Process Rewards for Outcome-Guided Reasoning Steps in LLM Reinforcement Learning

1st Mohammad Rezaei Ravari
Faculty of Computer Science
Technical University of Dresden
Dresden, Germany
mohammad.rezaei@tu-dresden.de

2nd Jens Lehmann*
Amazon Science
Technical University of Dresden
Dresden, Germany
jens.lehmann@tu-dresden.de

3rd Sahar Vahdati
TIB Leibniz Information Centre
Leibniz University of Hanover
Hanover, Germany
sahar.vahdati@tib.eu

***Abstract*—Mathematical reasoning in large language models has improved substantially with reinforcement learning using verifiable rewards, where final answers can be checked automatically and converted into reliable training signals. Most such pipelines optimize outcome correctness only, which yields sparse feedback for long, multi-step solutions and offers limited guidance on intermediate reasoning errors. Recent work therefore introduces process reward models (PRMs) to score intermediate steps and provide denser supervision. In practice, PRM scores are often imperfectly aligned with final correctness and can reward locally fluent reasoning that still ends in an incorrect answer. When optimized as absolute rewards, such signals can amplify fluent failure modes, induce reward hacking, and destabilize policy updates. Existing PRM-based methods improve PRM quality, filter trajectories, or modify reward aggregation, but they do not directly constrain how process rewards interact with outcome correctness during optimization. We propose PROGRS, a framework that leverages PRMs while keeping outcome correctness dominant. PROGRS treats process rewards as *relative preferences within outcome groups* rather than absolute targets. We introduce *outcome-conditioned centering*, which shifts PRM scores of incorrect trajectories to have zero mean within each prompt group. This removes systematic bias while preserving informative relative rankings. To stabilize process signals, PROGRS combines a frozen *quantile-regression PRM* with a *multi-scale coherence evaluator* that penalizes short-window confidence volatility. We integrate the resulting centered process bonus into *Group Relative Policy Optimization (GRPO)* without auxiliary objectives or additional trainable components. Across MATH-500, AMC, AIME, MinervaMath, and OlympiadBench, PROGRS consistently improves Pass@1 over outcome-only baselines (e.g., 74.9% vs. 69.7% on MATH-500; 59.0% vs. 52.0% on AMC-2023) and achieves stronger performance with fewer rollouts. These results show that outcome-conditioned centering enables safe and effective use of process rewards for mathematical reasoning.**

***Index Terms*—Reinforcement Learning with Verifiable Rewards, Mathematical Reasoning, Process Reward Models.**

I. Introduction

Large language models (LLMs) achieve strong performance on many natural language tasks, but complex, multi-step mathematical reasoning remains challenging. Models often produce hallucinated steps, brittle chains of thought, or overconfident yet invalid solutions [1], [2]. Mathematical reasoning requires precise logical dependency tracking, structured decomposition, and verifiable solution construction.

A widely adopted direction is reinforcement learning (RL) with LLMs for mathematical reasoning [2], [3]. RL with verifiable rewards (RLVR) leverages automatically checkable correctness signals to optimize final-answer accuracy [1], [4]. Most RLVR pipelines still rely on sparse outcome-only reward models, which provide limited guidance for long reasoning trajectories. Process reward models (PRMs) were introduced to provide denser supervision. PRMs assign scores to intermediate reasoning steps, so optimization can reflect trajectory quality rather than only the final outcome [1], [5]. In practice, PRMs are often miscalibrated and can assign high scores to locally coherent reasoning that still ends in an incorrect answer. Using PRM scores naively as absolute rewards can induce reward hacking and destabilize training [6]–[8].

Recent methods mitigate these issues through generative judges with thought-level evaluation and capability-adaptive rewards [9], theoretical derivations from entropy-regularized RL [10], or sample filtering based on score consistency [11]. These strategies improve evaluation, grounding, or data reliability. A key gap remains. Existing approaches do not explicitly control how process rewards interact with outcome supervision during optimization.

