h	128	256
Query width Q / KV width K	4,096 / 1,024	4,096 / 1,024
Vocabulary size V	151,936	248,320
A: fixed backbone FLOPs per position	7,266,631,680	7,228,030,976
B: causal-attention coefficient	294,912	65,536
C: output-head FLOPs per output token	777,912,320	1,271,398,400

C.1 Full Attention, MLPs, and the Output Head

Let $Q = h_q d_h$ and $K = h_{\text{kv}} d_h$ be the query and key/value projection widths. The three SwiGLU matrices cost $6dm$ FLOPs per layer and position. Query, key, value, and output projections cost $4d(Q+K)$ FLOPs in a full-attention layer; Qwen3.5 additionally uses an output-gate projection costing $2dQ$.

When a position attends to n tokens, the query–key and attention–value products together cost $4Qn$ FLOPs per layer. Counting causally valid pairs gives

$$\frac{I(I+1)}{2} + \sum_{j=1}^{O-1} (I+j) = \frac{S(S+1)}{2}. \tag{8}$$

Consequently $B = 2L_f Q$. The output projection costs $C = 2dV$ FLOPs per generated token. For Qwen3-4B, all L layers use full attention, giving

$$\begin{aligned} A_{\text{Qwen3}} &= L\,[6dm + 4d(Q+K)] \\ &= 7{,}266{,}631{,}680. \end{aligned} \tag{9}$$

Here $B = 2L_f Q = 294{,}912$ and $C = 2dV = 777{,}912{,}320$. This calculation applies to our trained Qwen3-4B-Thinking-2507 model and the Qwen3-4B-based QED-Nano comparison.

C.2 Gated DeltaNet Accounting for Qwen3.5-4B

Qwen3.5-4B has L_f full-attention layers and $L_\Delta = L - L_f$ Gated DeltaNet layers. Let $h_k^\Delta = 16$ and $h_v^\Delta = 32$ denote the key- and value-head counts of each DeltaNet layer, with $d_k = d_v = 128$. Queries and keys are expanded to h_v^Δ value heads for the recurrence. Each head maintains a $d_k \times d_v$ matrix state. With decay and update gates α_t and β_t, the core operations are

$$\bar{H}_t = \alpha_t H_{t-1}, \qquad p_t = k_t^\top \bar{H}_t, \qquad \delta_t = \beta_t(v_t - p_t), \tag{10}$$

$$H_t = \bar{H}_t + k_t \delta_t^\top, \qquad y_t = q_t^\top H_t. \tag{11}$$

They denote state decay, prediction, residual gating, state update, and readout cost, respectively, and cost

$$d_k d_v + 2d_k d_v + 2d_v + 2d_k d_v + 2d_k d_v = 7d_k d_v + 2d_v \tag{12}$$

FLOPs per head and position. The recurrence thus costs

$$F_{\Delta,\mathrm{rec}} = h_v^\Delta(7d_k d_v + 2d_v) = 3{,}678{,}208 \tag{13}$$

FLOPs per layer and position.

The key/query projection width before head expansion is $K_\Delta = h_k^\Delta d_k = 2{,}048$ and the value width is $V_\Delta = h_v^\Delta d_v = 4{,}096$. Counting the projections gives

$$F_{\Delta,\mathrm{proj}} = 2d\left(2K_\Delta + 3V_\Delta + 2h_v^\Delta\right) = 84{,}213{,}760 \tag{14}$$

FLOPs. Let $k_c = 4$ denote the depthwise-convolution kernel size. Convolution over the $2K_\Delta + V_\Delta$ query/key/value channels contributes

$$F_{\Delta,\mathrm{conv}} = 2k_c(2K_\Delta + V_\Delta) = 65{,}536 \tag{15}$$

FLOPs. The total non-MLP cost per layer and position is

$$F_\Delta = F_{\Delta,\mathrm{proj}} + F_{\Delta,\mathrm{conv}} + F_{\Delta,\mathrm{rec}} = 87{,}957{,}504. \tag{16}$$

Including all MLPs and the additional full-attention gates,

$$\begin{aligned} A_{\mathrm{Qwen3.5}} &= 6Ldm + L_f\left[4d(Q+K) + 2dQ\right] + L_\Delta F_\Delta \\ &= 7{,}228{,}030{,}976. \end{aligned} \tag{17}$$

Here $B = 2L_f Q = 65{,}536$ and $C = 2dV = 1{,}271{,}398{,}400$.

