Expanding the Solvable Frontier: Policy-Adaptive Curriculum Learning for Reasoning RL

Anonymous ACL submission

Abstract

Existing curriculum learning methods for reasoning reinforcement learning often treat problem difficulty as a static property of the problem. We argue that difficulty is instead policy-dependent: problems beyond the current policy provide sparse learning signals, while problems already solved by greedy decoding may still yield informative stochastic trajectories during RL. We propose PAD-Curriculum, a policy-adaptive curriculum framework for GRPO that estimates the student's evolving solvable frontier from student-generated trajectories. For currently solved problems, PAD-Curriculum ranks examples by a normalized Scorer-to-Student KL discrepancy computed against a frozen stronger scorer; unsolved problems are deferred and periodically re-evaluated as the policy improves. Experiments on GSM8K, MATH, and NuminaMath-CoT with Llama-3.1 and Qwen3 models show consistent gains over standard GRPO and curriculum/distillation baselines, with particularly strong improvements on challenging mathematical reasoning benchmarks. Analyses further show that adaptive discrepancy estimation is more effective than static scorer-based difficulty.

1 Introduction

Large language models (LLMs) have shown strong reasoning abilities (Chen et al., 2025; Ferrag et al., 2025; Li et al., 2025; Wang et al., 2026, 2022), but reinforcement learning (RL) for reasoning remains challenging due to sparse outcome rewards (Müller et al., 2025; Zhong et al., 2026). In mathematical reasoning, a model typically receives feedback only after producing a complete solution, making it difficult to learn effectively from problems that are far beyond its current capability. Curriculum learning offers a natural way to mitigate this issue by organizing training examples from easier to harder ones (Lightman et al., 2023; Milani et al., 2024; Wang et al., 2025). Existing curriculum methods for reasoning often depend on external or static difficulty signals. Some approaches rely on stronger models to assign problem-level difficulty scores (Shi et al., 2025), while others use annotated reasoning steps or process supervision to control training difficulty (Amani et al., 2025; Xi et al., 2024). Although useful, these signals may not align with the actual state of the student model. Figure 1 illustrates this mismatch: the same problem can have different effective difficulty for a scoring model and for students with different capability levels. A problem that appears easy to a stronger scoring model may still be too difficult for a weaker student, while a problem that a stronger student can already solve may still provide useful learning signal. Moreover, static difficulty labels do not adapt as the student policy improves.

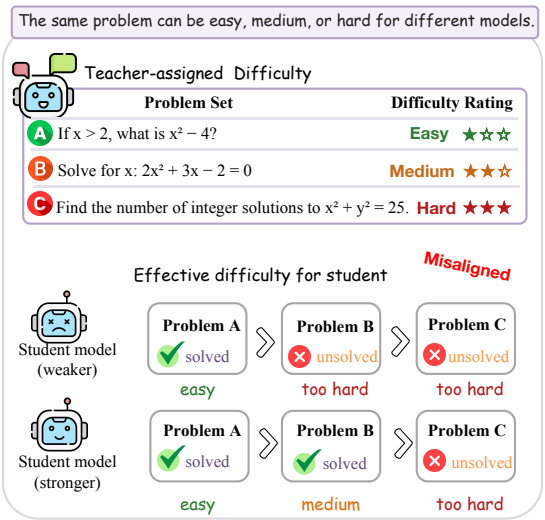

Figure 1: Illustration of model-dependent difficulty. A fixed externally assigned difficulty label may be misaligned with the current student capability, motivating curriculum construction relative to the student policy.

We argue that curriculum difficulty in reasoning RL should be defined relative to the current student policy. A problem that the student can solve

provides a reliable trajectory for estimating how close the student's behavior is to the frozen curriculum scorer, whereas a problem that the student currently fails to solve may not provide a meaningful fine-grained difficulty estimate. Based on this view, we propose **PAD-Curriculum**, a policy-adaptive discrepancy-guided curriculum learning framework for GRPO.

The key distinction is that PAD-Curriculum does not ask which problems are globally easy; instead, it asks which problems are currently learnable for the current policy. To this end, PAD-Curriculum builds the curriculum from student-generated trajectories. The current policy first generates a response for each problem, and a verifier determines whether the final answer is correct. For currently solved problems, PAD-Curriculum computes a normalized KL discrepancy between the student and a frozen curriculum scorer along the student trajectory, and sorts these problems from low to high discrepancy. Currently unsolved problems are deferred and re-evaluated in later stages. This creates an easy-to-hard schedule within the student's evolving solvable frontier. Importantly, the scorer is used only for scheduling; it is not used as a distillation target or as the KL-regularization reference policy in GRPO.

Experiments on GSM8K, NuminaMath-CoT, and MATH show that PAD-Curriculum improves over standard GRPO and competitive baselines across multiple model families. Additional analyses indicate that student-dependent discrepancy provides a more effective scheduling signal than static scorer-model difficulty estimates. Our main contributions are summarized as follows:

- **A student-adaptive formulation of curriculum difficulty for reasoning RL.** We operationalize policy-dependent difficulty through the student's evolving solvable frontier, which is repeatedly estimated from student-generated trajectories rather than fixed difficulty labels.

- **A discrepancy-guided scheduling signal without token-level imitation.** We introduce a normalized Scorer→Student KL discrepancy score, $D_{\mathrm{KL}}(\pi_{\mathrm{score}} \| \pi_\theta)$, computed on correctly solved student trajectories. This score identifies solvable yet informative problems for curriculum scheduling, while unsolved problems are deferred and re-evaluated as the policy improves. Unlike distillation methods, the frozen curriculum scorer is used only for scheduling and does not provide token-level training targets.

- **Comprehensive validation across models and mathematical reasoning benchmarks.** We evaluate PAD-Curriculum on GSM8K, MATH, and NuminaMath-CoT using Llama-3.1 and Qwen3 model families. Results show improvements over GRPO and competitive baselines, and ablations validate the complementary roles of correctness-aware filtering and discrepancy-guided ordering.

2 Related Work

Curriculum Learning for Reasoning. Curriculum learning presents examples in an organized order to mitigate sparse learning signals in reasoning tasks (Rane, 2020; Soviany et al., 2022; Parashar et al., 2025). Recent reasoning curricula often rely on external difficulty estimators or process-level supervision, such as stronger-model scores, golden trajectories, or annotated reasoning steps (Shi et al., 2025; Xi et al., 2024; Amani et al., 2025). PAD-Curriculum instead estimates difficulty from the interaction between the current student policy, a verifier, and a frozen scorer, without requiring annotated reasoning steps.

Self-Generated Data and On-Policy Distillation. Self-generated trajectories have been widely used to reduce reliance on human annotations (Wang et al., 2023; Tan et al., 2024; Fuchs et al., 2025; Karim et al., 2025). Prior work explores filtering, refinement, and distillation over model-generated responses, including STaR (Zelikman et al., 2022), Re-ReST (Dou et al., 2024), and on-policy distillation methods such as GKD (Agarwal et al., 2024). These methods primarily treat self-generated trajectories as training targets or filtering units, whereas PAD-Curriculum uses them to dynamically schedule optimization during reinforcement learning.