We propose **Process-Reward Outcome-Guided Reasoning Steps (PROGRS)** [1], a framework integrating process-level guidance into RLVR while preserving outcome dominance. In PROGRS, process rewards act as *relative preferences* within groups defined by outcome quality. To enforce this principle, we introduce *outcome-conditioned centering*, which removes systematic positive bias from PRM scores on incorrect trajectories while retaining informative relative rankings. PROGRS constructs stable process signals using three components. First, it uses calibrated step-level PRMs [12]. Second, it applies a hierarchical multi-scale coherence evaluator that aggregates PRM scores over short reasoning windows and penalizes abrupt confidence fluctuations. Third, it forms trajectory-level process scores that are centered on final outcomes and combined with outcome-based advantages during optimization. PROGRS introduces no new trainable components and operates within standard Group Relative Policy Optimization (GRPO) [3], ensuring correctness remains the dominant signal.

*This work was done outside of Amazon.

[1]https://github.com/NIMI-research/PROGRS-Process-Reward

II. Related Work

We review relevant strands of prior work: RL for mathematical reasoning with verifiable rewards, process reward models, and efforts to reconcile process and outcome signals in RLVR.

RL for Mathematical Reasoning with Verifiable Rewards. RLVR optimizes policies using automatically checkable outcome rewards, typically binary correctness for final answers [2], [3], [5]. Because outcome feedback is sparse, stable policy optimization is critical. Classic policy-gradient methods (e.g., REINFORCE) and variance-reduction techniques (e.g., GAE) underpin modern RL pipelines, while trust-region style objectives (TRPO, PPO) stabilize updates and are widely used in RLHF settings [13]–[19]. For reasoning tasks, GRPO [2], [3] improves stability by normalizing advantages within groups of sampled solutions per prompt. DAPO [20] further stabilizes updates via asymmetric clipping. We build on GRPO-style training and use asymmetric clipping as an orthogonal stabilization method. Our main contribution is a modified advantage that controls how process signals interact with outcome correctness.

Process Reward Models (Supervision, Calibration and Failure Models). PRMs provide denser feedback than outcome-only reward models (ORMs) [1], [5]. Early PRMs relied on human annotations [1], while later works reduced labeling cost through automated rollouts [5], [21], adaptive search labeling [22], and LLM-based or hybrid pipelines [23]. However, due to miscalibration, PRMs may assign high scores to trajectories that are coherent locally yet incorrect globally. Optimizing such scores as absolute rewards can induce reward hacking and amplify fluent failure modes [6]–[8]. Several approaches mitigate this issue by down-weighting rewards or aggregating step signals independently, but they do not constrain process feedback with outcome supervision during gradient computation. Recent calibrated PRMs [12] estimate step-level success probabilities and uncertainty (e.g., via quantile heads), enabling uncertainty-aware selection at inference time. In our setting, a calibrated PRM is a fixed training-time evaluator with a focus on how to use imperfect process scores without overriding outcome correctness during optimization.

Reconciling Process and Outcome Signals in RLVR.

Recent methods integrate process supervision into RLVR in different ways. TP-GRPO [9] addresses step segmentation ambiguity with thought-level evaluation and introduces capability-adaptive rewards to reduce repeated-step reward exploitation. PRL [10] derives process rewards from entropy-regularized RL objectives, avoiding expensive search-based labeling. PROF [11] reduces PRM–ORM conflicts by filtering trajectories based on consistency between process and outcome evaluations. These approaches improve PRM quality, grounding, or data reliability, and are complementary to our goal. Our focus is the missing *semantic constraint* in PRM-based RLVR pipelines. Even with better PRMs or filtering, positive process bonuses on incorrect trajectories can distort optimization. We address this by centering within outcome groups only for incorrect trajectories, preserving outcome dominance.

III. Proposed Method

PROGRS is an advantage construction method for reinforcement learning with verifiable rewards in mathematical reasoning. It operates over prompt-level groups of sampled solutions and combines an outcome-based advantage with a *centered* PRM-derived process bonus. This makes process scores act as relative preferences among incorrect trajectories, preventing fluent but incorrect reasoning from being systematically favored while preserving outcome correctness as the dominant signal.