D Implementation Details

D.1 Problem selection for building loop dataset

We construct the loop dataset from Nemotron-Math-v2 (Du et al., 2025). We retain problems with simple final-answer formats for which Math-Verify[2] is able to judge the correctness of a generated answer against the reference answer. For example, a numerical answer of 0.5 can be verified as equivalent to a reference answer of $1/2$. This allows us to use Math-Verify for automatic reward computation during RL. This procedure retains approximately 6,000 problems. Moreover, after building the multi-loop dataset, eight offline continuations are generated conditioned on Loop-4 contexts, to provide statistics A and D for Loop-4 contexts.

D.2 Feedback-quality evaluator

We use GPT-5 to evaluate the feedback. This evaluator receives the reference answer along with the problem, solution, and feedback. It gives a score that assesses how specifically and constructively the feedback addresses the solution's reasoning. The score supplies the feedback-quality value in Equation 1. Figure 17 gives the rubric.

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

D.3 Training Objective and Rewards

RL training. We employ the GRPO algorithm (Shao et al., 2024) and eliminate the KL divergence term. Let x be a fixed training context, and sample G fresh responses $y_i \sim \pi_{\text{old}}(\cdot \mid x)$ from the rollout policy. The stored solutions and feedback are part of x and are excluded from the policy loss. The clipped GRPO surrogate uses the token probability ratio

$$\rho_{i,j}(\theta) = \frac{\pi_\theta(y_{i,j} \mid x, y_{i,<j})}{\pi_{\text{old}}(y_{i,j} \mid x, y_{i,<j})}. \tag{18}$$

The training objective is

$$\begin{aligned} \ell_{i,j}(\theta) &= \min\left\{\rho_{i,j}(\theta)\widehat{a}_i,\ \text{clip}(\rho_{i,j}(\theta), 1-\epsilon, 1+\epsilon)\,\widehat{a}_i\right\}, \\ J_{\text{clip}}(\theta) &= \mathbb{E}_{x,\{y_i\}_{i=1}^{G}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{j=1}^{|y_i|}\ell_{i,j}(\theta)\right]. \end{aligned} \tag{19}$$

Here $\widehat{a}_i$ is the outcome advantage shared by response tokens:

$$\widehat{a}_i = \frac{R_i - \text{mean}(\{R_i : i \in [G]\})}{\text{std}(\{R_i : i \in [G]\}) + \delta}, \tag{20}$$

where δ is a numerical stabilizer.

Task reward, truncation, and answer format. Generated answers are judged with Math-Verify. Correct and incorrect answers receive task rewards of 1 and 0. For a rollout that hits the generation limit, we remove its final 3,000 tokens, append the instruction in Figure 18, and generate at most 3,000 new tokens to obtain an answer for reward evaluation. This forced continuation is excluded from the policy loss, and the original rollout receives a truncation penalty of 0.1.

For a response that ends without truncation, we apply a format penalty of 0.1 if it lacks `</think>`, because evaluation consumes the answer text after this delimiter. Let T_i indicate truncation, let F_i indicate that an untruncated response contains the delimiter, and let $\widetilde{y}_i$ be the text used for verification, including a forced answer when needed. The reward is

$$R_i = \mathcal{V}(q, \widetilde{y}_i) - 0.1T_i - 0.1(1 - T_i)(1 - F_i), \tag{21}$$

where $\mathcal{V}$ is the verifier. The format penalty applies only to untruncated responses; the two penalties are not added for the same truncated rollout.

Training hyperparameters. Table 7 shows the hyperparameters for training Qwen3-4B-Thinking-2507 (used in Section 3 and Section 4) and Qwen3.5-4B (used in Section 4).

Table 7: Training hyperparameters for Qwen3-4B-Thinking-2507 and Qwen3.5-4B.