Reinforcement Learning with Outcome Supervision. Outcome-supervised reinforcement learning has become an effective paradigm for improving reasoning models, especially in domains with verifiable final answers (Eschmann, 2021; Ladosz et al., 2022; Li et al., 2021; Wen et al., 2025). Prior work shows that outcome supervision can be competitive with process supervision (Lightman et al., 2023), and policy optimization methods such as PPO (Schulman et al., 2017) and GRPO (Shao

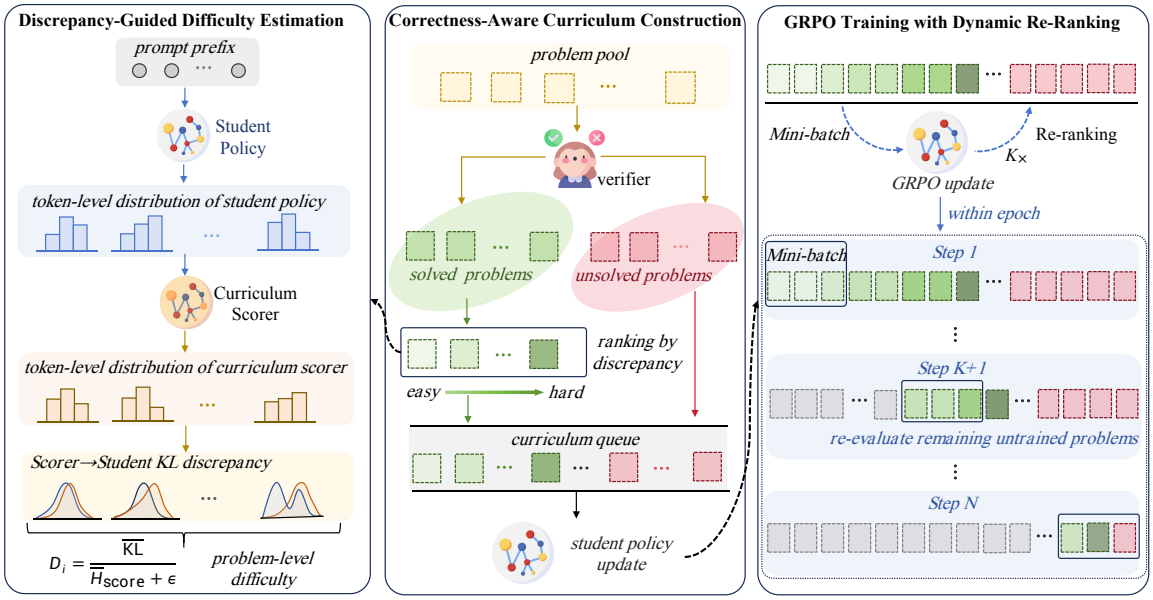

Figure 2: Overview of PAD-Curriculum. The student policy first generates a response for each problem, and a verifier separates solved from unsolved problems. Solved problems are ranked by normalized student–scorer discrepancy, while unsolved problems are deferred to later stages. During GRPO training, the remaining queue is re-evaluated and re-ranked every K policy updates as the student policy improves.

et al., 2024) have been applied to mathematical reasoning. However, outcome rewards are often sparse: when the policy is trained on problems it cannot yet solve, sampled responses provide limited learning signal. PAD-Curriculum mitigates this issue without modifying the GRPO objective. Instead, it dynamically reorders problems according to the current policy's capability, allowing the model to consolidate currently solvable examples before gradually moving to harder ones.

3 Method

We propose **PAD-Curriculum**, a policy-adaptive discrepancy-guided curriculum method for GRPO. Figure 2 illustrates the overall procedure. Algorithm 1 summarizes training, and Algorithm 2 details discrepancy-guided evaluation and sorting.

3.1 Preliminaries

We consider a dataset $\mathcal{D} = \{(q_i, y_i)\}_{i=1}^{N}$, where q_i denotes a problem prompt and y_i is the ground-truth final answer. The goal is to train a policy model π_θ to generate responses that can be verified as correct. We assume access to a verifier $R(q, o, y)$ that assigns an outcome reward to a response o. In our mathematical reasoning setting, R is a rule-based answer checker that returns a binary reward:

$$r_{i,g} = R(q_i, o_{i,g}, y_i) \in \{0, 1\}. \tag{1}$$

During GRPO training, for each problem q_i, the current policy samples a group of G responses:

$$\{o_{i,1}, \ldots, o_{i,G}\} \sim \pi_\theta(\cdot \mid q_i), \tag{2}$$

which are scored by the verifier to compute group-relative advantages.

PAD-Curriculum additionally uses a stronger frozen curriculum scorer π_{score} to estimate the discrepancy between the current student policy and a more capable model distribution. The scorer is used only for curriculum construction; it is not used as a distillation target or as the KL-regularization reference policy in the GRPO objective.

3.2 Discrepancy-Guided Difficulty Estimation

PAD-Curriculum estimates difficulty relative to the current student policy and periodically updates it throughout training. For a problem q_i, the student model first generates a single response using greedy decoding for curriculum scoring:

$$s_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,L_i}) \leftarrow \mathrm{Decode}(\pi_\theta, q_i), \tag{3}$$

where s_i denotes the generated response tokens. We then apply a correctness verifier:

$$c_i = \mathcal{V}(q_i, s_i), \tag{4}$$

where $c_i \in \{0, 1\}$ indicates whether the final answer is correct.

Correctness-aware filtering. If $c_i = 0$, the problem is considered currently outside the student's solvable frontier. We do not assign it a fine-grained discrepancy score, because token-level discrepancy on a failed trajectory can be difficult to interpret. A failed solution may contain arithmetic mistakes, invalid reasoning steps, or spurious intermediate claims; in such cases, a large KL value may reflect arbitrary generation errors rather than a meaningful curriculum signal. PAD-Curriculum defers these problems to the end of the queue and revisits them after the policy has been updated. This design allows the curriculum to expand naturally: a problem that is currently unsolved can later enter the ranked region once the student begins to solve it.

It is important to note that $c_i = 1$ does not imply that the problem is already saturated for GRPO training. The curriculum-scoring rollout is generated by greedy decoding, whereas GRPO training samples a group of stochastic responses for the same problem. Therefore, a greedy-correct problem may still produce mixed outcome rewards across sampled trajectories, resulting in nonzero group reward variance and meaningful group-relative advantages. In this sense, correctness-aware filtering identifies examples that are reliable enough to schedule, not examples that no longer require learning.

Discrepancy scoring for solved problems. If $c_i = 1$, the problem is within the student's currently solvable region. We then measure how much the student's token-level behavior differs from a stronger frozen curriculum scorer along the student-generated trajectory. Low discrepancy indicates that the student already behaves similarly to the curriculum scorer on the current trajectory, whereas high discrepancy identifies solvable yet insufficiently consolidated behaviors.

Let $x_{i,k} = (q_i, s_{i,<k})$ denote the prefix up to position $k-1$. We define the **Scorer→Student KL** discrepancy as the KL divergence from the frozen curriculum scorer to the current student policy along the student-generated trajectory:

$$\overline{\mathrm{KL}}_i = \frac{1}{L_i} \sum_{k=1}^{L_i} D_{\mathrm{KL}} \left(\pi_{\mathrm{score}}(\cdot \mid x_{i,k}) \parallel \pi_\theta(\cdot \mid x_{i,k}) \right). \tag{5}$$

We use the KL direction $D_{\mathrm{KL}}(\pi_{\mathrm{score}} \| \pi_\theta)$ because it highlights tokens where the curriculum scorer assigns high probability but the student does not. These positions correspond to residual behavioral gaps on trajectories that the student can already solve. In contrast, the Student→Scorer KL, $D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{score}})$, emphasizes tokens to which the student already assigns high probability. Such tokens may reflect the student's current preferences rather than missing probability mass under the scorer, making this direction less suitable for identifying solvable but under-consolidated examples. To reduce the effect of positions where the curriculum scorer is intrinsically uncertain, we normalize the average KL by the average scorer entropy:

$$\overline{H}_{\mathrm{score},i} = \frac{1}{L_i} \sum_{k=1}^{L_i} H\big(\pi_{\mathrm{score}}(\cdot \mid x_{i,k})\big). \tag{6}$$

The final discrepancy-guided difficulty score for correctly answered problems is:

$$D_i = \frac{\overline{\mathrm{KL}}_i}{\overline{H}_{\mathrm{score},i} + \epsilon}, \tag{7}$$

where ϵ is a small constant for numerical stability. The averaging over response length avoids assigning higher difficulty merely because a response contains more tokens.