A. Advantage Construction

Group sampling and signals. For each prompt q, we sample a group of K trajectories $\{o^{(i)}\}_{i=1}^{K}$ from the policy. Each trajectory receives: (i) an outcome reward $r^{(i)}_{\text{outcome}} \in \{0,1\}$ indicating final-answer correctness, and (ii) a trajectory-level process score $S^{(i)}_{\text{PRM}}$ computed in Section III-B.

1) Outcome-Based Advantage: We compute a normalized outcome-based advantage within each prompt group:

$$A^{(i)}_{\text{outcome}} = \begin{cases} \dfrac{r^{(i)}_{\text{outcome}} - \bar{r}_{\text{outcome}}}{\sigma_{\text{outcome}} + \epsilon}, & \sigma_{\text{outcome}} \geq \epsilon, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where $\bar{r}_{\text{outcome}}$ and σ_{outcome} are computed over the K samples within the same prompt group, and $\epsilon > 0$ is a small constant.

2) Outcome-Conditioned Centering: PRMs can assign high process scores to incorrect but locally fluent solutions. If used as an absolute bonus, this can systematically reinforce incorrect trajectories. PROGRS removes this offset by centering process scores *within the incorrect subset*.

Let $\mathcal{I} = \{i \mid r^{(i)}_{\text{outcome}} = 0\}$ denote the indices of incorrect samples in the group and define

$$\mu^{\text{incorrect}}_{\text{PRM}} = \begin{cases} \dfrac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} S^{(i)}_{\text{PRM}}, & |\mathcal{I}| > 0, \\ 0, & |\mathcal{I}| = 0. \end{cases}$$

We then center:

$$\tilde{S}^{(i)}_{\text{PRM}} = \begin{cases} S^{(i)}_{\text{PRM}}, & r^{(i)}_{\text{outcome}} = 1, \\ S^{(i)}_{\text{PRM}} - \mu^{\text{incorrect}}_{\text{PRM}}, & r^{(i)}_{\text{outcome}} = 0. \end{cases} \tag{2}$$

We apply centering only to incorrect trajectories. Applying it to correct trajectories would reduce separation between correct and incorrect solutions, weakening outcome dominance, the primary optimization objective in RLVR. In preliminary experiments, this reduced performance. By construction, the mean of $\tilde{S}^{(i)}_{\text{PRM}}$ over $i \in \mathcal{I}$ is zero, so incorrect samples receive no systematic process bonus.

3) Final Advantage Integration: We combine outcome and centered process signals additively:

$$A^{(i)}_{\text{final}} = A^{(i)}_{\text{outcome}} + \lambda_{\text{PRM}} \, \tilde{S}^{(i)}_{\text{PRM}}, \tag{3}$$

where λ_{PRM} controls the strength of the process guidance. When all K samples in a prompt group share the same outcome, $A^{(i)}_{\text{outcome}} = 0$ and learning is driven by within-group preferences induced by $\tilde{S}^{(i)}_{\text{PRM}}$.

B. Computing the Process Score S_{PRM}

We compute S_{PRM} from step-level PRM scores and a windowed coherence penalty.

1) Frozen Quantile-Regression PRM for Step Scoring: We use a frozen quantile-regression Process Reward Model (PRM) from [12] as an external evaluator, using the released checkpoint without additional fine-tuning.

Step delimitation and representation extraction. We represent each reasoning trajectory as a sequence of T textual steps $s_{1:T}$ and insert a dedicated delimiter token <extra_0> immediately after each step. We define the full token sequence

$$x = \text{concat}\big(q,\, s_1,\, \texttt{<extra\_0>},\, \ldots,\, s_T,\, \texttt{<extra\_0>}\big).$$

Running the decoder-only PRM with a causal attention mask yields hidden states at each delimiter position. Let h_t denote the hidden state at the delimiter immediately following step s_t. Because the model is evaluated under a causal mask, h_t depends only on tokens up to that delimiter, i.e., only on the prefix $(q, s_{1:t})$.