Setting	Qwen3-4B-Thinking-2507	Qwen3.5-4B
Optimization method	GRPO	GRPO
Learning rate	3×10^{-6}	1×10^{-6}
Rollout temperature	1.0	0.8
Top-p	1.0	1.0
Maximum response length (Section 3)	40,000 tokens	—
Maximum response length (Section 4)	56,000 tokens	56,000 tokens
Group size, G (Section 3)	8	—
Group size, G (Section 4)	16	16
Batch size (Section 3)	128	—
Batch size (Section 4)	64	64

D.4 Evaluation Procedure and Hyperparameters

At inference loop 0, the model solves each benchmark problem from the problem alone (Figure 13). Sequential refinement on the HMMT/AIME benchmarks conditions each new solution on the original problem and the immediately preceding solution–feedback pair (Figure 14). The multi-branch

aggregation scaffold on IMO-AnswerBench maintains eight candidates per problem: each new candidate receives three distinct pairs from the preceding pool (Figure 16), with the subset sampled independently for each new input. The evaluated model generates both solutions and feedback (Figure 15).

Table 8: Inference hyperparameters for the controlled studies (Section 3) and extended scaling (Section 4). Controlled studies use only Qwen3-4B. In both settings, values before and after the slash correspond to HMMT/AIME and IMO-AnswerBench, respectively.

	Controlled studies	Extended scaling	
Setting	Qwen3-4B	LOOPRL-Qwen3-4B	LOOPRL-Qwen3.5-4B
Solution token budget	60,000 / 100,000	80,000 / 100,000	80,000 / 100,000
Feedback token budget	40,000 / 80,000	60,000 / 80,000	60,000 / 80,000
Temperature	1.0	1.0	0.8
Top-p	1.0	1.0	1.0

Final-answer verification. At evaluation, we use DeepSeek-V4-Flash-0731 as a final-answer judge, motivated by the complex reference answers in IMO-AnswerBench and HMMT Feb 2026. The judge receives the problem, generated answer, and reference answer. It checks final-answer equivalence, allowing algebraic, numerical, and unordered-set equivalence, and gives no partial credit. An absent or ambiguous final answer is graded incorrect. Figure 19 gives the rubric.

D.5 Data Selection for the Diversity Ablation Study

For the matched-accuracy diversity experiment in Figure 3a, we use LOOP-1 contexts throughout. Each of its four subsets contains 2,582 contexts and has the accuracy distribution in Figure 11.

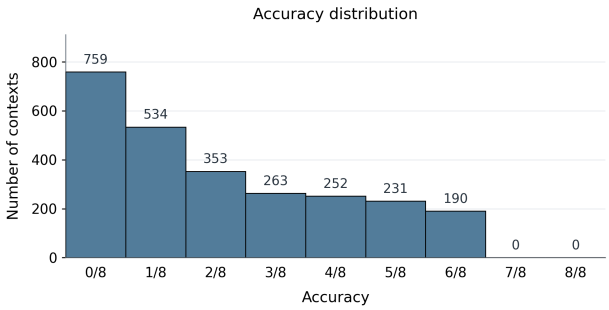


Figure 11: **Accuracy distribution for the diversity ablation.** The four subsets share this distribution, each containing 2,582 contexts.

We construct the four subsets with mean diversity $1.7, 2.0, 2.8$ and 3.7 as follows, subject to the shared accuracy distribution:

- $\overline{D} = 1.7$: Select contexts from the $D = 1$ and $D = 2$ pools.
- $\overline{D} = 2.0$: Select contexts exclusively from the $D = 2$ pool.
- $\overline{D} = 2.8$: Select contexts from the $D = 2$ and $D > 2$ pools.
- $\overline{D} = 3.7$: Select contexts exclusively from the $D > 2$ pool.

D.6 Accuracy and Diversity Controls for the Loop Count Ablation Study

In Section 3.3, we match the training subsets on rollout accuracy distribution and answer diversity distribution. Each loop-specific subset contains 1,628 contexts. Figure 12 shows the accuracy (a)

and diversity (b) histograms matched across LOOP-0 through LOOP-4. Their mean accuracy is $\overline{A} = 0.2645$ and mean diversity is $\overline{D} = 2.5326$.