3.3 Curriculum Construction

After computing discrepancy-guided difficulty scores for correctly solved problems, PAD-Curriculum constructs the training queue in two parts. First, correctly solved problems are sorted by their difficulty scores D_i in ascending order. Second, incorrectly solved problems are appended to the end of the queue without assigning a fine-grained ordering among them.

This creates an easy-to-hard schedule within the current solvable frontier: lower-discrepancy solvable problems are optimized earlier, while unresolved problems are deferred and periodically revisited as the policy improves. We use ascending discrepancy because PAD-Curriculum follows a competence-based curriculum rather than a hard-example mining strategy. Low-discrepancy solved problems correspond to more stable behaviors that the student can already reproduce relatively well, whereas high-discrepancy solved problems are closer to the edge of the current solvable frontier. Training from low to high discrepancy allows the policy to first consolidate reliable reasoning behaviors before moving toward solvable but under-consolidated examples.

Importantly, PAD-Curriculum does not define an absolute difficulty level for each problem. Difficulty is defined relative to the current student policy.

As training progresses, a problem may move from the deferred region into the ranked region once the student begins to solve it correctly.

3.4 Dynamic Re-Ranking

Given the sorted queue $\mathcal{Q}_{\text{sorted}}$, training proceeds by taking mini-batches from the front of the queue. Each problem is used at most once per epoch. For each mini-batch, the student samples G responses per problem, receives verifier rewards, and performs a GRPO update.

To keep the curriculum aligned with the evolving policy, PAD-Curriculum re-ranks the remaining problems every K policy updates. For each problem, the current student generates a response, the verifier checks correctness, and the discrepancy score is recomputed for correctly solved problems. The queue is then rebuilt by sorting solved problems according to their updated discrepancy scores, followed by unresolved problems. This periodic re-ranking prevents the curriculum from becoming stale while avoiding recomputation for problems already consumed in the current epoch. We set $K = 40$ in our main experiments and study the effect of re-ranking frequency in Appendix C.

3.5 Discussion

PAD-Curriculum continuously updates curriculum difficulty relative to the evolving student policy rather than relying on fixed difficulty labels. Unlike distillation methods, the curriculum scorer is used only for curriculum scoring rather than token-level supervision, while the student is optimized entirely through GRPO rewards on its own sampled trajectories. We distinguish the frozen curriculum scorer from the reference policy commonly used for KL regularization in RLHF-style objectives. The scorer is not part of the GRPO optimization objective and does not constrain the student policy during policy updates; it only provides a difficulty signal for ordering training problems.

Our method assumes access to a reliable correctness verifier, such as rule-based checking for mathematical reasoning or code generation. Periodic re-ranking introduces additional computation due to student rollouts and scorer-model forward passes. This cost is controlled by the re-ranking interval K and by correctness-aware filtering, since scorer-model discrepancy is computed only for problems solved by the current student policy. We provide a theoretical cost analysis in Appendix E.

4 Experiments

4.1 Experimental Setup

Datasets. Our experiments are conducted on three widely adopted mathematical reasoning benchmarks: **GSM8K** (grade-school math word problems) (Cobbe et al., 2021), **MATH** (high-school-level competition math) (Hendrycks et al.), and **NuminaMath-CoT** (which includes problems from AMC/AIME, the AoPS forum, Chinese K12, Olympiad competitions, and synthetic sources) (Li et al., 2024).

Baselines. We compare against five baselines: (1) **Original**, the pretrained model without any fine-tuning; (2) **GRPO** (Shao et al., 2024), standard Group Relative Policy Optimization applied without curriculum learning; (3) **RFT** (Yuan et al., 2023), supervised fine-tuning using self-generated correct reasoning trajectories; (4) **AdaBack** (Amani et al., 2025), a curriculum method that controls difficulty by progressively providing correct reasoning prefixes. AdaBack relies on ground-truth reasoning prefixes and represents curriculum methods with process-level supervision; and (5) **OPD** (Agarwal et al., 2024), a post-training method that minimizes reverse KL divergence between the student's and teacher's output distributions using self-generated trajectories. Full dataset and implementation details are provided in Appendix A.

4.2 Main Results

We evaluate all methods on Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen3-8B-Instruct (Yang et al., 2025). Tables 1 and 2 report the main results. All reported numbers are averaged over three random seeds.

PAD-Curriculum improves over GRPO and curriculum baselines. PAD-Curriculum achieves the strongest average performance among the compared methods across the evaluated model families and benchmarks. The improvements are observed on grade-school problems, competition-level mathematics, and several challenging NuminaMath-CoT subsets. These results suggest that adapting the curriculum to the current student policy provides useful training signal beyond standard GRPO and externally defined curriculum baselines.

Comparison among baselines reveals complementary strengths. All strong baselines improve over the original model on most benchmarks, confirming the effectiveness of outcome-supervised RL

Model	Method	GSM8K	MATH	AMC/AIME	Olympiads
Llama-3.1-8B-Instruct	Original	62.09	39.67	1.03	9.68
	GRPO	88.07	37.48	9.80	12.47
	RFT	82.19	40.35	13.74	11.73
	OPD	86.47	40.96	14.28	13.47
	AdaBack	90.98	39.82	11.07	13.53
	PAD-Curriculum	**92.51**	**42.13**	**16.67**	**15.71**
Qwen3-8B-Instruct	Original	83.71	59.18	16.67	14.93
	GRPO	92.90	65.59	32.35	22.55
	RFT	90.33	66.17	27.63	23.89
	OPD	90.86	64.93	30.63	24.11
	AdaBack	94.01	67.13	33.36	25.46
	PAD-Curriculum	**95.83**	**70.41**	**37.67**	**27.05**

Table 1: Main results on GSM8K, MATH, and selected subsets of NuminaMath-CoT.

Model	Method	AoPS	CN K12	ORCA	Syn AMC	Syn Math
Llama-3.1-8B-Instruct	Original	8.22	27.26	63.71	17.71	47.94
	GRPO	10.96	30.93	64.99	20.14	48.45
	RFT	14.06	29.91	67.14	18.52	48.77
	OPD	13.91	30.75	66.25	19.68	49.01
	AdaBack	13.21	31.09	64.71	19.42	49.48
	PAD-Curriculum	**15.77**	**33.31**	**68.41**	**22.06**	**50.55**
Qwen3-8B-Instruct	Original	3.01	45.73	75.08	17.27	55.31
	GRPO	15.07	51.92	80.39	23.74	61.60
	RFT	15.33	48.15	83.31	24.35	59.23
	OPD	15.81	50.32	82.46	25.19	61.29
	AdaBack	17.81	53.85	81.23	27.34	61.86
	PAD-Curriculum	**19.06**	**55.65**	**83.92**	**28.78**	**62.84**

Table 2: Main results on other subsets of NuminaMath-CoT.

and curriculum-based optimization for reasoning tasks. Standard GRPO can be unstable on challenging settings where many problems remain outside the policy's current capability. PAD-Curriculum consistently achieves the strongest overall performance, particularly on more difficult reasoning benchmarks.

Gains are more pronounced on challenging subsets. PAD-Curriculum shows clear improvements on MATH and several difficult NuminaMath-CoT subsets, including Olympiads and AoPS. The AMC/AIME subset follows a similar trend, but it contains only a small number of test examples in our split and should therefore be interpreted as auxiliary evidence. Overall, the results indicate that policy-adaptive curriculum scheduling is especially helpful when problems require more complex multi-step reasoning.

