PEARL: Perturbation Efficient Alignment Group Relative Reinforcement Learning

Anonymous ACL submission

Abstract

Reinforcement learning with verifiable rewards (RLVR) has become a widely used approach for improving the mathematical reasoning ability of large language models (LLMs). However, recent studies show that LLMs may solve a math problem in its original form but fail when the same problem is reworded without changing its meaning. This suggests that the model has not fully acquired the underlying reasoning ability and instead relies on the surface form of the problem. In this paper, we propose PEARL, an RLVR framework that uses different versions of the same math problem as structured training signals. PEARL first groups each original problem with its checked variants to build semantic perturbation clusters, and then uses partially solved clusters as the active RL training set. PEARL further performs cluster-level RL optimization by comparing model outputs across different versions of the same mathematical problem. Experiments on mathematical reasoning benchmarks show that PEARL achieves strong results against competitive RLVR baselines while using a small active training set. These results suggest that semantically equivalent variants can serve as useful training units for improving mathematical reasoning in LLMs.

1 Introduction

Recently, large language models have demonstrated remarkable progress in mathematical reasoning, including multistep reasoning and Olympiad level problem solving (Zhang et al., 2025). Alongside this progress, reinforcement learning with verifiable rewards has emerged as an important paradigm for further improving the reasoning capabilities of LLMs (Wen et al., 2025; Chen et al., 2026a). Within this paradigm, methods such as GRPO (Shao et al., 2024) show that group-wise relative comparison can effectively guide reasoning behavior without relying on dense process supervision (Shao et al., 2024). Building on these advances, recent work has improved RLVR from both reward optimization (Yu et al., 2026; Liu et al., 2025; Zheng et al., 2025) and data enhancing (Yang et al., 2025c; Dai et al., 2026) perspectives.

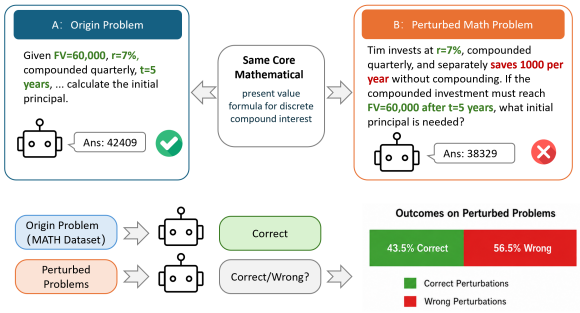

Figure 1: Semantic perturbations can expose unstable mathematical reasoning. On our constructed perturbation set of MATH dataset, only 43.5% of perturbed problems remain correct when their original problems are solved correctly.

Despite these improvements, current mathematical reasoning models still have clear limitations. A model may solve a problem in its original form, but fail when the same problem is rewritten or perturbed without changing its mathematical meaning (Li et al., 2024; Mirzadeh et al., 2025; Huang et al., 2025). As shown in Figure 1, a model may produce different answers for semantically equivalent formulations of the same mathematical problem. On the MATH dataset and its perturbation set, we find that when the model correctly solves an original problem, it remains correct on only 43.5% of its perturbed versions. Such failures suggest that the model may not have fully learned the reasoning process needed to solve the problem. Instead, it may simply have been exposed to similar problems during training and reproduce the corresponding solutions from memory, rather than actually reasoning through the question. A capable reasoning model should derive the answer from the under-

lying mathematical relations, rather than from a specific form of the question.

Semantically equivalent perturbations can help reveal deficiencies in mathematical reasoning. They can also serve as useful training signals for improving this ability. Recent work further suggests that augmented or reformulated samples can contribute not only to data diversity, but also to the optimization process itself. Some studies construct harder or more diverse mathematical problems while preserving answer semantics, while others exploit intermediate reasoning information or sample difficulty to provide more informative training signals (Yang et al., 2025c; Dai et al., 2026) .However, most existing augmentation based approaches still treat each reformulated question as an independent training instance. They fail to capture the inherent connections between equivalent variants, since the original problem and its rewrites share the same mathematical semantics and expected answer. This observation lead to a specific question: can semantically equivalent perturbations be organized as structured training units, allowing RLVR to derive comparative learning signals from different formulations of the same problem and thereby strengthen the model's mathematical reasoning ability?

In this paper, we propose PEARL, an RLVR framework that trains on semantic perturbation clusters. PEARL first performs equivalent semantic perturbation on math problems because RLVR can use answer verification to compare whether different versions of the same problem lead to the same correct answer. Instead of treating each reworded problem as an isolated prompt, PEARL groups each original problem and its checked variants as one training unit with the same expected answer. During cluster-level RL optimization, the policy samples rollouts from all variants in an active cluster, and an answer verifier scores the final answers. The rewards are converted into relative advantages across the whole cluster, so model outputs on different versions of the same problem can be compared directly. However, not all data clusters provide useful training signals, because some clusters are already solved in all versions and some remain unsolved in all versions. Active filtering is therefore used to keep partially solved clusters as the active RL training set, which focuses the RL budget on informative clusters and reduces noisy updates. In this way, PEARL turns semantically equivalent variants into structured training signals for improving mathematical reasoning.

The main contributions of this paper are as follows:

- We introduce the idea of cluster-level training for RLVR with semantic perturbations, where an original problem and its answer-preserving variants are treated as a unified optimization unit rather than independent samples.
- We propose PEARL, which performs cluster-level RL optimization over these clusters. PEARL compares model outputs across semantically equivalent variants and uses the group structure to provide clearer training signals for mathematical reasoning.
- We conduct experiments on six mathematical reasoning benchmarks and multiple robustness benchmarks. PEARL achieves strong results against competitive RLVR baselines while using a small active RL training set.

2 Related Work

2.1 Reinforcement Learning with Verifiable Rewards

Reinforcement learning with verifiable rewards (RLVR) has become a central paradigm for improving mathematical reasoning in large language models. DeepSeekMath (Shao et al., 2024) introduces GRPO, which estimates advantages by comparing multiple sampled responses within the same prompt group and removes the need for a separate value model. Building on this line, subsequent studies improve the stability and efficiency of GRPO-style optimization. DAPO (Yu et al., 2026) investigates decoupled clipping, dynamic sampling, and reward shaping for large-scale long-CoT training. Dr. GRPO (Liu et al., 2025) analyzes optimization bias in GRPO and proposes an unbiased variant. Other analyses examine what RLVR signals and sampling distributions actually reinforce: Shao et al. (2025) study learning under spurious rewards, while Chen et al. (2026b) investigate whether RLVR elicits reasoning beyond capabilities already present in the base model. A complementary direction reuses each problem in multiple reformulated forms during training. MathForge introduces Multi-Aspect Question Reformulation (MQR), which generates answer-preserving variants and combines them with difficulty-aware GRPO (Dai et al., 2026).

These methods all treat each prompt, whether original or reformulated, as an independent training instance and focus on algorithmic refinements such as advantage estimation, sample weighting, or reward shaping. PEARL groups each problem with its equivalent variants into a perturbation cluster and uses the cluster as the basic comparison unit, turning cross-variant consistency into an additional supervision signal that is compatible with these algorithmic improvements.

2.2 Perturbation Learning for Reasoning

A growing body of work studies how mathematical reasoning changes when a problem is rewritten without changing its core meaning. GSM-Plus (Li et al., 2024) modifies grade-school problems with changed targets or added statements. GSM-Symbolic shows that models can give different results for different versions of the same symbolic template (Mirzadeh et al., 2025). MATH-Perturb extends this study to competition-level problems with both simple and hard changes (Huang et al., 2025). Related controlled studies show that distracting context can change the model's reasoning path even when the needed computation is unchanged (Yang et al., 2025b).

These studies suggest that equivalent rewrites can show whether a model has learned the underlying mathematical relation, rather than only a surface form of the problem. A related data-construction direction synthesizes problems around weaknesses detected during reinforcement learning, as in SwS (Liang et al., 2026). However, most perturbation work uses rewrites mainly for evaluation or treats each rewritten problem as a separate example. PEARL instead uses validated answer-preserving variants for training. It groups equivalent versions of the same problem and trains the model to reason correctly across the group.

3 Methodology

3.1 Preliminaries

We consider reinforcement learning with verifiable rewards for mathematical reasoning. The initial training set is denoted as $\mathcal{D}_0 = \{(x_{i,0}, a_i^*)\}_{i=1}^{M}$, where $x_{i,0}$ is the original problem and a_i^* is its gold answer. A semantic perturbation of $x_{i,0}$ is a reformulated problem that changes the surface realization of the question while preserving the underlying mathematical task. We write the k-th perturbation as $x_{i,k}$, where $k = 1, \ldots, K$.

PEARL groups each original problem and its variants into a semantic perturbation cluster $\mathcal{C}_i = \{x_{i,k}\}_{k=0}^{K}$. The key learning challenge is that a policy π_θ may produce different outcomes on problems of the same cluster, even though all problems share the same answer a_i^*. PEARL therefore treats $\mathcal{C}_i$ as the basic unit for filtering and optimization, rather than treating $x_{i,0}, x_{i,1}, \ldots, x_{i,K}$ as unrelated augmented samples.

3.2 PEARL Framework

We now introduce PEARL, a reinforcement learning framework that uses semantically equivalent variants as structured training signals. PEARL consists of four modules. In the **preprocessing** module, PEARL removes original problems that are not suitable for perturbation training. In the **data augmentation** module, each retained problem is rewritten into variants that preserve the answer. PEARL then groups these variants with the original problem into a semantic perturbation cluster. In the **data filtering** module, PEARL tests each cluster with the current policy. It selects the clusters worth training. Finally, in the **cluster level RL optimization** module, PEARL optimizes all variants and rollouts in an active cluster as one comparison unit. The objective combines consistency rewards for correct answers, relative advantages within the cluster, perturbation alignment, and KL regularization.