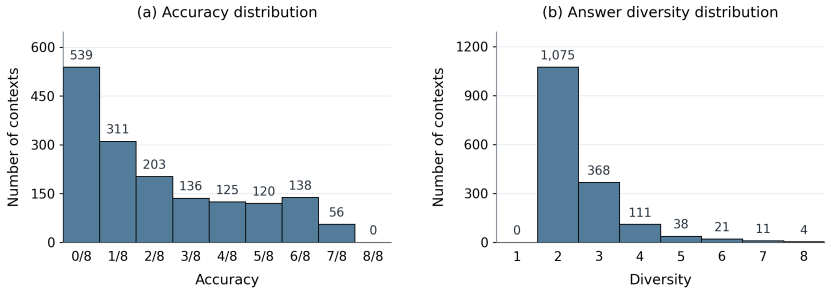


Figure 12: **Accuracy and diversity distributions for the loop-depth ablation.** The subsets for LOOP-0 through LOOP-4 share these distributions, each containing 1,628 contexts. (a) Accuracy distribution, with mean 0.2645. (b) Answer-diversity distribution, with mean 2.5326.

E PROMPT TEMPLATES

Figures 13– 20 provide detailed illustrations of the prompt templates we use.

Problem-Only Generation

Problem:
{problem}

Reason step by step, and put your final answer within \boxed{}.

Figure 13: Prompt template for generating a solution from the problem alone.

Single-Pair Revision

Problem:
{problem}

Previous solution:
{solution}

Feedback on previous solution:
{feedback}

Carefully follow the feedback above. If the feedback identifies any errors, correct them thoroughly. If the feedback confirms the solution is correct, verify the reasoning and reproduce a clean solution. Reason step by step, and put your final answer within \boxed{}.

Figure 14: Prompt template for revising a solution using its feedback.

Feedback Generation

You are a math solution evaluator. Your goal is to carefully analyze the proposed solution and provide constructive, concise feedback—without revealing the final answer.

You are given:

- A Problem Statement
- A Proposed Solution

Instructions
1. If the proposed solution is fully correct
In feedback, first output "The solution is correct." Then confirm correctness by explaining *which reasoning steps are valid* and *why they are logically sound*.

2. If the proposed solution is incorrect or incomplete
In feedback, first output "The solution is incorrect." Then provide **constructive guidance**:

- Highlight any reasoning steps or ideas that are valid.
- Identify the specific steps, calculations, or concepts that are incorrect, unclear, or incomplete.
- Offer hints or guiding questions that could help refine the reasoning.
- DO NOT reveal the final answer or provide a complete corrected solution.

Problem: {problem}

Proposed Solution: {solution}

Figure 15: Prompt template for generating feedback on a proposed solution.

Multi-Branch Aggregation

You are given a {problem_kind} and several candidate solutions with feedback. Some candidates may be incorrect or contain errors. Aggregate the useful ideas and use the feedback to produce a single, high-quality solution. Reason carefully; if candidates disagree, choose the correct path. If all are incorrect, then attempt a different strategy. End with the final result in {format_hint}.

Problem:
{problem}

Candidate solutions and feedback (may contain mistakes):

Solution 1:
{solution_1}

Feedback 1:
{feedback_1}

Solution 2:
{solution_2}

Feedback 2:
{feedback_2}

Solution 3:
{solution_3}

Feedback 3:
{feedback_3}

Now write a single improved solution. Provide clear reasoning and end with the final answer in {format_hint}.

Figure 16: Prompt template for aggregating three candidate solutions and their feedback.

Feedback-Quality Scoring

You are a strict evaluator for math-solution feedback quality. You will receive:

- Problem
- Correct Answer
- Proposed Solution
- Feedback

Score the feedback using TWO criteria:

1) **Judgment correctness:** Check whether the feedback's verdict (correct / incorrect) is consistent with the actual quality of the proposed solution.
2) **Final-answer leakage:** Check whether the feedback directly gives the final correct answer.

Scoring rules:

- judgment_correct: true or false
- reveals_final_answer: true or false
- score: integer from 1 to 5, focused only on feedback quality
  - 5: feedback clearly identifies important process-level issues, gives specific and useful guidance, and is well written.
  - 4: feedback catches meaningful process-level problems and is mostly constructive.
  - 3: feedback is partially helpful, but vague, shallow, or only catches minor issues.
  - 2: feedback is weak, generic, or only loosely related to the solution quality.
  - 1: feedback is poor, unhelpful, or misses the real process-level problems.