Results are consistent across model families. The improvements of PAD-Curriculum are observed for both Llama-3.1-8B-Instruct and Qwen3-8B-Instruct. Although Qwen3-8B-Instruct achieves higher absolute accuracy across most benchmarks, PAD-Curriculum improves both model families over their corresponding GRPO baselines. This suggests that the proposed curriculum strategy is not tied to a single model family, although further experiments on additional architectures would be needed to establish broader generality.

4.3 Static vs. Adaptive Difficulty Estimation

To isolate the effect of policy-adaptive discrepancy-guided difficulty estimation, we compare PAD-Curriculum against a static scorer-based curriculum on Qwen3-8B-Instruct. Both methods use Qwen3-72B-Instruct as the frozen curriculum scorer. For the static baseline, we roll out ten trajectories from Qwen3-72B-Instruct for each

Method	GSM8K	MATH
Original	83.71	59.18
Static Difficulty	93.47	65.31
PAD-Curriculum	95.83	70.41

Table 3: Comparison between static and adaptive difficulty estimation on Qwen3-8B-Instruct.

problem and compute the scorer accuracy $\hat{a}_i = \frac{1}{10}\sum_{m=1}^{10}\mathcal{V}(q_i, o_{i,m})$. Problems with higher scorer accuracy are treated as easier, and the static curriculum sorts problems from high to low $\hat{a}_i$. The resulting ordering is computed once before training and remains fixed throughout optimization.

As shown in Table 3, the static scorer-based curriculum improves over the original model but underperforms PAD-Curriculum. This result suggests that using a stronger scoring model alone is not sufficient for effective curriculum construction. A problem that is easy for the Qwen3-72B-Instruct scorer may still be difficult for an 8B student, while a problem solved by the student may still expose meaningful discrepancies in intermediate token distributions. PAD-Curriculum captures this student-dependent difficulty by computing discrepancy along the student's own generated trajectory and updating the ordering as the policy evolves. The component ablation in Section 4.4 further shows that both correctness-aware filtering and discrepancy-guided ordering contribute to the final performance, while Appendix B studies the effect of KL direction.

4.4 Component Ablation

Method	GSM8K	MATH
Original	83.71	59.18
GRPO	92.90	65.59
PAD w/o Discrepancy	94.61	67.07
PAD w/o Filtering	95.04	69.38
PAD-Curriculum	**95.83**	**70.41**

Table 4: Ablation of correctness-aware filtering and discrepancy-guided ordering on Qwen3-8B-Instruct.

To further isolate the contributions of correctness-aware filtering and discrepancy-guided ordering, we compare PAD-Curriculum with two ablated variants. **PAD w/o Discrepancy** keeps correctness-aware filtering: currently solved problems are placed before currently unsolved ones, but solved problems are randomly ordered rather than ranked by student–scorer discrepancy. **PAD w/o Filtering** keeps discrepancy-guided ordering but removes correctness-aware filtering, ranking all problems solely by their normalized KL discrepancy regardless of whether the current student solves them correctly. For this variant, we compute the same discrepancy score on the student's generated trajectory for every problem, including incorrectly solved ones. These two variants disentangle frontier identification from within-frontier ordering. PAD w/o Discrepancy isolates the effect of correctness-aware frontier filtering by preserving the solved-before-unsolved curriculum structure while removing fine-grained discrepancy-based ordering. PAD w/o Filtering isolates the effect of discrepancy-guided ordering by applying the normalized KL score to all problems regardless of current correctness. Together, these variants test whether the gains of PAD-Curriculum come from identifying the current solvable frontier, ordering examples within that frontier, or the combination of both.

As shown in Table 4, both ablated variants improve over standard GRPO, indicating that correctness-aware filtering and discrepancy-guided ordering each provide useful curriculum signals. PAD w/o Discrepancy improves over GRPO by 1.71 points on GSM8K and 1.48 points on MATH, showing that postponing currently unsolved problems is beneficial even without fine-grained discrepancy ordering. PAD w/o Filtering further improves performance to 95.04 on GSM8K and 69.38 on MATH, suggesting that student–scorer discrepancy provides an effective ordering signal. However, PAD-Curriculum achieves the best results, outperforming PAD w/o Filtering by 0.79 points on GSM8K and 1.03 points on MATH. These results suggest that the two components are complementary: discrepancy-guided ordering provides a fine-grained curriculum signal, while correctness-aware filtering prevents currently unsolved problems from dominating the early curriculum.

4.5 Performance by Difficulty Level

To further understand how PAD-Curriculum improves reasoning capabilities across different difficulty levels, we conduct a fine-grained analysis on the MATH dataset using its pre-annotated difficulty ratings. The test set is divided into five tiers from L1 (easiest) to L5 (hardest). Figure 3 reports the accuracy improvement of each method over the

Model	Method	GSM8K	MATH	AMC/AIME	Olympiads
Qwen3-14B-Instruct	Original	84.53	65.06	20.51	19.17
	GRPO	94.24	68.90	38.72	23.87
	PAD-Curriculum	**96.13**	**71.98**	**40.07**	**28.76**

Table 5: Model scaling experiments on Qwen3-14B-Instruct.

Model	Method	AoPS	CN K12	ORCA	Syn AMC	Syn Math
Qwen3-14B-Instruct	Original	11.72	47.12	77.11	24.46	60.82
	GRPO	19.18	53.37	81.51	30.53	62.89
	PAD-Curriculum	**21.77**	**56.39**	**85.07**	**32.41**	**64.21**

Table 6: Model scaling experiments on Qwen3-14B-Instruct on remaining subsets of NuminaMath-CoT.

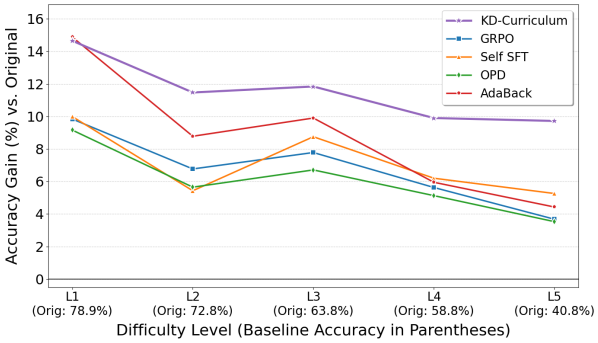

Figure 3: Accuracy improvement over the Original model across difficulty levels on MATH dataset.

Original model across difficulty tiers.

Several observations emerge from this breakdown. All methods exhibit lower accuracy on harder tiers, but PAD-Curriculum maintains substantially stronger relative improvements as difficulty increases. The gains are modest on easier problems but become increasingly pronounced on L4–L5, suggesting that policy-adaptive curriculum scheduling is particularly beneficial when sparse outcome rewards make optimization more difficult. Additional analyses of training dynamics further show that PAD-Curriculum progressively expands the model's solvable frontier during training, particularly on harder reasoning tiers (Appendix D).

4.6 Scaling to Larger Models

To examine whether PAD-Curriculum remains effective for larger models, we conduct additional experiments on Qwen3-14B-Instruct. Tables 5 and 6 present the results.

PAD-Curriculum improves over both the Original model and standard GRPO across the evaluated benchmarks. The gains are visible on standard benchmarks such as GSM8K and MATH, as well as on several NuminaMath-CoT subsets. In particular, the improvements on Olympiads and AoPS suggest that policy-adaptive curriculum scheduling remains useful for more challenging reasoning problems. The AMC/AIME results follow the same trend, but should again be interpreted cautiously due to the small test size.

The results also suggest that the benefits of curriculum scheduling are complementary to increasing model size. Although a stronger base model leaves less room for improvement, the 14B model still benefits from a discrepancy-guided training schedule that adapts to its own learning dynamics. This indicates that PAD-Curriculum can be applied beyond a single model scale.