3.2.1 Preprocessing

The preprocessing module operates on the original problem set $\mathcal{D}_0$ before any perturbation is generated. Easy problems are often solved by the base model in most forms, so their variants provide little training signal. To build a cleaner candidate pool for later cluster construction, we drop the easy part. PEARL measures this range with question perplexity. The score is computed by the base model. It is computed on the raw problem text $x_{i,0}$, without using the gold answer. Given the tokenized question $u_{i,1:T_i}$, we define

$$p_i^q = \exp\left(-\frac{1}{T_i - 1}\sum_{t=2}^{T_i} \log \pi_{\text{base}}(u_{i,t} \mid u_{i,<t})\right). \tag{1}$$

After calculating the perplexity of the questions, we experimentally identified a perplexity range better suited for training, which serves as the final selection result. Appendix C analyzes the details of this preprocessing module.

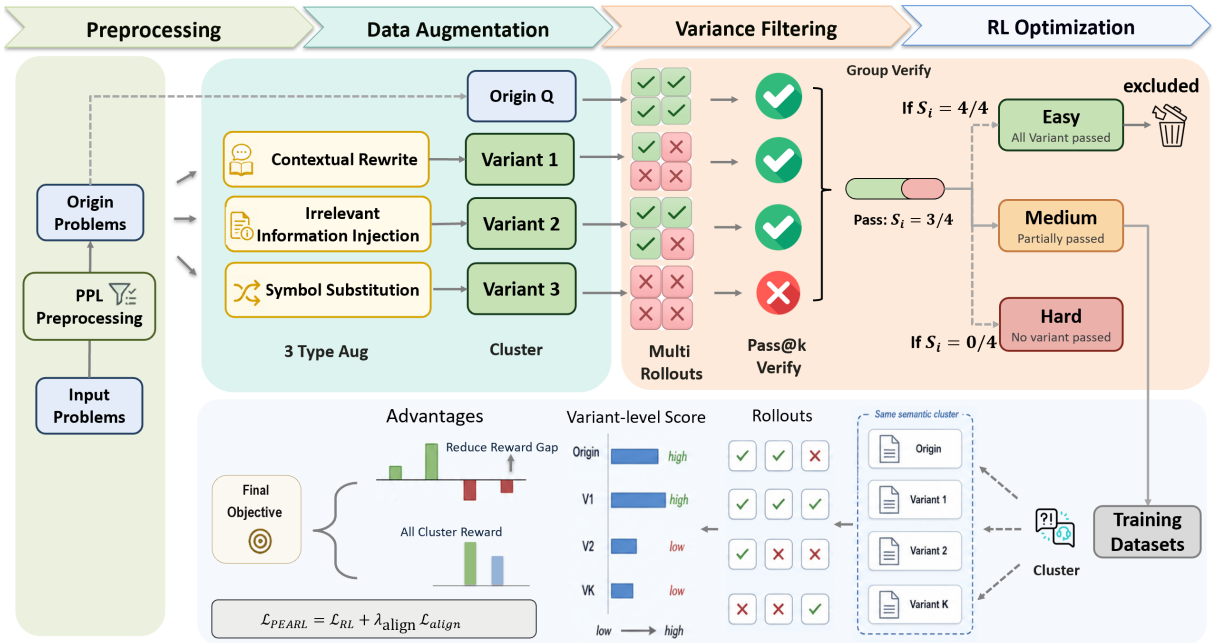

Figure 2: Overview of PEARL. PEARL constructs semantic perturbation clusters from preprocessed math problems, verifies each variant with multiple rollouts, and selects partially solved clusters as the active RL dataset. During optimization, all variants in the same cluster are treated as one training unit, allowing PEARL to derive cluster-level rewards, relative advantages, and perturbation alignment signals from semantically equivalent formulations.

3.2.2 Data Augmentation

For each preprocessed problem $(x_{i,0}, a_i^*)$, PEARL generates K semantically equivalent variants and forms the cluster $\mathcal{C}_i$. In this work, we set three type of variants. The three variants are generated by scenario injection (Variant 1), distractor injection (Variant 2), and variable renaming with paraphrasing (Variant 3). Variant 1 places the problem in a concrete setting while preserving the same equations, constraints, and target quantity. Variant 2 adds irrelevant information that must not introduce a new constraint or change the requested computation. Variant 3 changes symbols, entity names, or wording while preserving the underlying mathematical relation. A concrete example of these perturbation types is shown in Appendix B.

Variants are validated before training to ensure that the perturbations preserve the original mathematical semantics. We use an automatic LLM-based validator to filter out candidates that change the core mathematical computation, introduce ambiguity, or fail to preserve answer equivalence. Appendix B provides the generation procedure, prompt templates, and validation criteria.

3.2.3 Variance Filtering

After augmentation, PEARL filters problem clusters according to their current learnability. For each variant $x_{i,k}$, the current policy performs N screening rollouts. The variant is regarded as passed if at least one rollout produces the correct final answer a_i^*. Let s_i denote the number of passed variants in the cluster.

After multi-rollout verification, PEARL counts how many variants are solved in each semantic cluster. Let s_i denote the number of passed variants in cluster i. Each cluster contains $K + 1$ variants, including the original problem. As shown in Figure 2, PEARL uses s_i to assign each cluster to one of three buckets. When $s_i = K + 1$, all variants are solved. The cluster is labeled as `easy` and excluded from training. When $s_i = 0$, no variant is solved. The cluster is labeled as `hard`. The remaining cases consist of instances where the model failed to achieve either a perfect score or a complete failure on the original or perturbed problems, which are all labeled as `medium`.

Only `medium` clusters are used as the active RL pool. They contain both solved and unsolved variants under the same mathematical semantics. This makes them useful for comparing model behavior across equivalent formulations. In contrast, `easy` clusters provide little new training signal, and `hard` clusters are not yet suitable for direct optimization. We further explored the effects of hard sample injection in the Appendix F.

3.2.4 Cluster-Level RL Optimization

As shown in Figure 2, each training cluster contains one original problem and several variants that share the same gold answer. PEARL treats the whole cluster as one training unit.

The current policy samples G rollouts per prompt, and we collect them into the cluster rollout set $\mathcal{B}_i$, indexed by the cluster identifier i. The cluster optimization objective has three parts. Cluster reward couples rollout correctness with answer consistency inside the cluster. A main RL loss applies a standard GRPO update over each cluster and skips clusters that the policy has already mastered. Perturbation alignment loss adds an extra update to variants whose mean reward is below the original problem.

Cluster reward. A standard binary correctness reward is sufficient for a single prompt but cannot distinguish two correct rollouts that disagree on the final answer across reformulations of the same problem. To impose this constraint at the cluster level, we couple correctness with the fraction of cluster rollouts that share the same normalized answer. For any rollout y in a cluster, let $e(y) \in \{0, 1\}$ be the verifier label and let $c(y) \in (0, 1]$ be the frequency of its normalized answer over all rollouts in the same cluster. The reward decomposes into a base correctness term and a consistency bonus,

$$r(y) = r_{\text{base}}(y) + r_{\text{cons}}(y), \tag{2}$$

$$r_{\text{base}}(y) = e(y), \qquad r_{\text{cons}}(y) = e(y)\, c(y). \tag{3}$$

Both terms vanish when the rollout is incorrect, so the consistency bonus is active only when the answer is correct and wrong but recurring answers are not reinforced.

Optimization objectives. PEARL standardizes $r(y)$ over $\mathcal{B}_i$ into a cluster-wide advantage and applies the standard GRPO clipped surrogate with length-normalized old-policy log-ratio, yielding a cluster objective J_i (Shao et al., 2024). The shared baseline transfers competence from successful variants to failing rollouts on other variants of the same task. The main RL loss aggregates these objectives over all active clusters $\mathcal{I}$,

$$\mathcal{L}_{\text{RL}} = -\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} J_i, \tag{4}$$

incorporating an implicit old-policy KL term with weight β_{KL}. However, because the cluster-wide advantage narrows variant gaps only on average, variants that clearly lag behind the original problem require extra optimization pressure. Let $\bar{r}_i^0$ and $\bar{r}_i^k$ be the mean rewards of the original problem and of variant k in cluster $\mathcal{C}_i$. We define the reward gap $d_i^k = \max(0, \bar{r}_i^0 - \bar{r}_i^k)$, which is positive only when variant k is weaker than the original. Using d_i^k as a penalty weight, we introduce a perturbation alignment loss defined as a vanilla policy-gradient term strictly on the variant rollouts:

$$\mathcal{L}_{\text{align}} = -\mathbb{E}_i \Bigg[\sum_{k \geq 1} d_i^k \cdot \mathbb{E}_{y \in \mathcal{B}_i^k} \big[A(y) \, \log \pi_\theta(y \mid x_{i,k}) \big] \Bigg], \tag{5}$$

where $\mathcal{B}_i^k$ is the rollout set for variant k in cluster $\mathcal{C}_i$. Here, $A_i^k(y)$ is the variant-standardized advantage: it is computed in the same way as $A_i^{\text{cluster}}(y)$, but the mean and standard deviation of $r(y)$ are taken over $\mathcal{B}_i^k$ instead of $\mathcal{B}_i$. This removes the cluster-level offset already captured by $\mathcal{L}_{\text{RL}}$, ensuring the two losses act on independent parts of the update.

Final objective. The final PEARL objective combines the main RL loss with the perturbation alignment loss: $\mathcal{L}_{\text{PEARL}} = \mathcal{L}_{\text{RL}} + \lambda_{\text{align}} \mathcal{L}_{\text{align}}$. Here, λ_{align} balances the perturbation alignment term against the cluster-level RL objective.

4 Experiment

PEARL is evaluated on diverse mathematical reasoning benchmarks spanning standard competition sets and robustness sets. The experiments suggest that organizing semantically equivalent variants into clusters leads to favorable empirical trends compared with strong RLVR baselines, and that the resulting models tend to behave more consistently under meaning preserving reformulations.