Step score. For each step representation h_t, the quantile heads predict success probability estimates $q_\tau(h_t)$ for $\tau \in \{0.1, 0.5, 0.9\}$. We use the median prediction as the step-level score: $r_{\text{fine}}(s_t) = q_{0.5}(h_t)$, which serves as a robust point estimate of step-level correctness.

Implementation note (prefix-only consistency). Conceptually, step t can be scored by a prefix-only PRM call on $x_t = \text{concat}(q, s_1, \texttt{<extra\_0>}, \ldots, s_t, \texttt{<extra\_0>})$. In practice, extracting delimiter states from a single forward pass on x is equivalent under a causal mask.

2) Windowed Coherence and Trajectory Aggregation: While step-level PRM scores capture local confidence, they do not reflect the stability of reasoning across neighboring steps. Across all benchmarks, reasoning trajectories are relatively short, typically ranging between 12 and 18 steps on average. This consistent scale motivates the use of a small window size ($w = 3$), which captures local coherence while maintaining sufficient granularity. In contrast, larger window sizes would reduce each trajectory to only a few segments, weakening the ability to detect local instability in reasoning. We therefore compute a coherence-modulated process score using contiguous windows of step scores. We use non-overlapping windows to reduce correlation between adjacent estimates and to avoid redundant smoothing effects observed with overlapping windows. In practice, overlapping windows introduced highly correlated scores that provided limited additional signal while increasing computational cost.

Windowed variance analysis. Given $\{r_{\text{fine}}(s_t)\}_{t=1}^{T}$, we partition the trajectory into contiguous, non-overlapping windows of nominal size w. Let $N_w = \lceil T/w \rceil$ and define

$$W_j = \{(j-1)w + 1, \ldots, \min(jw, T)\}, \quad j \in \{1, \ldots, N_w\}.$$

For each window j, we compute the local mean and standard deviation, under setting of $\sigma_j = 0$ when $|W_j| = 1$:

$$\mu_j = \frac{1}{|W_j|} \sum_{t \in W_j} r_{\text{fine}}(s_t), \quad \sigma_j = \sqrt{\frac{1}{|W_j|} \sum_{t \in W_j} \big(r_{\text{fine}}(s_t) - \mu_j\big)^2}. \tag{4}$$

Coherence-modulated window score. To translate local variability into a coherence signal, we define

$$r_{\text{coh},j}(w) = \mu_j \cdot \exp\left(-\lambda_{\text{var}} \frac{\sigma_j}{\mu_j + \epsilon}\right), \tag{5}$$

where $\lambda_{\text{var}} > 0$ controls the penalty strength and $\epsilon > 0$ ensures numerical stability. This yields a multiplicative downweighting of high-variance windows without imposing hard thresholds.

Hierarchical aggregation with coherence blending. To balance raw step quality with local stability, we compute a blended window score:

$$R_j(w) = \alpha_{\text{coh}} \cdot r_{\text{coh},j}(w) + (1 - \alpha_{\text{coh}}) \cdot \mu_j, \tag{6}$$

and aggregate across windows to obtain a trajectory score at scale w:

$$S_{\text{PRM}}(w) = \frac{1}{N_w} \sum_{j=1}^{N_w} R_j(w).$$

Multi-scale (optional) form. More generally, one can average across window sizes $\mathcal{W}$:

$$S_{\text{PRM}} = \frac{1}{|\mathcal{W}|} \sum_{w \in \mathcal{W}} S_{\text{PRM}}(w).$$

In our main experiments we use a single scale $w = 3$ (i.e., $\mathcal{W} = \{3\}$), with $\lambda_{\text{var}} = 2.0$ and $\alpha_{\text{coh}} = 0.6$.

C. Policy Optimization with GRPO and Asymmetric Clipping

We optimize with Group Relative Policy Optimization (GRPO) and asymmetric clipping following DAPO [20], using $A^{(i)}_{\text{final}}$ from Eq. (3):

$$\mathcal{L}_{\text{policy}} = -\frac{1}{B} \sum_{i=1}^{B} \min\Big(r_i A^{(i)}_{\text{final}}, \text{clip}_{\epsilon_l, \epsilon_h}(r_i)\, A^{(i)}_{\text{final}}\Big), \tag{7}$$

where B is the number of sampled trajectories in the minibatch (across prompts and samples) and $r_i = \pi_\theta(o^{(i)} \mid q)/\pi_{\text{ref}}(o^{(i)} \mid q)$ is the sequence-probability ratio between the current policy and a reference policy.