Output format:
Return ONLY one-line JSON with this exact schema:
{"judgment_correct": true/false, "reveals_final_answer": true/false, "score": 1-5, "reason": "short reason"}

Problem: {problem}

Correct Answer: {correct_answer}

Proposed Solution: {solution}

Feedback: {feedback}

Figure 17: Prompt template for scoring feedback quality.

Forced-Answer Continuation

Time is up. Given the time I've spent and the approaches I've tried, I should stop thinking and formulate a final answer based on what I already have.
</think>

Figure 18: Forced-answer continuation appended to the retained generation prefix, beginning with a newline. During training, the continuation supplies a final answer for reward calculation and is excluded from the policy loss.

Final-Answer Judge (Part 1)

System Role: Deterministic Mathematical Autograder

You are a precise, automated grading system. Your sole function is to determine if the final answer provided in the Model Solution is mathematically equivalent to the Golden Answer. You must NOT grade the reasoning or steps, only the final result.

1. Grading Guidelines (Equivalence Rules)

Equivalence is mandatory for a correct grade. You must rigorously verify if the answers represent the exact same mathematical value or expression, even if the format differs.

- **Algebraic Equivalence:** e.g., 'n(n+1)/2' is equivalent to 'n^2/2 + n/2'. You must verify the algebra.
- **Numerical Equivalence:** e.g., '1/2' is equivalent to '0.5'; 'sqrt(2)/2' is equivalent to '1/sqrt(2)'.
- **Set/List Equivalence:** Unless specified as an ordered tuple/vector, the order of elements does not matter (e.g., $\{1, 2\}$ is equivalent to $\{2, 1\}$).
- **Partial Credit:** No partial credit is allowed. If the answer is incomplete or partially incorrect, it is incorrect.
- **No Answers:** If no clear, unambiguous final answer can be extracted, the solution must be graded as incorrect.

2. Extraction Protocol

Extract the final answer from the Model Solution. The final answer is the last clearly stated result that directly addresses the problem's question. Look for signals like "the answer is", "therefore", "thus", "hence", or a boxed/highlighted expression. Ignore intermediate steps.

Figure 19: Final-answer judge prompt (part 1): equivalence rules and answer extraction. Continued in Figure 20.

Final-Answer Judge (Part 2)

3. Output Protocol (Strict Compliance Required)

You must execute the task using a two-part structure. Failure to follow this structure will result in task failure.

Part 1: Analysis (Chain-of-Thought)

You MUST perform your analysis within <thinking> tags. Make your thinking concise. This section details your reasoning process and must follow these steps sequentially:

1. **Golden Answer:** State the Golden Answer.
2. **Extracted Model Answer:** State the extracted answer based on the Extraction Protocol. If none found, state "No clear final answer found."
3. **Equivalence Analysis:** Compare the two answers using the Grading Guidelines. Detail the steps taken to verify mathematical equivalence (e.g., simplification, algebraic manipulation). You must actively try to prove they are the same before concluding they are different.
4. **Conclusion:** State the final determination ("Correct" or "Incorrect").

Part 2: Final Grade

Immediately following the closing </thinking> tag, output **ONLY** the final grade.

- If Correct: \boxed{Correct}
- If Incorrect: \boxed{Incorrect}

CRITICAL CONSTRAINT: Do not add any text, explanations, or formatting outside the <thinking> tags or the final \boxed{} output.

Example:
<thinking>

1. **Golden Answer:** 3
2. **Extracted Model Answer:** The solution states "the smallest positive integer n such that there exists infinitely many triples is n = 3". Modified answer: 3
3. **Equivalence Analysis:** 3 = 3. The answers are numerically identical.
4. **Conclusion:** Correct

</thinking>
\boxed{Correct}

4. Input Data
Here is the problem, model solution, and golden answer to grade:

Problem: {Problem_Statement}

Model Solution: {Model_Solution}

Golden Answer: {Golden_Answer}

Figure 20: Final-answer judge prompt (part 2): output protocol, example, and input fields. Continued from Figure 19.