4.1 Experimental Setup

Models and Datasets We train Qwen2.5-Math-7B (Yang et al., 2024) mainly on the MATH Dataset (Hendrycks et al., 2021). To examine whether the method transfers beyond the main backbone, additional runs are conducted on Qwen2.5-Math-1.5B (Yang et al., 2024), DeepSeek-Math-7B (Shao et al., 2024), and Qwen3-4B (Yang et al., 2025a).

Benchmarks The evaluation uses six main benchmarks: AIME2025, AIME2024, AMC23,

Methods	AIME25	AIME24	AMC23	MATH500	Minerva	Olympiad	Avg. (Δ)
Qwen2.5-Math-7B	4.79	12.19	35.23	48.60	15.07	16.33	22.04
GRPO	8.44	20.94	58.98	72.20	27.76	37.33	37.61 (–)
Dr.GRPO	8.23	21.04	58.59	72.05	28.58	35.89	37.40 (−0.21)
GPG	9.06	21.98	59.61	72.05	27.21	37.67	37.93 (+0.32)
DAPO	8.75	21.25	58.20	72.70	29.50	37.22	37.94 (+0.33)
GSPO	8.33	19.38	60.16	73.00	28.12	37.26	37.71 (+0.10)
GRPO-AD	9.48	21.56	59.06	73.25	29.14	37.07	38.26 (+0.65)
DGPO	10.21	23.85	61.02	74.25	31.07	38.33	39.79 (+2.18)
MQR	11.77	**25.00**	59.38	77.85	31.43	40.81	41.04 (+3.43)
PEARL	**14.16**	24.16	**67.50**	**79.60**	**36.03**	**42.58**	**44.01 (+6.40)**

Table 1: Comparative results of methods trained on the MATH dataset using Qwen2.5-Math-7B.

MATH500 (Hendrycks et al., 2021), Minerva(Lewkowycz et al., 2022), and OlympiadBench(He et al., 2024). To ensure stable results, we perform 32 runs for AIME2025, AIME2024, and AMC23, and 4 runs for other benchmarks, reporting the average performance across the respective runs. These datasets cover both short competition problems and longer olympiad-style reasoning. For robustness evaluation, each benchmark is evaluated with 4 runs. Robustness evaluation uses two groups of test sets. The first group contains perturbed versions of the main benchmarks. The second group contains external robustness benchmarks, including GSM-Plus(Li et al., 2024), GSM-Symbolic(Mirzadeh et al., 2025), MATH-Perturb and Problematic. These benchmarks test wording changes, symbolic changes, distractors, and broader distribution shifts.

Baseline PEARL is compared with the base model without RL post-training and several RLVR baselines, including GRPO (Shao et al., 2024), Dr. GRPO (Liu et al., 2025), GPG (Chu et al., 2025), DAPO (Yu et al., 2026), GSPO (Zheng et al., 2025), GRPO-AD (Zhang and Zuo, 2025), DGPO, and MQR (Dai et al., 2026). For a controlled comparison, all reported baselines use the same backbone, training split, verifier, rollout budget, decoding settings, and evaluation scripts.

Implementation Details We train PEARL with full-parameter fine-tuning on 8 NVIDIA A100 GPUs under the same RLVR setting used by the baselines. More implementation details are given in Appendix A.

4.2 Main Results

PEARL outperforms across methods. Table 1 reports the main results. PEARL elevates the average score to **44.01%**, yielding a substantial gain of **+6.40%** over GRPO and the best result among all compared RLVR methods. It also achieves the highest score on five of the six benchmarks and the best average score, with particularly large improvements on AMC23, MATH500 and Minerva. This suggests that the gain is consistent rather than driven by a single benchmark. Methods that mainly adjust optimization details, such as Dr. GRPO, GPG, DAPO, GSPO, and GRPO-AD, stay close to GRPO, while approaches that introduce stronger data or optimization strategies, such as DGPO and MQR, move the frontier further. PEARL extends this trend by a clear margin over the strongest baseline MQR, suggesting that perturbation-aware cluster optimization offers a complementary direction to existing RLVR improvements. By selecting partially solved clusters and aligning rollouts across equivalent variants, PEARL converts perturbation structure into a more informative training signal.

PEARL transfers across model scales. Table 2 evaluates whether PEARL is tied to a single backbone by applying the same training data, prompt template, verifier, and evaluation protocol to DeepSeek-Math-7B, Qwen2.5-Math-7B, Qwen2.5-Math-1.5B, and Qwen3-4B. These models cover different families and parameter scales. Across the six reasoning benchmarks, PEARL improves the average score for all evaluated backbones. This suggests that the benefit of perturbation-cluster training comes from the training structure itself, rather than from a backbone-specific effect.

4.3 PEARL Enhances Robustness to Semantic Perturbations

PEARL is compared with GRPO, DAPO, and GSPO on two groups of benchmarks focused on perturbations. The first group contains perturbation variants of the standard math benchmarks

Model	Method	AIME25	AIME24	AMC23	MATH500	Minerva	Olympiad	Avg.
DeepSeek-Math-7B	Base	0.21	2.19	21.02	43.65	18.29	14.54	16.65
	PEARL	0.94	6.56	35.15	48.80	20.58	18.39	21.74
Qwen2.5-Math-7B	Base	4.79	12.19	35.23	48.60	15.07	16.33	22.04
	PEARL	14.16	24.16	67.50	79.60	36.03	42.58	44.01
Qwen2.5-Math-1.5B	Base	5.31	11.88	30.94	34.95	8.55	21.93	18.93
	PEARL	11.56	16.56	52.42	72.15	32.35	33.68	36.45
Qwen3-4B	Base	16.56	23.44	41.25	55.80	22.79	23.88	30.62
	PEARL	23.44	26.56	52.50	73.40	40.72	32.56	41.53

Table 2: Comparative results of PEARL across different base models.

Method	AIME25*	AIME24*	AMC23*	MATH500*	Minerva*	Olympiad*
Qwen2.5-Math-7B	3.33	8.89	34.16	40.86	9.39	13.24
GRPO	6.94	12.50	45.62	72.20	15.18	34.78
DAPO	11.39	11.11	49.38	75.60	15.44	35.47
GSPO	8.61	13.06	57.71	75.95	16.91	37.39
PEARL	**13.61**	**16.39**	**67.29**	**80.52**	**24.63**	**41.51**

Table 3: Robustness comparison on held-out pipeline perturbations. Columns marked with * are augmented using the PEARL data augmentation framework to enlarge the benchmark.

held out from training. These variants are generated from evaluation problems using the perturbation protocol in Section 3.2 and the same equivalence validation procedure used for training clusters. The second group contains external robustness benchmarks, including GSM-Plus, GSM-Symbolic, MATH-Perturb and Problematic. Together, the two groups evaluate both pipeline-style reformulations and independently constructed perturbation benchmarks. Appendix D gives the detailed robustness protocol.

Table 3 shows that PEARL improves over GRPO, DAPO, and GSPO on the perturbation benchmarks generated by the same pipeline and held out from training. Table 4 further shows that PEARL obtains strong results on external perturbation benchmarks, indicating that the benefit extends beyond the exact perturbation templates used during training. Together, these results demonstrate that training with validated semantic perturbation clusters improves final-answer accuracy under diverse reformulations. Appendix E provides representative examples.

4.4 Ablation Study

Table 5 ablates the main components of PEARL. *w/o Variance Filtering* trains on all generated perturbation clusters, *w/o Cluster-Level RL Optimization* treats variants as independent prompts, *w/o Cluster Reward* keeps only the base correctness reward, and *w/o Perturbation Alignment Loss* removes the auxiliary alignment loss. PEARL also uses three types of checked variants to construct each semantic perturbation cluster: scenario injection, distractor injection, and variable renaming with paraphrasing.

The ablation results show that variance filtering is mainly an efficiency-oriented component rather than a mechanism that always maximizes every individual benchmark score. Training without variance filtering can be competitive and even slightly better on MATH500 and Minerva, since it keeps a broader set of perturbation clusters. However, these gains are limited, while the model must train on substantially more data. Under the fixed RL budget considered in this paper, the filtered PEARL setting achieves the best results on most benchmarks and keeps the active training pool much smaller, making it a more efficient default choice.

The degradation from *w/o Cluster-Level RL Optimization* suggests that PEARL's gains are not explained by data augmentation alone. Optimizing semantically equivalent variants as a unified training unit is more effective than treating them as independent prompts. The results for *w/o Cluster Reward* and *w/o Perturbation Alignment Loss* further indicate that the two auxiliary objectives provide complementary benefits. The consistency term in the cluster reward encourages stable answers across equivalent variants, while the perturbation

Method	GSM-Plus	GSM-Symbolic	MATH-Perturb	Problematic
Qwen2.5-Math-7B	52.84	58.91	31.42	54.77
GRPO	69.43	74.21	34.76	61.65
DAPO	69.27	76.71	35.66	62.78
GSPO	74.31	77.40	36.20	63.91
PEARL	**76.18**	**80.28**	**40.51**	**65.03**

Table 4: Robustness comparison on external perturbation benchmarks.

Variant	AIME25	AIME24	AMC23	MATH500	Minerva	Olympiad
w/o Variance Filtering	13.44	21.04	62.42	**80.05**	**36.49**	42.54
w/o Cluster-Level RL Optimization	10.21	20.83	62.27	76.25	31.80	40.36
w/o Cluster Reward	11.35	22.50	64.69	77.10	33.09	41.77
w/o Perturbation Alignment Loss	11.56	22.81	65.23	78.60	33.64	42.47
w/o Scenario Data	10.31	18.23	65.31	78.80	34.56	38.05
w/o Distractor Data	11.67	18.33	66.87	79.25	34.93	37.65
w/o Renaming Data	13.28	19.58	67.34	79.40	35.11	38.13
PEARL	**14.16**	**24.16**	**67.50**	79.60	36.03	**42.58**

Table 5: Ablation study of PEARL components.

alignment loss reduces performance gaps between original and perturbed formulations. The data-type ablations show that all three perturbation sources contribute to the final model, and the full PEARL setting provides the strongest overall balance across benchmarks. Among these data-type ablations, *w/o Scenario Data* performs clearly worse than *w/o Renaming Data*. This gap suggests that scenario perturbations provide particularly useful learning signals, likely because they induce broader changes in problem presentation while preserving the underlying mathematical semantics.