5 Conclusion

We introduced PAD-Curriculum, a policy-adaptive curriculum framework for reasoning RL. Instead of treating problem difficulty as fixed, PAD-Curriculum tracks the student's evolving solvable frontier and orders currently solvable problems by normalized student–scorer discrepancy. The curriculum scorer is used only for scheduling, while policy learning remains driven by GRPO outcome rewards. Experiments on GSM8K, MATH, and NuminaMath-CoT across Llama-3.1 and Qwen3 model families show consistent improvements over GRPO and competitive curriculum baselines, especially on challenging reasoning benchmarks. These results suggest that curriculum learning for reasoning RL is better viewed as dynamically tracking the current policy's learnable frontier than as constructing a fixed global ordering over problems.

Limitations

PAD-Curriculum assumes access to a stronger frozen curriculum scorer for curriculum construction and a reliable correctness verifier for determining whether student-generated trajectories are solvable. This makes the method naturally suited to domains such as mathematical reasoning and code generation, where rule-based or execution-based verification is available.

Periodic re-ranking introduces additional computation due to student rollouts and scorer-model forward passes. Since the curriculum scorer can be substantially larger than the student model, the practical wall-clock overhead depends on implementation details such as batching, decoding efficiency, hardware configuration, and the re-ranking interval. Future work could explore cheaper discrepancy estimators or cached scorer distributions to further reduce this cost.

Ethics Statement

This work focuses on improving mathematical reasoning through curriculum learning for reinforcement learning. Potential applications include educational tools, automated problem solving, and reasoning-oriented language model training. At the same time, improved reasoning capabilities may be used in broader downstream systems whose impacts depend on deployment context.

Our experiments use publicly available mathematical reasoning datasets, including GSM8K, MATH, and NuminaMath-CoT. We do not collect new human data or use private, sensitive, or personally identifiable information. The proposed method relies on final-answer verification and does not require additional human annotation.

The curriculum scorers used in this work are publicly available open-weight models. We use them only for curriculum scoring rather than for generating token-level supervision targets. Following conference policy, we acknowledge the use of LLM-based assistance for improving the clarity of the writing. The authors have reviewed the AI-assisted content and take full responsibility for the validity, originality, and presentation of the work.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*.

Mohammad Hossein Amani, Aryo Lotfi, Nicolas Mario Baldwin, Samy Bengio, Mehrdad Farajtabar, Emmanuel Abbe, and Robert West. 2025. Rl for reasoning by adaptively revealing rationales. *arXiv preprint arXiv:2506.18110*.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. 2024. Re-rest: Reflection-reinforced self-training for language agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15394–15411.

Jonas Eschmann. 2021. Reward function design in reinforcement learning. *Reinforcement learning algorithms: Analysis and Applications*, pages 25–33.

Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. 2025. Reasoning beyond limits: Advances and open problems for llms. *ICT express*.

Gilad Fuchs, Alex Nus, Yotam Eshel, and Bracha Shapira. 2025. Enhancing data annotation for student models: A self-training approach with large language models. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 2706–2712.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Md Monjurul Karim, Sangeen Khan, Dong Hoang Van, Xinyue Liu, Chunhui Wang, and Qiang Qu. 2025. Transforming data annotation with ai agents: A review of architectures, reasoning, applications, and impact. *Future Internet*, 17(8):353.

Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. 2022. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and 1 others. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9.

Kevin Li, Abhishek Gupta, Ashwin Reddy, Vitchyr H Pong, Aurick Zhou, Justin Yu, and Sergey Levine. 2021. Mural: Meta-learning uncertainty-aware rewards for outcome-driven reinforcement learning. In *International conference on machine learning*, pages 6346–6356. PMLR.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The twelfth international conference on learning representations*.

Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. 2024. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, 56(7):1–36.

Henrik Müller, Lukas Berg, and Daniel Kudenko. 2025. Using incomplete and incorrect plans to shape reinforcement learning in long-sequence sparse-reward tasks. *Neural Computing and Applications*, 37(23):18851–18866.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and 1 others. 2025. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *arXiv preprint arXiv:2506.06632*.

Sunayana Rane. 2020. *Learning with curricula for sparse-reward tasks in deep reinforcement learning*. Ph.D. thesis, Massachusetts Institute of Technology.

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

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*.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. 2025. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520*.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 930–957.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Ziniu Li, Yidi Wang, Shu Yan, Chengxing Jia, Xu-Hui Liu, Xinwei Chen, Jiacheng Xu, and 1 others. 2026. A survey on large language models for mathematical reasoning. *ACM Computing Surveys*, 58(8):1–35.

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

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, and 1 others. 2025. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

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*.

Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin Guo, Junzhe Wang, and 1 others. 2024. Training large language models for reasoning through reverse curriculum reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 54030–54048.

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

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *Preprint*, arXiv:2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Wenlin Zhong, Chengyuan Liu, Yiquan Wu, Bovin Tan, Changlong Sun, Yi Wang, Xiaozhong Liu, and Kun Kuang. 2026. P2s: Probabilistic process supervision for general-domain reasoning question answering. *arXiv preprint arXiv:2601.20649*.

A Experimental Details

A.1 Dataset Details

We conduct our experiments on three widely used mathematical reasoning benchmarks: GSM8K, MATH, and NuminaMath-CoT. The statistics of the training and test sets are presented in Table 7.

GSM8K (Cobbe et al., 2021). GSM8K is a dataset of 8,500 high-quality grade school math problems. Each problem requires 2 to 8 steps to solve, involving basic arithmetic operations. We use the standard training split for training and the test split for evaluation. The evaluation metric is exact match accuracy.

MATH (Hendrycks et al.). MATH is a dataset comprising 12,500 competition-level mathematics problems from AMC 10, AMC 12, AIME, and other contests. It covers seven subjects: algebra, geometry, number theory, combinatorics, prealgebra, calculus, and intermediate algebra. We use the standard training split for training and the test split for evaluation.

NuminaMath-CoT (Li et al., 2024). A large-scale collection of mathematical problems with chain-of-thought annotations, sourced from diverse origins including competition forums, K12 curricula, and synthetic generation. The dataset covers a wide spectrum of difficulty levels, from elementary problems to olympiad-level challenges. In our experiments, we extract 8,000 problems for training and 2,000 problems for testing. The dataset comprises the following seven subsets:

- **cn_k12:** Problems collected from Chinese K12 mathematics curricula, covering grades 1 through 12. This subset represents standard textbook-style problems with moderate difficulty.
- **synthetic_math:** Synthetically generated mathematical problems with step-by-step solutions, covering diverse topics including algebra, geometry, and calculus. This subset provides high-volume training data with consistent formatting.
- **orca_math:** A curated collection of mathematical problems from the Orca dataset, emphasizing multi-step reasoning and diverse problem-solving strategies.
- **olympiads:** Problems sourced from various national and international mathematical olympiads, including the International Mathematical Olympiad (IMO). This subset represents the highest difficulty level, requiring advanced reasoning and creative problem-solving.
- **synthetic_amc:** Synthetically generated problems modeled after the American Mathematics Competitions (AMC), targeting the difficulty range of AMC 10 and AMC 12 contests.
- **aops_forum:** Problems sourced from the Art of Problem Solving (AoPS) online forum, a community-driven platform where participants discuss competition-level mathematics. These problems exhibit high variety and often require non-standard solution approaches.
- **amc_aime:** Problems from actual AMC and AIME (American Invitational Mathematics Examination) contests. This subset provides an authentic competition-style evaluation source, but its small test size in our split means that results should be interpreted as auxiliary evidence.

We remove exact duplicates between training and evaluation splits after normalizing whitespace, LaTeX formatting, and answer markers. For NuminaMath-CoT, we additionally check overlap with GSM8K and MATH evaluation sets using normalized problem statements. Near-duplicate detection is performed using token-level similarity, and examples above a similarity threshold are manually inspected or removed.