4.5 Training Data Usage and Efficiency

We further examine RL-stage data usage to clarify whether PEARL's gains come from simply using more augmented examples. PEARL starts from the same raw training source, constructs answer-preserving perturbation clusters, and then keeps only partially solved clusters for policy optimization. Thus, perturbation is used to expose informative cross-variant signals rather than to indiscriminately enlarge the RL training set.

As shown in Figure 3, PEARL uses about 3K active origin training records, which is substantially fewer than the data used by the compared GRPO-style methods. Combined with the ablation in Table 5, this suggests that PEARL's gains come from structured cluster selection and optimization rather than naive data expansion.

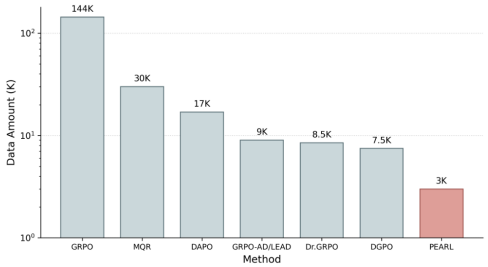

Figure 3: RL-Stage Training Data Usage Statistics.

5 Conclusion

In this paper, we propose PEARL, an RLVR framework using semantic perturbation clusters as structured training units for math reasoning. PEARL groups each problem with its validated answer-preserving variants, applies variance filtering to retain partially solved clusters as the active RL training set, and performs cluster-level RL optimization over the selected clusters. Its objective combines a cluster reward that couples correctness with answer consistency and a perturbation alignment loss that targets weaker variants within each cluster. Experiments on six mathematical benchmarks and robustness benchmarks show that PEARL achieves strong results against competitive RLVR baselines while using a small RL training set. The robustness evaluation further indicates that PEARL improves accuracy under semantically equivalent reformulations. These results suggest that semantically equivalent variants can serve as effective training units for improving mathematical reasoning in LLMs.

Limitation

PEARL intentionally applies data filtering before RL training to keep the active pool close to the model's learning frontier. While this makes the training signal more concentrated and helps use the RL budget efficiently, it also means that not every available perturbation cluster is used during optimization. The scope of perturbation-based augmentation can also be enriched further. This work focuses on answer-preserving semantic variants and organizes them as cluster-level training units, while other augmentation perspectives, such as different presentation formats, abstraction levels, symbolic forms, or harmless contextual changes, may provide complementary views of the same underlying reasoning process without changing the main PEARL framework.

Ethics Statement

This work studies reinforcement learning for mathematical reasoning using publicly available and open-source math datasets and model checkpoints. The data consist of mathematical problem statements and reference answers, and do not involve private, personally identifiable, or sensitive user information. Our perturbation construction is answer-preserving and is designed to rewrite mathematical problem statements without changing their underlying semantics. All training and evaluation are conducted on public research benchmarks under their corresponding licenses and usage terms.

References

Huayu Chen, Kaiwen Zheng, Qinsheng Zhang, Ganqu Cui, Yin Cui, Haotian Ye, Tsung-Yi Lin, Ming-Yu Liu, Jun Zhu, and Haoxiang Wang. 2026a. Nft: Bridging supervised learning and reinforcement learning in math reasoning. In *The Fourteenth International Conference on Learning Representations*.

Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2026b. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Advances in Neural Information Processing Systems*, 38:57654–57689.

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. 2025. Gpg: A simple and strong reinforcement learning baseline for model reasoning. *arXiv preprint arXiv:2504.02546*.

Yanqi Dai, Yuxiang Ji, Xiao Zhang, Yong Wang, Xiangxiang Chu, and Zhiwu Lu. 2026. Harder is better: Boosting mathematical reasoning via difficulty-aware GRPO and multi-aspect question reformulation. In *The Fourteenth International Conference on Learning Representations*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. 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)*, pages 3828–3850.

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

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, and 1 others. 2025. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations. *arXiv preprint arXiv:2502.06453*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984.

Xiao Liang, Zhong-Zhi Li, Yeyun Gong, Yang Wang, Hengyuan Zhang, Ying Nian Wu, Weizhu Chen, and 1 others. 2026. Sws: Self-aware weakness-driven problem synthesis in reinforcement learning for llm reasoning. *Advances in Neural Information Processing Systems*, 38:56801–56839.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, volume 2025, pages 94743–94765.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan

Lambert, Sewon Min, Ranjay Krishna, and 1 others. 2025. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, and 1 others. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Minglai Yang, Ethan Huang, Liang Zhang, Mihai Surdeanu, William Yang Wang, and Liangming Pan. 2025b. How is llm reasoning distracted by irrelevant context? an analysis using a controlled benchmark. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13340–13358.

Zhaohui Yang, Yuxiao Ye, Shilei Jiang, Chen Hu, Linjing Li, Shihong Deng, and Daxin Jiang. 2025c. Unearthing gems from stones: Policy optimization with negative sample augmentation for llm reasoning. *arXiv preprint arXiv:2505.14403*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2026. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244.

Jixiao Zhang and Chunsheng Zuo. 2025. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5642–5665.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, and 1 others. 2025. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

A Implementation Details

A.1 Training Setup

PEARL is trained on `Qwen2.5-Math-7B` using 8 NVIDIA A100 GPUs. The policy is updated with full-parameter fine-tuning. Under the controlled comparison protocol, all baseline methods use the same backbone, rollout budget, verifier, and training length. Table 6 lists the training hyperparameters.

The implementation follows the common GRPO-style mathematical RL setup used in prior work, including group sampling, clipped policy ratios, old-policy KL regularization, temperature-based rollout sampling, cosine learning-rate decay, and full-parameter optimization (Shao et al., 2024; Yu et al., 2026). These settings are kept fixed across PEARL and the reproduced baselines to make the comparison controlled. We use a clipping range of $\epsilon_c = 0.2$.

The training prompt shown in table4 is designed to make answer extraction stable. The system instruction asks the model to give concise reasoning and then place the final answer inside an `<answer>` tag.

A.2 Evaluation Setup

Unless otherwise stated, evaluation follows the reporting protocol in Section 4: AIME2025, AIME2024, and AMC23 are averaged over 32 runs, while the other main benchmarks are averaged over 4 runs. Robustness benchmarks are evaluated over 4 runs.

All methods in the same comparison use the same decoding settings. We use stochastic sampling with temperature $T = 0.7$ and top-$p = 0.95$, with a maximum output length of 2048 tokens for all benchmarks. The reported scores are mean accuracies averaged over the repeated runs described above. `math-verify` checks equivalence between generated answers and reference solutions. The same answer normalization and verifier thresholds are used for all methods, and these normalized verifier outputs are also used when computing the consistency reward r_{cons} and cluster-level advantages. The evaluation prompt matches the training template, so the model is not tested under a different answer format.

B Perturbation Construction Details

B.1 Generation Procedure

In the data augmentation module, each retained original problem $x_{i,0}$ is rewritten into $K = 3$ answer-preserving variants and grouped with the original problem into a semantic perturbation cluster $\mathcal{C}_i$. The three variants are generated by scenario injection, distractor injection, and variable renaming with paraphrasing, matching the terminology used in Section 3. We use `Qwen3-32B` (Yang et al., 2025a) to generate candidate variants with separate prompts for the three perturbation types. The prompts are written to change only the surface form of the problem, and explicitly restrict the generator from changing the mathematical structure, target quantity, or final answer.

To maintain equivalence, the target quantity, numerical values, and constraints must remain identical to the original problem. The generator is also asked not to introduce new solvable conditions or hidden assumptions. This is especially important for scenario injection and distractor injection, where added context can easily change the problem if it becomes part of the solution. The generator outputs only the rewritten problem text, without any solution or explanation.

Construction implementation and decoding settings. Candidate generation is implemented with an OpenAI-compatible vLLM server hosting `Qwen3-32B`. The server exposes the checkpoint under the served model name `Qwen3` and uses the `qwen3` reasoning parser. For each original problem, we request one candidate for each of the three perturbation prompts, subject to the retry policy below. Each request receives the original problem statement only. The gold answer is never passed to the generator and is reserved entirely for the subsequent equivalence validation stage. We audited the construction pipeline by scanning all rendered generator requests and confirmed that no gold-answer field was ever materialized in the prompt sent to `Qwen3-32B`. We further scanned all generated variants and confirmed that none reproduces the gold-answer string verbatim, ruling out trivial answer-aligned rewrites. Table 7 reports the generation settings used in data construction.

Generation-side output screening. We first generate candidates with Qwen3 thinking enabled and retain only the final response content as the rewritten problem. A response is retried if it is

Category	Value
Base Model	Qwen2.5-Math-7B
GPUs	8 × NVIDIA A100
Training framework	DDP full-parameter fine-tuning
Effective batch size	16 clusters (1/GPU, accum. 2)
Optimizer	8-bit AdamW
Learning rate	5×10^{-6}
RL sampling	4 rollouts, $T = 0.7$, top-$p = 0.95$
Sequence limits	2048
Precision	bfloat16
KL anchor	old policy log probabilities
Clipping range	$\epsilon_c = 0.2$
PEARL loss	$\lambda_{\text{align}} = 0.5$

Table 6: Training hyperparameters used for PEARL on the Qwen2.5-Math-7B backbone unless otherwise stated.

Configuration	**Value**
Generator	`Qwen3-32B`
Serving engine	vLLM OpenAI-compatible server
Served model name	`Qwen3`
Reasoning parser	`qwen3`
Temperature	0.6
Top-p	0.95
Maximum generated tokens	4096
Maximum thinking-on attempts	3
Fallback after failed attempts	One thinking-off generation
Request timeout	300 seconds

Table 7: Generation settings used to construct semantic perturbation candidates.

empty, substantially shorter than the original problem, contains leaked thinking tags, or contains solution-like material such as an explanation or an attempted derivation. We allow at most three thinking-enabled attempts. If all three attempts fail these output checks, we make one fallback call with thinking disabled. Before validation, formatting wrappers such as code fences and generic output prefixes are removed. These checks detect malformed generations and answer leakage; they are distinct from semantic-equivalence validation, which is performed by Q wen as described below.

Table 8 gives a representative semantic perturbation cluster used by PEARL. The variants change the surface realization of the problem through scenario injection, distractor injection, and variable renaming, while preserving the equations, intervals, continuity condition, target expression, and final answer.

B.2 Prompt Templates and Acceptance Criteria

This section describes the prompt and validation pipeline used to construct semantic perturbation clusters. The pipeline has two stages. First, the generator produces one candidate variant for each perturbation type under shared generation constraints. Second, an automatic judge verifies whether the candidate preserves the original mathematical task and answer. Candidates that fail any validation check are rejected before cluster formation.

Candidate generation. Figure 5 shows the three prompts used to create candidate variants. All prompts receive only the original problem and require the output to be a rewritten problem. The gold answer is not part of the generator input and only appears in the downstream validation prompts (Figures 7 and 8). The shared constraints prevent the generator from changing quantities, equations, domains, assumptions, or the requested answer. The type-specific instruction then determines whether the candidate is produced by scenario injection, distractor injection, or variable renaming with paraphrasing.

Validation protocol. A candidate variant is accepted only if it preserves the original problem's target quantity, mathematical conditions, solution set, and final answer. The automatic judge evaluates target consistency, condition consistency, the absence of newly introduced relevant information, answer consistency, and ambiguity. For distractor injection, the added information must be demonstrably unnecessary, meaning that removing it leaves the problem unchanged and that treating it as an additional constraint is not required to derive the answer. We use Qwen2.5-72B for this validation step, with the prompt shown in Figure 7. The judge is instructed to follow a conservative criterion and return `FAIL` whenever semantic equivalence is uncertain.

Validation protocol. A candidate variant is accepted only if it preserves the original problem's target quantity, mathematical conditions, solution set,

Type	Problem	Perturbation Description
Original	Let $$f(x) = \begin{cases} ax+3, & x>2, \\ x-5, & -2 \le x \le 2, \\ 2x-b, & x<-2. \end{cases}$$ Find $a+b$ if the piecewise function is continuous.	This is the original problem. It defines the mathematical structure, target quantity, and gold answer used as the semantic anchor for later perturbations.
Scenario Injection	In a city planning project, the park maintenance cost $f(x)$ is defined according to a standardized visitor-flow index x. If $x>2$, the cost is $ax+3$ thousand dollars; if $-2 \le x \le 2$, the cost is $x-5$ thousand dollars; and if $x<-2$, the cost is $2x-b$ thousand dollars. Determine $a+b$ if the cost function is continuous, meaning that there are no sudden jumps in cost as x changes.	This variant embeds the same piecewise continuity problem into a concrete application scenario. The equations, intervals, continuity condition, and target quantity remain unchanged.
Distractor Injection	In an experimental data record, the variable x represents a standardized measurement value, and the function is defined as $$f(x) = \begin{cases} ax+3, & x>2, \\ x-5, & -2 \le x \le 2, \\ 2x-b, & x<-2. \end{cases}$$ The record also notes that the temperature at noon was $20^\circ\mathrm{C}$ and the wind speed was $10\,\mathrm{km/h}$. If the piecewise function is continuous, find $a+b$.	This variant adds distracting background facts. The added temperature and wind-speed information is irrelevant to the continuity constraints and should not affect the solution.
Variable Renaming and Paraphrasing	Let $$g(z) = \begin{cases} cz+3, & z>2, \\ z-5, & -2 \le z \le 2, \\ 2z-d, & z<-2. \end{cases}$$ Determine $c+d$ if the function $g(z)$ is continuous.	This variant renames the function and variables from f, x, a, b to g, z, c, d. The algebraic structure, intervals, continuity condition, and requested expression are preserved.

Table 8: Example of answer-preserving semantic perturbations used in PEARL.

```
<|im_start|>system
You are a helpful assistant that solves math problems.
You should provide a concise reasoning process, then give the final answer.
Please format the final answer as <answer>FINAL</answer>.
Reasoning can be free form before the final answer tag.
If the answer is an expression or fraction, output it cleanly inside <answer>.
<|im_end|>
<|im_start|>user
Problem:
{Problem Text}

Please reason step by step, and end with a single line in the form <answer>FINAL</answer>.
<|im_end|>
<|im_start|>assistant
```

Figure 4: Training prompt template used in PEARL.

and final answer. The automatic judge evaluates target consistency, condition consistency, the absence of newly introduced relevant information, answer consistency, and ambiguity. For distractor injection, the added information must be demonstrably unnecessary, meaning that removing it leaves the problem unchanged and that treating it as an additional constraint is not required to derive the answer. We use Qwen2.5-72B for this validation step, with the prompt shown in Figure 7. The judge is instructed to follow a conservative criterion and return `FAIL` whenever semantic equivalence is uncertain.

Answer consistency check. In addition to the structural judge, a separate answer consistency check is used for candidates that pass the target and condition checks but may still contain subtle semantic drift. The prompt is shown in Figure 8. This check asks the judge to decide whether the original gold answer remains valid for the candidate without producing a full solution.

```
[Scenario injection prompt]
You are an expert in mathematical data augmentation. Rewrite a competition math problem by
embedding the same mathematical structure into a concrete setting.

Original problem: {Problem Text}

Preserve exactly the same mathematical meaning, target quantity, constraints, solution set,
and final answer. Keep every numerical value, equation, inequality, domain restriction, and
answer-determining condition unchanged. Do not add any fact that changes the computation,
introduces a new constraint, or creates an alternative interpretation.

Output only the rewritten problem text. Do not include a solution, proof, explanation,
answer, markdown wrapper, or commentary.
```

```
[Distractor injection prompt]
You are an expert in mathematical data augmentation. Rewrite a competition math problem by
adding one or two harmless statements that may look relevant but are not needed to solve the
problem.

Original problem: {Problem Text}

Preserve exactly the same mathematical meaning, target quantity, constraints, solution set,
and final answer. Keep every numerical value, equation, inequality, domain restriction, and
answer-determining condition unchanged. The added information must be ignorable and must not
constrain the answer, change the computation, introduce a hidden assumption, or create an
alternative interpretation.

Output only the rewritten problem text. Do not include a solution, proof, explanation,
answer, markdown wrapper, or commentary.
```

```
[Variable renaming and paraphrasing prompt]
You are an expert in mathematical data augmentation. Rewrite a competition math problem by
renaming variables, symbols, functions, or labels where appropriate and paraphrasing the
wording.

Original problem: {Problem Text}

Preserve exactly the same mathematical meaning, target quantity, constraints, solution set,
and final answer. Keep every numerical value, equation, inequality, domain restriction,
and answer-determining condition unchanged. Do not add a new scenario, extra facts, new
constraints, hidden assumptions, or alternative interpretations.

Output only the rewritten problem text. Do not include a solution, proof, explanation,
answer, markdown wrapper, or commentary.
```

Figure 5: Standalone generation prompts used to create candidate variants for the three perturbation types.

C Data Curation and Filtering Analysis

PEARL uses preprocessing before perturbation generation to control the broad difficulty range of the candidate pool. The preprocessing module scores only the original problem statement with the base model and does not use gold answers, teacher models, or correctness checks. We use question perplexity as a lightweight proxy for whether a problem is close to the model's learning frontier. The lower quantile q_{low} controls how many familiar original problems are removed, while the upper quantile is fixed at 1.0. The question-PPL window bounds are smoothed with an exponential moving average coefficient of 0.8.

After semantic perturbation clusters are constructed, PEARL performs rollout-based filtering to decide which clusters should enter RL training. For each variant in a cluster, the current policy samples N rollouts with vLLM using temperature 0.7, top-p 0.95, and maximum length 2048. A verifier checks the final answers. Clusters solved in all variants are treated as `easy`, clusters solved in no variants are treated as `hard`, and partially solved clusters are treated as `medium`. Only `medium` clusters are used as the active RL training set, because they provide cross-variant learning signals under the same mathematical semantics.

```
[System] You are a strict verifier for mathematical problem perturbations. You compare an
original problem and a rewritten candidate. Your goal is to decide whether the candidate
preserves exactly the same mathematical task and final answer. Be conservative: if equivalence
is uncertain, output FAIL.

User
Original problem: {Original Problem}
Candidate problem: {Candidate Problem}
Gold answer: {Gold Answer}
Perturbation type: {Type}

Check the following items independently:
1. Target consistency: Does the candidate ask for the same quantity, expression, unit, and final
answer format?
2. Condition consistency: Are all original numbers, equations, inequalities, domains, and
constraints preserved?
3. No new relevant information: Does the candidate avoid adding constraints or facts that change
the solution set?
4. Answer consistency: Would a correct solution to the candidate produce an answer equivalent
to the gold answer?
5. Ambiguity: Does the candidate avoid vague wording, changed references, missing assumptions,
or conflicting statements?

Additional check for Distractor Injection: identify each added statement. For each
added statement, decide whether it is needed, helpful, or usable as an extra constraint in
solving the problem. If any added statement changes the computation, narrows the answer, creates
another valid interpretation, or is required to solve the problem, output FAIL. A distractor may
look related, but it must be ignorable.

Output format:
Verdict: PASS or FAIL
Failed checks: list item numbers, or None
Rationale: one concise sentence
```

Figure 6: Automatic judge prompt used for target, condition, answer, and distractor-validity checks.

q_{low}	AIME25	AIME24	AMC23	MATH500	Minerva	Olympiad	Avg.
0.1	13.02	23.33	67.10	80.05	36.76	43.24	43.92
0.2	13.75	24.48	67.81	80.20	34.38	42.32	43.82
0.3	14.16	24.16	67.50	79.60	36.03	42.58	44.01
0.4	13.23	21.77	65.78	80.55	33.63	40.38	42.56

Table 9: Effect of the Stage I question-PPL lower quantile q_{low} on final accuracy.