Detailed statistics for the three datasets used in our experiments are presented in Table 7.

A.2 Baseline Method Details

All experiments were repeated three times with different random seeds (42, 43, 44). We report the mean accuracy across the three runs. The standard deviations were consistently below 1.2%, indicating stable and reliable results. We compare

Dataset / Subset	Train	Test
GSM8K	7,473	1,319
MATH	7,500	5,000
NuminaMath-CoT	8,000	2,000
cn_k12	2,594	624
synthetic_math	1,566	388
orca_math	1,427	357
olympiads	1,375	377
synthetic_amc	584	139
aops_forum	278	73
amc_aime	176	42

Table 7: Dataset information with train and test set sizes.

PAD-Curriculum with representative baselines categorized into four groups:

1. Original The original pretrained model without any fine-tuning, used as a lower bound reference to measure the effectiveness of all fine-tuning methods.

2. Standard Reinforcement Learning Baselines

- **GRPO (Group Relative Policy Optimization)** (Shao et al., 2024): Standard GRPO training without curriculum learning. The model samples k trajectories for each problem, and advantage is computed via within-group standardization. This baseline serves to isolate the contribution of curriculum learning, as it shares the same sampling strategy and RL objective with PAD-Curriculum but lacks policy-adaptive curriculum ordering.
- **OPD (On-Policy Distillation)** (Agarwal et al., 2024): A post-training technique where a student model learns from its own generated trajectories by minimizing the reverse KL divergence relative to a distillation teacher's output distribution. Unlike GRPO which uses within-group advantage normalization, OPD directly aligns the student's distribution with the teacher distribution through reverse KL minimization. We implement OPD following the standard OPD framework.

3. Rejection sampling Fine-Tuning

- **RFT:** A baseline that uses the same self-generated trajectory filtering mechanism as PAD-Curriculum but replaces reinforcement learning with supervised fine-tuning. Specifically, for each training epoch, the current model samples $k = 10$ trajectories per problem, retains those with correct answers, and directly fine-tunes the model on these trajectories using cross-entropy loss. This baseline evaluates whether the policy-adaptive curriculum ordering and RL optimization provide advantages over simple imitation of self-generated correct trajectories.

4. External Curriculum Baseline

- **AdaBack** (Amani et al., 2025): A curriculum method that controls task difficulty by providing varying lengths of correct reasoning prefixes. Following the original implementation, we assume access to ground-truth chain-of-thought trajectories in the training data. Shorter prefixes indicate harder problems. This baseline represents existing curriculum learning approaches that depend on external process supervision (i.e., annotated reasoning steps), unlike our method which relies only on final answer verification.

Curriculum scorer. PAD-Curriculum uses family-matched stronger frozen curriculum scorers for discrepancy-guided difficulty estimation. For Llama-3.1-8B-Instruct, we use Meta-Llama-3.1-70B-Instruct as the scorer. For Qwen3-8B-Instruct and Qwen3-14B-Instruct, we use Qwen3-72B-Instruct. This family-matched design ensures that token-level Scorer→Student KL discrepancy is computed over a compatible tokenization space. The scorer is used only during curriculum construction and periodic re-ranking. It provides token distributions for computing the normalized discrepancy score on student-generated correct trajectories, but it is not updated, not used to generate training targets, and not used as the KL-regularization reference policy in the GRPO objective. For a fair comparison, OPD uses the same family-matched model as its distillation teacher, whereas PAD-Curriculum uses it only as a curriculum scorer.

A.3 Hyperparameter Details

Table 8 summarizes all hyperparameters used in our experiments.

B Effect of KL Direction

We further study the effect of KL direction in discrepancy-guided difficulty estimation. PAD-Curriculum uses the Scorer→Student KL discrepancy, $D_{\mathrm{KL}}(\pi_{\mathrm{score}} \| \pi_\theta)$, to measure how much prob-

Parameter	Value
GPUs	8 $\times A100$
Global batch size	128
Learning rate	1e-6
Training epochs	20
Group size G	10
Max response length	2048
Re-ranking interval K	40
Numerical stability ϵ	1e-8

Table 8: Summary of all hyperparameters.

ability mass assigned by the stronger frozen curriculum scorer is missing under the current student policy along the student's own trajectory. As an alternative, we evaluate a Student→Scorer KL variant that replaces this score with $D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{score}})$, while keeping the remaining curriculum construction procedure unchanged.

Method	GSM8K	MATH
Original	83.71	59.18
Student→Scorer KL	92.93	66.28
PAD-Curriculum	**95.83**	**70.41**

Table 9: Effect of KL direction on Qwen3-8B-Instruct. The Student→Scorer variant uses $D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{score}})$ for discrepancy-guided ordering, whereas PAD-Curriculum uses the Scorer→Student discrepancy $D_{\mathrm{KL}}(\pi_{\mathrm{score}} \| \pi_\theta)$.

As shown in Table 9, the Student→Scorer variant improves over the original model but performs worse than PAD-Curriculum by a clear margin. This suggests that the direction of the discrepancy measure matters. The Scorer→Student KL used in PAD-Curriculum better captures probability mass that the stronger scorer considers important but the student underestimates on student-generated trajectories. In contrast, the Student→Scorer direction can be dominated by tokens that the student already assigns high probability to, making it less effective for identifying solvable but still informative examples. These results support our choice of Scorer→Student KL for curriculum scoring.

C Effect of Re-Ranking Frequency

We also investigate the impact of difficulty re-ranking frequency on model performance. Recall that PAD-Curriculum dynamically updates the difficulty ordering based on the model's evolving student–scorer discrepancy scores. The re-ranking interval determines how often this update occurs. Table 10 reports the MATH accuracy under different re-ranking intervals.

Re-Ranking Interval	MATH Accuracy
No Re-Ranking	69.73
Every 20 steps	70.56
Every 40 steps	70.41
Every 60 steps	70.19
Every 80 steps	70.24

Table 10: Effect of re-ranking interval on MATH accuracy.

The results show that dynamic re-ranking consistently outperforms the static curriculum without re-ranking, providing evidence that updating the difficulty ordering during training is beneficial. Among the different intervals, updating every 20 steps yields the best performance, while longer intervals lead to slightly lower accuracy. However, all intervals with re-ranking achieve comparable results, suggesting that the method is not overly sensitive to the exact re-ranking frequency as long as updates are performed regularly. In our main experiments, we adopt a re-ranking interval of every 40 steps as a balanced choice between adaptation frequency and computational overhead.

D Training Dynamics and Curriculum Behavior

To better understand how curriculum learning affects optimization across reasoning difficulty levels, we further analyze the training dynamics of GRPO and PAD-Curriculum on the MATH benchmark (Qwen3-8B-Instruct). In particular, we track the accuracy of different MATH difficulty tiers throughout training.

D.1 Difficulty-wise Learning Dynamics

Figure 4 presents the training dynamics across MATH difficulty tiers from L1 (easiest) to L5 (hardest). We evaluate model performance every four training epochs for both standard GRPO and PAD-Curriculum. Several observations emerge from the training dynamics.