C.1 Effect of Preprocessing

The preprocessing module scores only the original problem statements with the base model. This step aims to keep problems close to the model's learning frontier. In this analysis, we vary q_{low} from 0.1 to 0.4 and keep the question-PPL upper quantile fixed at 1.0. Smaller values of q_{low} remove fewer familiar problems, while larger values make preprocessing more selective. The resulting models are evaluated on the six main math benchmarks.

Table 9 shows that the question-PPL lower quantile q_{low} has a non-monotonic effect on the average accuracy across the six benchmarks, and the best average score is achieved at $q_{\text{low}} = 0.3$ (44.01%). We therefore adopt $q_{\text{low}} = 0.3$ as the default setting. The accuracy first increases as q_{low} grows from 0.1 to 0.3, and then drops at 0.4. When q_{low} is too small (0.1 and 0.2), preprocessing keeps too many low-PPL problems that the base model already recognizes well. Many of the resulting clusters fall into the `easy` bucket after rollout-based filtering, which dilutes the active pool with low-signal samples and weakens the cluster-level relative comparison. When q_{low} is too large (0.4), preprocessing becomes overly aggressive and discards a noticeable portion of problems that are still close to the model's learning frontier. The remaining candidate pool loses diversity, and the active pool becomes biased toward harder problems, which produces a clear drop on AIME24, AMC23, Minerva, and Olympiad. The setting $q_{\text{low}} = 0.3$ sits between these two failure modes: it removes problems that the base model already masters while preserving enough learnable problems for perturbation clus-

```
[System] You are a strict verifier for mathematical problem perturbations. You compare an
original problem and a rewritten candidate. Your goal is to decide whether the candidate
preserves exactly the same mathematical task and final answer. Be conservative: if equivalence
is uncertain, output FAIL.

User
Original problem: {Original Problem}
Candidate problem: {Candidate Problem}
Gold answer: {Gold Answer}
Perturbation type: {Type}

Check the following items independently:
1. Target consistency: Does the candidate ask for the same quantity, expression, unit, and final
answer format?
2. Condition consistency: Are all original numbers, equations, inequalities, domains, and
constraints preserved?
3. No new relevant information: Does the candidate avoid adding constraints or facts that change
the solution set?
4. Answer consistency: Would a correct solution to the candidate produce an answer equivalent
to the gold answer?
5. Ambiguity: Does the candidate avoid vague wording, changed references, missing assumptions,
or conflicting statements?

Additional check for Distractor Injection: identify each added statement. For each
added statement, decide whether it is needed, helpful, or usable as an extra constraint in
solving the problem. If any added statement changes the computation, narrows the answer, creates
another valid interpretation, or is required to solve the problem, output FAIL. A distractor may
look related, but it must be ignorable.

Output format:
Verdict: PASS or FAIL
Failed checks: list item numbers, or None
Rationale: one concise sentence
```

Figure 7: Automatic judge prompt used for target, condition, answer, and distractor-validity checks.

N	AIME25	AIME24	AMC23	MATH500	Minerva	Olympiad	Avg.
1	12.19	20.31	63.91	79.45	35.47	41.80	42.19
2	14.16	24.16	67.50	79.60	36.03	42.58	44.01
4	14.44	24.27	68.28	77.80	36.94	42.21	43.99

Table 10: Effect of the variance-filtering rollout budget N per variant.

tering, leading to the best balance between data compactness, diversity, and final accuracy.

C.2 Effect of Variance-Filtering

The Variance Filtering module estimates whether each cluster is useful for RL training. PEARL keeps clusters that are partially solvable, since these clusters expose differences among semantically equivalent variants. Additional rollouts refine this learnability estimate by averaging over multiple sampled answers. The preprocessing quantile is therefore fixed at $q_{\text{low}} = 0.3$, while the number of sampled answers N is varied.

Table 10 shows that the variance-filtering rollout budget N also has a non-monotonic effect on the average accuracy, and the best average is obtained at $N = 2$ (44.01%). We therefore adopt $N = 2$ as the default setting. When $N = 1$, only a single rollout is used to decide whether a variant is solvable. This estimate is noisy: a variant that the policy can usually solve may fail on its single rollout, and a variant that the policy occasionally solves by chance may be marked as passed. As a result, the boundary between `easy`, `medium`, and `hard` clusters becomes unreliable, and the active pool contains a non-trivial portion of mislabeled clusters, which directly translates into a clear drop across all benchmarks (Avg. 42.19%). Increasing N from 1 to 2 averages out most of this rollout-level noise and provides a much more stable separation of partially solved clusters, leading to the best average score. Further increasing N to 4 refines the screening signal only marginally, since most easy and hard clusters can already be identified with two rollouts. The benchmark-level differences between $N = 2$ and $N = 4$ are small and inconsistent,

```
[System] You verify answer preservation for rewritten math problems. Do not solve the problem
from scratch. Compare the original problem, the candidate problem, and the gold answer. Determine
whether the gold answer remains the correct final answer for the candidate under the same
interpretation.

User
Original problem: {Original Problem}
Candidate problem: {Candidate Problem}
Gold answer: {Gold Answer}

Questions:
1. Does the candidate ask for exactly the same final quantity as the original?
2. Are all conditions needed to obtain the gold answer preserved?
3.  Does the candidate add, delete, or modify any condition that would make the gold answer
invalid?
4.  If extra context or distractors are present, can they be ignored without changing the
derivation of the gold answer?

Output format:
Answer preserved: YES or NO
Reason: one concise sentence
```

Figure 8: Answer consistency check prompt used after structural validation.

suggesting that the additional rollouts mainly redistribute borderline clusters between the active and the inactive pools rather than producing a meaningful learnability signal. Considering that $N = 4$ doubles the screening cost compared with $N = 2$ while yielding essentially the same average accuracy, we adopt $N = 2$ as the efficiency-oriented default.

C.3 Summary

Based on the above analysis, the main PEARL setting uses $q_{\text{low}} = 0.3$ for preprocessing and $N = 2$ rollouts per variant for rollout-based filtering. Both values are selected by the average accuracy across the six main benchmarks: $q_{\text{low}} = 0.3$ achieves the highest average (44.01%) among the four tested quantiles, and $N = 2$ achieves the highest average among the three tested rollout budgets while costing only half of the screening compute compared with $N = 4$. Smaller values ($q_{\text{low}} \leq 0.2$, $N = 1$) underperform because the active pool either retains too many already-mastered problems or relies on a noisy single-rollout learnability estimate, while larger values ($q_{\text{low}} = 0.4$) overprune the candidate set and reduce the diversity available to perturbation clustering. Under this configuration, the screening input contains 9,582 semantic perturbation clusters, corresponding to 38,328 problem versions. Rollout filtering assigns 5,585 clusters to the `easy` bucket, 1,051 clusters to the `hard` bucket, and keeps 2,946 partially solved clusters as the active RL training set. The active set accounts for 30.75% of the screened clusters and provides the final training pool used by PEARL.

Filtering outcome	Clusters	Problem versions
Stage II screening input	9,582	38,328
`easy` mastered	5,585	22,340
`hard` backlog	1,051	4,204
Active RL training set	2,946	11,784

Table 11: Final rollout-filtering results under the default PEARL data curation configuration. Each semantic perturbation cluster contains one original problem and three generated variants.

D Robustness Evaluation Details

Robustness evaluation uses two groups of benchmarks. The first group contains perturbation variants of AIME2025, AIME2024, AMC23, MATH500, Minerva, and Olympiad that are held out from training. These sets are generated with the same perturbation procedure used for training, but the procedure is applied only to evaluation problems. These examples do not overlap with the training clusters. Therefore, they test whether PEARL transfers to unseen rewrites rather than memorizing training variants.

The second group contains public external benchmarks: GSM-Plus, GSM-Symbolic, MATH-Perturb and Problematic. These datasets are not generated by the PEARL pipeline. They provide an independent check of robustness under symbolic changes, altered wording, distracting information, and distribution shifts.

Both groups are evaluated with the same pro-

tocol used for the main benchmarks. The inference engine, prompt format, decoding settings, answer extraction, and `math-verify` checker are kept fixed across methods. This ensures that differences in robustness results come from training rather than evaluation choices.

E Case Study

We present three representative cases that illustrate where PEARL diverges from the baselines: a geometry problem from Olympiad (Table 12) and a controlled original/perturbed pair from AMC23-22 (Tables 13–15). Each table reports the problem, the retained generated output for each method, the verdict, and a short analysis. In the model-output rows below, only Markdown markup and line wrapping are normalized for LaTeX; the generated reasoning content, including erroneous steps and any verification attempts, is retained for manual inspection and later shortening. Within Python verification blocks, executable statements and displayed outcomes are retained, while non-semantic inline comments are omitted.

F Curriculum Reinjection Analysis

This section studies an optional curriculum reinjection variant for clusters classified as `hard` during variance filtering. These clusters are not used in the default PEARL active pool because none of their variants has a correct rollout during screening. We keep them in a backlog and test whether a subset can become useful later in training. If a hard cluster becomes partially solvable, it can be considered for active training; if all variants become solvable, it is treated as mastered and is not used for further optimization.

We run a conservative data injection variant in which hard backlog clusters are screened by an online LLM judge before entering the active RL batch. The judge is instructed to accept a hard cluster only when the problem is well posed, answerable, aligned with the provided gold answer, and useful as a hard training sample. Ambiguous rewrites, malformed statements, duplicate-looking examples, and answer-mismatched variants are rejected. This setting uses the same Qwen2.5-Math-7B backbone and full-parameter RL training, but it is reported as a diagnostic variant rather than as part of the default PEARL results. The hard probe starts at step 10 and is repeated every 5 optimization steps. At each probe, 8 still-locked hard clusters are sampled and judged using up to two variants from each cluster. The judge uses a minimum confidence threshold of 0.5; accepted clusters are added to an unlocked hard pool, while rejected clusters remain excluded from training.

During optimization, hard examples are injected by replacing a subset of the current active batch rather than by appending extra data. The replacement probability is the hard-sample ratio, which linearly ramps from 0 after step 10 to a maximum of 0.35 over 80 steps. For each batch slot, the trainer samples whether the slot should be replaced; if so, it draws one cluster from the unlocked hard pool and uses that hard cluster in place of the original medium cluster. Table 16 summarizes the injection statistics up to the step-900 checkpoint.

Overall, judged hard injection produces a small positive change in average accuracy (from 44.01 to 44.26, $+0.25$ points), suggesting that the hard backlog is not pure noise and can carry usable training signal once a judge filters out malformed or answer-mismatched variants. The gains are concentrated on benchmarks that reward broader problem coverage and longer reasoning chains, with clear improvements on Olympiad ($+3.41$) and additional gains on Minerva ($+1.03$) and MATH500 ($+1.25$). On the competition-style sets the picture is more nuanced: AMC23 is unchanged while AIME24 and AIME25 move slightly in the opposite direction, so the overall improvement is mainly driven by the broader benchmarks rather than by uniform gains across all sets.

We present this analysis as an exploratory extension to PEARL. The variant already recovers measurable signal from clusters that the default PEARL pipeline excludes, and the gains on the broader benchmarks indicate that mining the hard backlog is a promising direction for further improving PEARL beyond its default medium-cluster recipe.

Entry	Retained generated output	Verdict
Problem	$\triangle ABC$ with $AB = AC$. Points $M, N \in \overline{BC}$ such that AM, AN trisect $\angle BAC$, with M closer to C. Given $\angle AMC = 95^\circ$, compute $\angle ACB$ (gold = 75).	–
PEARL	Triangle ABC is isosceles with $AB = AC$, and points M, N lie on $\overline{BC}$ such that $\overline{AM}, \overline{AN}$ trisect $\angle BAC$, so $\angle BAM = \angle MAN = \angle NAC$. Let $\angle BAM = \angle MAN = \angle NAC = x$; then $\angle BAC = 3x$. We are given $\angle AMC = T^\circ = 95^\circ$. Since $AB = AC$, let $\angle ABC = \angle ACB = y$. The angle sum in $\triangle ABC$ gives $$3x + y + y = 180^\circ \implies 3x + 2y = 180^\circ \implies y = \tfrac{180^\circ - 3x}{2}.$$ Next, in $\triangle AMC$, $$\angle MAC + \angle ACM + \angle AMC = 180^\circ \implies x + y + 95^\circ = 180^\circ \implies x + y = 85^\circ.$$ Substituting $y = \frac{180^\circ - 3x}{2}$ into $x + y = 85^\circ$ gives $$x + \tfrac{180^\circ - 3x}{2} = 85^\circ \implies 2x + 180^\circ - 3x = 170^\circ \implies x = 10^\circ.$$ Then $y = \frac{180^\circ - 3\cdot 10^\circ}{2} = \frac{150^\circ}{2} = 75^\circ$. Therefore $\angle ACB = y = 75^\circ$, and the value of U is $\boxed{75}$.	✓
GRPO	Given that $T = 95$ and triangle ABC is isosceles with $AB = AC$, points M, N lie on $\overline{BC}$ such that $\overline{AM}, \overline{AN}$ trisect $\angle BAC$, with M closer to C, and $\angle AMC = T^\circ = 95^\circ$. We need the measure of $\angle ACB$, denoted U. Let $\angle BAC = 3\theta$; since $\overline{AM}, \overline{AN}$ trisect $\angle BAC$, $\angle BAM = \angle MAN = \angle NAC = \theta$. In $\triangle AMC$, $\angle MAC + \angle ACM + \angle AMC = 180^\circ \implies \theta + \angle ACM + 95^\circ = 180^\circ \implies \angle ACM = 85^\circ - \theta$. Since $AB = AC$, let $\angle ABC = \angle ACB = x$; then $$3\theta + x + x = 180^\circ \implies x = 90^\circ - \tfrac{3\theta}{2}.$$ Since M lies on BC, $\angle MCB = \angle ACB - \angle ACM = x - (85^\circ - \theta) = x - 85^\circ + \theta$. However, since $\angle ACM = 85^\circ - \theta$ is also part of $\angle ACB$, $$\angle ACB = \angle ACM + \angle MCB = 85^\circ - \theta + \theta = 85^\circ.$$ Thus $U = 85^\circ$, so the value of U is $\boxed{85}$.	×
DAPO	Given that $T = 95$ and triangle ABC is isosceles with $AB = AC$, points M, N lie on $\overline{BC}$ such that $\overline{AM}, \overline{AN}$ trisect $\angle BAC$, with M closer to C, and $\angle AMC = T^\circ = 95^\circ$. We need the measure of $\angle ACB$, denoted U. Let $\angle BAC = 3\theta$, so $\angle BAM = \angle MAN = \angle NAC = \theta$. In $\triangle AMC$, $$\theta + \angle ACM + 95^\circ = 180^\circ \implies \angle ACM = 180^\circ - 95^\circ - \theta = 85^\circ - \theta.$$ Since $AB = AC$, let $\angle ABC = \angle ACB = x$; then $$3\theta + x + x = 180^\circ \implies x = 90^\circ - \tfrac{3\theta}{2}.$$ With M on BC, $\angle MCB = \angle ACB - \angle ACM = x - (85^\circ - \theta) = x - 85^\circ + \theta$. However, since $\angle ACM = 85^\circ - \theta$ is also part of $\angle ACB$, $$\angle ACB = \angle ACM + \angle MCB = 85^\circ - \theta + \theta = 85^\circ.$$ Thus $x = 85^\circ$, therefore $U = 85^\circ$ and the final answer is $\boxed{85}$.	×
GSPO	Given that $T = 95^\circ$ and triangle ABC is isosceles with $AB = AC$, $\angle BAC$ is trisected by AM and AN. Let $\angle BAC = 3\theta$; then $\angle BAM = \angle MAN = \angle NAC = \theta$. Since $\angle AMC = 95^\circ$, $$\theta + \angle ACM + 95^\circ = 180^\circ \implies \angle ACM = 85^\circ - \theta.$$ Since $AB = AC$, let $\angle ABC = \angle ACB = x$; then $$3\theta + x + x = 180^\circ \implies x = 90^\circ - \tfrac{3\theta}{2}.$$ We also have $\angle ACM = \angle ACB - \angle NCM$; since $\angle NCM = \angle NAC = \theta$, $\angle ACM = x - \theta$. Substituting the earlier value gives $$85^\circ - \theta = x - \theta \implies x = 85^\circ.$$ Therefore $U = 85^\circ$ and the final answer is $\boxed{85}$.	×
Analysis	Since $M \in \overline{BC}$, rays CM and CB coincide, so $\angle ACM = \angle ACB$. PEARL respects this and uses two compatible equations. The baselines treat $\angle ACM$ and $\angle ACB$ as distinct, introducing a spurious component at C and arriving at 85°.	–

Table 12: Retained generated outputs for Olympiad-465.

Entry	Retained generated output	Verdict
Problem	Three of four identical glasses are full of orange juice; the fourth is at $1/3$ capacity. What fraction m/n of a glass should be poured from each full glass into the fourth so all are equal? Report $m+n$ (gold $= 7$).	–
PEARL	Mrs. Jones has four identical glasses. The first three glasses are full, so each contains 1 unit of juice, while the fourth glass contains $\frac{1}{3}$ unit. Let x be the amount poured from each full glass into the fourth glass. After pouring, each of the first three glasses contains $1-x$ units, and the fourth glass contains $\frac{1}{3}+3x$ units. Since all four glasses must contain the same amount of juice, we have $$1-x=\frac{1}{3}+3x.$$ Solving gives $$1-\frac{1}{3}=3x+x, \qquad \frac{2}{3}=4x, \qquad x=\frac{2}{3}\cdot\frac{1}{4}=\frac{1}{6}.$$ Thus the fraction of a glass poured from each full glass is $\frac{1}{6}$. Hence $m=1$, $n=6$, and $m+n=7$.	✓
GRPO	Mrs. Jones has four identical glasses. The first three glasses are full, so each contains 1 unit of juice, while the fourth glass contains $\frac{1}{3}$ unit. Let x be the amount poured from each full glass into the fourth glass. After pouring, each of the first three glasses contains $1-x$ units, and the fourth glass contains $\frac{1}{3}+3x$ units. Since all four glasses must contain the same amount of juice, we have $$1-x=\frac{1}{3}+3x.$$ Solving gives $$1-\frac{1}{3}=3x+x, \qquad \frac{2}{3}=4x, \qquad x=\frac{2}{3}\cdot\frac{1}{4}=\frac{1}{6}.$$ Therefore, the fraction of a glass poured from each full glass is $\frac{1}{6}$. Since $m=1$ and $n=6$, the requested value is $m+n=7$.	✓
DAPO	Mrs. Jones has four identical glasses. The first three glasses are full, so each contains 1 unit of juice, while the fourth glass contains $\frac{1}{3}$ unit. Let x be the amount poured from each full glass into the fourth glass. After pouring, each of the first three glasses contains $1-x$ units, and the fourth glass contains $\frac{1}{3}+3x$ units. Since all four glasses must contain the same amount of juice, we have $$1-x=\frac{1}{3}+3x.$$ Solving gives $$1-\frac{1}{3}=3x+x, \qquad \frac{2}{3}=4x, \qquad x=\frac{2}{3}\cdot\frac{1}{4}=\frac{1}{6}.$$ Thus the fraction of a glass poured from each full glass is $\frac{1}{6}$. Hence $m=1$, $n=6$, and $m+n=7$.	✓
GSPO	The total amount of juice is $$3+\frac{1}{3}=\frac{10}{3}$$ glasses. After redistribution, each of the four glasses should contain $$\frac{10}{3}\div 4=\frac{10}{12}=\frac{5}{6}$$ glasses of juice. The fourth glass currently contains $\frac{1}{3}$ glass of juice, so it needs $$\frac{5}{6}-\frac{1}{3}=\frac{1}{2}$$ more glass of juice. This amount is supplied equally by the first three glasses, so each full glass must provide $$\frac{1}{2}\div 3=\frac{1}{6}.$$ Therefore, the required fraction is $\frac{1}{6}$, so $m=1$, $n=6$, and $m+n=7$.	✓
Analysis	All methods take "a glass" as the unit and solve correctly.	–

Table 13: Retained generated outputs for the original AMC23-22 input.