First, standard GRPO improves rapidly on easier difficulty tiers during early training stages, but progress on harder tiers becomes increasingly limited as training proceeds. This suggests that standard outcome-supervised RL tends to focus on

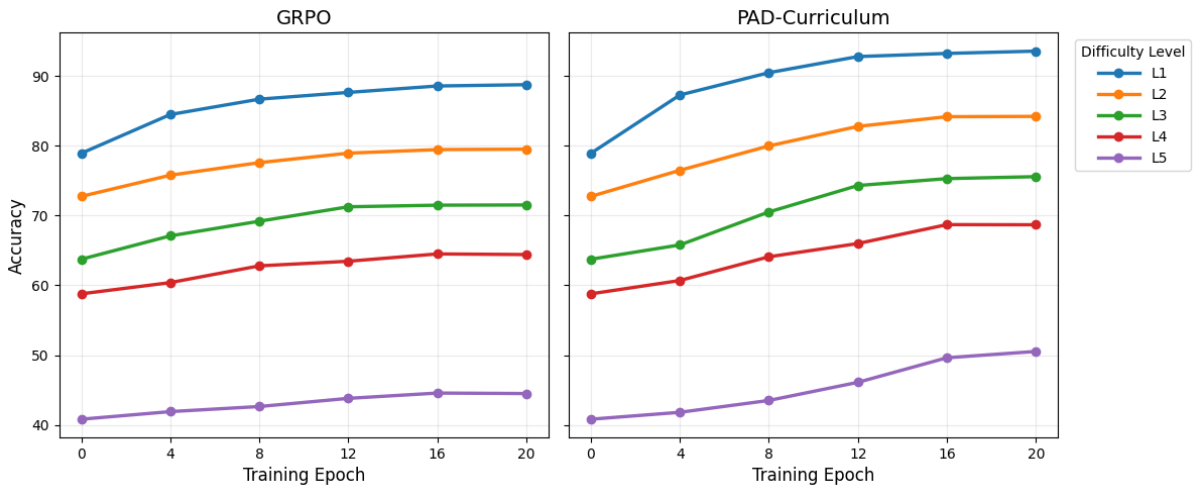

Figure 4: Difficulty-wise learning dynamics during training on the MATH benchmark. Standard GRPO rapidly improves on easier difficulty tiers but exhibits slow progress on harder tiers. In contrast, PAD-Curriculum progressively unlocks harder reasoning problems during later training stages, especially on L4–L5.

problems that already provide relatively stable reward signals, while optimization on more difficult problems remains challenging.

In contrast, PAD-Curriculum exhibits a different progression pattern across difficulty levels. Easier tiers improve first, while harder tiers begin to accelerate more noticeably during later stages of training. This delayed-but-progressive improvement is especially visible on higher-difficulty tiers, where PAD-Curriculum continues to improve even after standard GRPO begins to plateau.

Second, the performance gap between PAD-Curriculum and GRPO becomes progressively larger as difficulty increases. On easier tiers, both methods achieve relatively strong performance after sufficient training. However, on harder tiers, PAD-Curriculum maintains more stable improvement throughout optimization, suggesting that policy-adaptive curriculum scheduling is particularly beneficial for challenging multi-step reasoning problems.

Overall, these dynamics are consistent with the central hypothesis of PAD-Curriculum: curriculum difficulty should be defined relative to the evolving student policy rather than by static external difficulty labels. By prioritizing problems within the model's current solvable region and progressively introducing more challenging solvable examples, PAD-Curriculum gradually expands the model's solvable frontier throughout training.

D.2 Discussion: Curriculum Frontier Expansion

The observed dynamics suggest that PAD-Curriculum does not merely improve final accuracy, but also changes the optimization trajectory of reinforcement learning itself.

Under standard GRPO, optimization tends to concentrate on easier problems that already provide relatively reliable reward signals, while harder problems remain difficult to optimize because of sparse outcome supervision. As a result, learning on higher difficulty tiers often slows substantially during later stages of training.

In contrast, PAD-Curriculum dynamically adjusts the training distribution according to the evolving capability of the current student policy. Problems that are currently solvable but still exhibit substantial discrepancy from the curriculum scorer are gradually emphasized during later training stages. This creates a curriculum-like progression in which easier reasoning behaviors are consolidated first before progressively extending toward harder reasoning problems.

The observed training dynamics therefore provide additional evidence that policy-adaptive curriculum learning can progressively expand the model's solvable reasoning frontier over time.

E Theoretical Analysis of Re-ranking Cost

This section analyzes the additional forward-token cost introduced by periodic re-ranking. Since the

curriculum scorer is larger than the student model, this analysis should be interpreted as a theoretical cost decomposition rather than a direct wall-clock measurement. Actual wall-clock overhead also depends on model size, batching strategy, decoding efficiency, and hardware configuration.

Theoretical analysis. Each re-ranking requires greedy decoding and KL computation on the remaining problems. For a problem answered correctly, we need both student and scorer forwards; for incorrect problems, only student forwards. Let:

- N: total number of training problems
- $N_{\text{remain}}^{(r)}$: number of remaining problems at the r-th re-ranking
- L: average response length (tokens per generated response)
- $\alpha^{(r)}$: correctness rate of the student policy at the r-th re-ranking
- T: total number of training steps
- B: batch size
- G: group size (number of responses sampled per problem in GRPO)
- K: re-ranking interval (steps)
- $R = \lfloor T/K \rfloor$: total number of re-rankings

The total forward tokens required by baseline GRPO is:

$$\text{Base}_{\text{forward}} = T \times B \times G \times L.$$

For a single re-ranking, the number of forward tokens is:

$$\text{Forward}_{\text{rerank}}^{(r)} = N_{\text{remain}}^{(r)} \times L \times (1 + \alpha^{(r)}).$$

The first term $N_{\text{remain}}^{(r)} \times L$ accounts for greedy decoding (student forward only). The second term $N_{\text{remain}}^{(r)} \times \alpha^{(r)} \times L$ accounts for the additional curriculum-scorer forward required only for correctly answered problems.

The total re-ranking overhead across training is:

$$\text{Extra}_{\text{forward}} = L \sum_{r=1}^{R} N_{\text{remain}}^{(r)} \times (1 + \alpha^{(r)}).$$

Thus, the relative computational overhead is:

$$\text{Overhead} = \frac{\text{Extra}_{\text{forward}}}{\text{Base}_{\text{forward}} + \text{Extra}_{\text{forward}}}$$

$$= \frac{L \sum_{r=1}^{R} N_{\text{remain}}^{(r)} (1 + \alpha^{(r)})}{TBGL + L \sum_{r=1}^{R} N_{\text{remain}}^{(r)} (1 + \alpha^{(r)})}.$$

Canceling L from numerator and denominator:

$$\boxed{\text{Overhead} = \frac{\sum_{r=1}^{R} N_{\text{remain}}^{(r)} (1 + \alpha^{(r)})}{TBG + \sum_{r=1}^{R} N_{\text{remain}}^{(r)} (1 + \alpha^{(r)})}}.$$

Special case: uniform approximation. If we approximate $N_{\text{remain}}^{(r)} \approx N/2$ (average remaining problems) and $\alpha^{(r)} \approx \bar{\alpha}$ (average correctness rate) for all r, the overhead simplifies to:

$$\text{Overhead} \approx \frac{R \cdot \frac{N}{2} \cdot (1 + \bar{\alpha})}{TBG + R \cdot \frac{N}{2} \cdot (1 + \bar{\alpha})}.$$

Since $T = \frac{N \cdot \text{epochs}}{B}$, this can be further expressed in terms of training epochs E and the number of re-rankings per epoch R/E.

Key observations.

- The overhead is independent of response length L, as it cancels out in the ratio.
- Overhead decreases with larger batch size B or group size G, as baseline cost grows linearly while re-ranking cost depends only on N (dataset size), not on B or G.
- Overhead increases with more frequent re-ranking (larger R) and higher correctness rate α, as more problems require the additional scorer forward.

F Case Study: Why Token-Level Distillation Can Be Misleading

Token-level distillation methods (e.g., OPD (Agarwal et al., 2024)) train the student to match the teacher's token distribution at each position. While this can improve task performance, it may correct local token preferences without necessarily improving the underlying reasoning procedure. In contrast, PAD-Curriculum uses the curriculum scorer only for curriculum scoring. It does not train the student to match scorer-model token distributions; the student learns from its own complete trajectories via GRPO rewards.