Entry	Retained generated output	Verdict
Problem	Mrs. Jones is pouring orange juice into four identical glasses for her four sons. She fills the first three glasses completely but runs out of juice when the fourth glass is only $\frac{1}{3}$ full. What fraction of a glass must Mrs. Jones pour from each of the first three glasses into the fourth glass so that all four glasses will have the same amount of juice? The final answer can be written in the form $\frac{m}{n}$, where m and n are relatively prime positive integers. What is $m+n$? (Gold $= 7$.)	–
PEARL	Let's denote the capacity of each mug as C. Since Mrs. Smith filled the first three mugs to the brim, the total amount of lemonade in these three mugs is $3C$. The fourth mug is filled only up to $\frac{1}{3}$ of its capacity, so the amount of lemonade in the fourth mug is $\frac{1}{3}C$. The total amount of lemonade available is: $$3C + \frac{1}{3}C = \frac{9C}{3} + \frac{1}{3}C = \frac{10C}{3}.$$ Since Mrs. Smith wants to distribute the lemonade equally among the four mugs, each mug should contain: $$\frac{\frac{10C}{3}}{4} = \frac{10C}{12} = \frac{5C}{6}.$$ The fourth mug currently has $\frac{1}{3}C$, so it needs: $$\frac{5C}{6} - \frac{1}{3}C = \frac{5C}{6} - \frac{2C}{6} = \frac{3C}{6} = \frac{1}{2}C.$$ Therefore, Mrs. Smith needs to transfer $\frac{1}{2}C$ of lemonade from the first three mugs to the fourth mug. Since there are three mugs, the amount transferred from each of the first three mugs is: $$\frac{\frac{1}{2}C}{3} = \frac{1}{6}C.$$ Thus, Mrs. Smith should transfer $\frac{1}{6}$ of the capacity of each of the first three mugs into the fourth mug. The fraction $\frac{1}{6}$ is already in its simplest form, so $m = 1$ and $n = 6$. The sum $m + n$ is: $$1 + 6 = 7.$$ Therefore, the answer is $\boxed{7}$.	✓
GRPO	To determine how much lemonade Mrs. Smith should transfer from each of the first three mugs to the fourth mug so that each mug contains an equal amount of lemonade, we start by defining the total amount of lemonade as V. Since the mugs are identical, each mug has a capacity of $\frac{V}{4}$. The first three mugs are filled to the brim, so each contains $\frac{V}{4}$ of lemonade. The fourth mug is filled to $\frac{1}{3}$ of its capacity, so it contains $\frac{V}{4} \times \frac{1}{3} = \frac{V}{12}$ of lemonade. The total amount of lemonade in all four mugs is: $$3 \times \frac{V}{4} + \frac{V}{12} = \frac{3V}{4} + \frac{V}{12} = \frac{9V}{12} + \frac{V}{12} = \frac{10V}{12} = \frac{5V}{6}.$$ Since Mrs. Smith wants to redistribute the lemonade equally, each mug should contain: $$\frac{\frac{5V}{6}}{4} = \frac{5V}{24}.$$ Currently, the fourth mug contains $\frac{V}{12}$, so the amount that needs to be transferred to the fourth mug is: $$\frac{5V}{24} - \frac{V}{12} = \frac{5V}{24} - \frac{2V}{24} = \frac{3V}{24} = \frac{V}{8}.$$ Since the total transferred amount is $\frac{V}{8}$, and there are three mugs from which it is transferred, the amount transferred from each of the first three mugs is: $$\frac{\frac{V}{8}}{3} = \frac{V}{24}.$$ Therefore, the fraction of the lemonade from each of the first three mugs that should be transferred is $\frac{1}{24}$. The sum of the numerator and denominator is: $$1 + 24 = 25.$$ The final answer is $\boxed{25}$.	×

Table 14: Retained generated outputs for the perturbed AMC23-22 input (Part 1: PEARL and GRPO); the original is in Table 13, and the remaining methods are in Table 15.

Entry	Retained generated output	Verdict
DAPO	To determine how much lemonade Mrs. Smith should transfer from each of the first three mugs to the fourth mug so that each mug contains an equal amount, we start by defining the total amount of lemonade as L. Since the first three mugs are filled to the brim and the fourth is filled to $\frac{1}{3}$ of its capacity, the amount of lemonade in each mug is: $$\text{First mug: } \frac{L}{4}, \quad \text{Second mug: } \frac{L}{4}, \quad \text{Third mug: } \frac{L}{4}, \quad \text{Fourth mug: } \frac{L}{12}.$$ The total amount of lemonade is: $$L = \frac{L}{4} + \frac{L}{4} + \frac{L}{4} + \frac{L}{12} = \frac{3L}{4} + \frac{L}{12} = \frac{9L}{12} + \frac{L}{12} = \frac{10L}{12} = \frac{5L}{6}.$$ Since Mrs. Smith wants to redistribute the lemonade so that each mug contains an equal amount, each mug should contain: $$\frac{L}{4}.$$ Let x be the amount transferred from each of the first three mugs to the fourth mug. After transfer, the first three mugs contain $\frac{L}{4} - x$, and the fourth contains $\frac{L}{12} + 3x$. For the fourth mug: $$\frac{L}{12} + 3x = \frac{L}{4}.$$ Solving: $$\frac{L}{12} + 3x = \frac{3L}{12}, \quad 3x = \frac{2L}{12} = \frac{L}{6}, \quad x = \frac{L}{18}.$$ Therefore, the fraction to be transferred is $\frac{1}{18}$, and $$m + n = 1 + 18 = 19.$$ The final answer is $\boxed{19}$.	×
GSPO	To determine how much lemonade Mrs. Smith should transfer from each of the first three mugs to the fourth mug so that each mug contains an equal amount of lemonade, we start by defining the total amount of lemonade as V. Since the mugs are identical, each mug has a capacity of $\frac{V}{4}$. The first three mugs are filled to the brim, so each contains $\frac{V}{4}$. The fourth mug contains $\frac{V}{4} \times \frac{1}{3} = \frac{V}{12}$. The total amount in all four mugs is: $$3 \times \frac{V}{4} + \frac{V}{12} = \frac{3V}{4} + \frac{V}{12} = \frac{9V}{12} + \frac{V}{12} = \frac{10V}{12} = \frac{5V}{6}.$$ Since the lemonade is to be distributed equally among four mugs, each mug should contain: $$\frac{\frac{5V}{6}}{4} = \frac{5V}{24}.$$ Currently, the fourth mug contains $\frac{V}{12}$. The amount needed is: $$\frac{5V}{24} - \frac{V}{12} = \frac{5V}{24} - \frac{2V}{24} = \frac{3V}{24} = \frac{V}{8}.$$ Since this amount comes from the first three mugs, the amount transferred from each is: $$\frac{\frac{V}{8}}{3} = \frac{V}{24}.$$ Therefore, the fraction to transfer is $\frac{1}{24}$, and $$1 + 24 = 25.$$ The final answer is $\boxed{25}$.	×
Analysis	The perturbation only renames objects (mug for glass) but preserves capacities, fill levels, and the question. PEARL keeps a single unit "mug capacity C" and obtains the same fraction as in the original form. The baselines collapse two distinct quantities—total lemonade and per-mug capacity—into one variable V (or L), and the resulting unit inconsistency changes the computed transfer fraction. DAPO additionally targets a full mug rather than the average.	–

Table 15: Retained generated outputs for the perturbed AMC23-22 input (Part 2: DAPO, GSPO, and Analysis); see Table 14 for Part 1.

Statistic	Value	Statistic	Value
Probe start	Step 10	Probe interval	5 optimization steps
Candidates per probe	8 hard clusters	Variants judged per cluster	Up to 2
Judge confidence threshold	0.5	Maximum hard ratio	0.35
Hard candidates probed	1432	Accepted by judge	530
Rejected by judge	902	Acceptance rate	37.0%
Unlocked hard pool at step 900	525 clusters	Actual hard replacements	465
Logged optimization steps	180	Avg. replacements per logged step	2.58

Table 16: Hard-data injection statistics up to the step-900 checkpoint of the diagnostic curriculum-reinjection variant on the Qwen2.5-Math-7B backbone.

Method	AIME25	AIME24	AMC23	MATH500	Minerva	Olympiad	Avg.
PEARL	14.16	24.16	67.50	79.60	36.03	42.58	44.01
PEARL + judged hard injection	13.33	20.83	67.50	80.85	37.06	45.99	44.26

Table 17: Diagnostic result for conservative hard-data injection on the Qwen2.5-Math-7B backbone.