Problem. For two real values of n, the equation $9x^2 + nx + 36 = 0$ has exactly one solution in x. What is the positive value of n?

Ground Truth: $n = 36$

Figure 5: Example problem used in the case study.

F.1 Problem Setting

We select a Level 4 algebra problem from the MATH test set:

The student model (Qwen3-8B) applies the discriminant method. The critical step is computing $4ac = 4 \times 9 \times 36$. The student erroneously computes this as 324 instead of 1296, leading to $n = 18$ instead of $n = 36$:

$$\Delta = n^2 - 4(9)(36) = n^2 - \mathbf{324}$$
$$n^2 = 324 \quad \Rightarrow \quad n = \pm 18 \;\Rightarrow\; \boxed{18}$$

The student's reasoning is self-consistent; only the arithmetic is wrong.

F.2 The Problem with Token-Level Distillation

Token-level distillation shifts the student's token distribution at specific positions toward the teacher's distribution. At the error position, the teacher assigns high probability to "1296", so the student may become more likely to output it. However, correcting a local token preference does not necessarily ensure that the underlying arithmetic procedure has changed. This leads to two issues:

1. **Unreliable inference.** The student may produce correct or incorrect answers depending on whether the shifted distribution dominates.

2. **Limited reasoning correction.** The student may correct a local token preference without necessarily correcting the underlying arithmetic procedure.

F.3 How PAD-Curriculum Avoids This

Our method uses student–scorer KL discrepancy purely as a *scheduling signal*, not as a training target. The student always learns from its own complete trajectories via GRPO rewards. For this problem:

- If the student currently fails to solve it, the problem is deferred and re-evaluated in a later curriculum stage.
- When the problem re-enters training, the student generates its own trajectories and receives outcome rewards from the verifier.
- Any improvement therefore comes from reward-driven updates on sampled trajectories rather than direct imitation of scorer-model tokens.

This case study illustrates the distinction between using a scoring model as a token-level training target and using a frozen curriculum scorer only as a scheduling signal.

For clarity, Algorithm 1 abstracts the GRPO update and focuses on the curriculum construction procedure. In our implementation, the policy is optimized with the standard GRPO objective using group-normalized rewards, policy-ratio clipping, and KL regularization.

Algorithm 1 PAD-Curriculum: Policy-Adaptive Discrepancy Curriculum for GRPO

Require:
Problem set $\mathcal{Q}$, student policy π_θ, frozen curriculum scorer π_{score}
Verifier $\mathcal{V}$, group size G, re-ranking interval K, batch size B

function MAINTRAIN($\mathcal{Q}, \pi_\theta, \pi_{\text{score}}, \mathcal{V}, G, K, B$)
  Initialize update counter $t \leftarrow 0$
  $\mathcal{Q}_{\text{sorted}} \leftarrow$ EVALUATEANDSORT($\mathcal{Q}, \pi_\theta, \pi_{\text{score}}, \mathcal{V}$) ▷ Initial curriculum
  while not converged **and** $\mathcal{Q}_{\text{sorted}}$ is not empty **do**
    Sample mini-batch $\mathcal{B} \subset \mathcal{Q}_{\text{sorted}}$ of size B ▷ From front of queue
    for each problem $q_i \in \mathcal{B}$ **do**
      Sample G responses: $\{o_{i,g}\}_{g=1}^{G} \sim \pi_\theta(\cdot \mid q_i)$
      Compute rewards: $\{r_{i,g}\}_{g=1}^{G}$ via $\mathcal{V}$ ▷ $r_{i,g} \in \{0, 1\}$
    end for
    ▷ Compute advantages via within-group normalization
    for each problem $q_i \in \mathcal{B}$ **do**
      $\mu_i \leftarrow \frac{1}{G} \sum_{g=1}^{G} r_{i,g}$
      $\sigma_i \leftarrow \sqrt{\frac{1}{G} \sum_{g=1}^{G} (r_{i,g} - \mu_i)^2 + \epsilon}$
      for $g = 1$ to G **do**
        $A_{i,g} \leftarrow (r_{i,g} - \mu_i)/\sigma_i$
      end for
    end for
    ▷ GRPO policy gradient update
    $\mathcal{L}_{\text{GRPO}}(\theta) \leftarrow -\frac{1}{|\mathcal{B}|} \sum_{q_i \in \mathcal{B}} \frac{1}{G} \sum_{g=1}^{G} A_{i,g} \cdot \log \pi_\theta(o_{i,g} \mid q_i)$
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{GRPO}}(\theta)$
    $t \leftarrow t + 1$
    if $t \bmod K = 0$ **then**
      $\mathcal{Q}_{\text{remain}} \leftarrow$ remaining untrained problems in $\mathcal{Q}_{\text{sorted}}$
      $\mathcal{Q}_{\text{sorted}} \leftarrow$ EVALUATEANDSORT($\mathcal{Q}_{\text{remain}}, \pi_\theta, \pi_{\text{score}}, \mathcal{V}$)
    end if
  end while
  return π_θ
end function

Algorithm 2 EvaluateAndSort: Discrepancy-Guided Difficulty Estimation and Curriculum Construction

function EVALUATEANDSORT($\mathcal{Q}, \pi_\theta, \pi_{\text{score}}, \mathcal{V}$)
  Initialize $\mathcal{S}_{\text{correct}} \leftarrow \emptyset$, $\mathcal{S}_{\text{incorrect}} \leftarrow \emptyset$
  for each problem $q_i \in \mathcal{Q}$ **do**
    Generate response $s_i \leftarrow \text{GreedyDecode}(\pi_\theta, q_i)$ ▷ Curriculum-scoring rollout
    $c_i \leftarrow \mathcal{V}(q_i, s_i)$ ▷ Correctness check: $c_i \in \{0, 1\}$
    if $c_i = 0$ **then**
      $\mathcal{S}_{\text{incorrect}} \leftarrow \mathcal{S}_{\text{incorrect}} \cup \{q_i\}$
    else
      ▷ Compute discrepancy-guided difficulty for correct responses
      $\text{KL}_i \leftarrow 0$, $H_{\text{score},i} \leftarrow 0$
      for $k = 1$ to $|s_i|$ **do**
        $x_{i,k} \leftarrow (q_i, s_{i,<k})$ ▷ Prefix up to position $k - 1$
        $p_\theta \leftarrow \pi_\theta(\cdot \mid x_{i,k})$
        $p_{\text{score}} \leftarrow \pi_{\text{score}}(\cdot \mid x_{i,k})$
        $\text{KL}_i \leftarrow \text{KL}_i + D_{\text{KL}}(p_{\text{score}} \parallel p_\theta)$ ▷ Scorer→Student KL
        $H_{\text{score},i} \leftarrow H_{\text{score},i} + H(p_{\text{score}})$ ▷ Scorer entropy
      end for
      $\overline{\text{KL}}_i \leftarrow \text{KL}_i / |s_i|$
      $\overline{H}_{\text{score},i} \leftarrow H_{\text{score},i} / |s_i|$
      $D_i \leftarrow \overline{\text{KL}}_i / (\overline{H}_{\text{score},i} + \epsilon)$ ▷ Normalized discrepancy-guided difficulty
      $\mathcal{S}_{\text{correct}} \leftarrow \mathcal{S}_{\text{correct}} \cup \{(q_i, D_i)\}$
    end if
  end for
  Sort $\mathcal{S}_{\text{correct}}$ by D_i in ascending order ▷ Easiest first
  $\mathcal{Q}_{\text{sorted}} \leftarrow \mathcal{S}_{\text{correct}} \cup \mathcal{S}_{\text{incorrect}}$ ▷ Incorrect appended at end
  return $\mathcal{Q}_{\text{sorted}}$
end function