Heuristic though it is, this construction admits a control-variate interpretation, suggesting a variance-reduction effect.

D. Why Outcome-Conditioned Centering Improves Stability

Outcome-conditioned centering addresses a specific failure mode of PRM-based training: an incorrect trajectory can receive a high process score because its intermediate reasoning is locally coherent, even when the final answer is wrong. Consider a representative case in which the model solves the problem of maximizing $\frac{wx+xy+yz}{w^2+x^2+y^2+z^2}$ for positive real numbers w, x, y, z. The correct optimum is $\frac{1+\sqrt{5}}{4}$, attained

when $w = z$, $x = y$, and $\frac{w}{x} = \frac{\sqrt{5}-1}{2}$. However, the model instead assumes $w = x = y = z$ and predicts $\frac{3}{4}$. This solution is globally incorrect, but its reasoning is well structured and applies plausible inequalities, derives consistent equality conditions, and exhibits low local variance in step-level PRM scores. As a result, the raw trajectory-level process score remains high. If such scores are used directly, optimization may systematically favor fluent but incorrect trajectories.

Outcome-conditioned centering prevents this by converting the PRM signal on incorrect trajectories from an absolute reward into a relative preference. In the above example, the raw process score is $S_{\text{PRM}} = 0.736$, while the mean PRM score over incorrect samples in the same prompt group is 0.643. Centering therefore yields $\tilde{S}_{\text{PRM}} = S_{\text{PRM}} - \mu_{\text{PRM}}^{\text{incorrect}} = 0.093$, so the trajectory receives only a small positive bonus relative to other incorrect samples, rather than a large absolute reward. This preserves intra-group ranking among incorrect solutions while removing the systematic positive offset that would otherwise compete with outcome supervision.

Formally, let $\mathcal{I} = \{i \mid r_{\text{outcome}}^{(i)} = 0\}$ be the set of incorrect trajectories and $z_i := \nabla_\theta \log \pi_\theta(o^{(i)} \mid q)$ the score function. The PRM contribution to the gradient is proportional to $\sum_{i\in\mathcal{I}} S_{\text{PRM}}^{(i)} z_i$, while outcome-conditioned centering replaces $S_{\text{PRM}}^{(i)}$ by $S_{\text{PRM}}^{(i)} - \mu_{\text{PRM}}^{\text{incorrect}}$ and therefore subtracts the term $\lambda_{\text{PRM}}\mu_{\text{PRM}}^{\text{incorrect}} \sum_{i\in\mathcal{I}} z_i$. By the score-function identity, the expectation of $\sum_{i\in\mathcal{I}} z_i$ (conditioned on the prompt) is zero, so removing this rank-one common-mode component does not change the expected gradient but reduces its variance. In this sense, outcome-conditioned centering acts as an unbiased control variate: the PRM bonus within $\mathcal{I}$ has zero mean, and only relative deviations around this baseline influence the update.

Consequently, incorrect trajectories cannot receive a net PRM uplift at the group level, while correct trajectories remain separated by the outcome-based advantage. This keeps outcome supervision in control of inter-group preference learning and uses PRM scores primarily to refine intra-group ordering among incorrect samples.

Finally, the validity of the PRM signal depends on prefix-causal step scoring. We score step t using the hidden state at the delimiter immediately following s_t in a decoder-only PRM with a causal attention mask, so the score depends only on the prefix $(q, s_{1:t})$. In controlled tests, prefix-only evaluation and full-trajectory extraction produced nearly identical scores, and perturbing the suffix left the score unchanged up to numerical precision. This confirms that the process signal used by PROGRS reflects prefix-consistent reasoning quality rather than leakage from future tokens.

IV. Evaluations and Results

We evaluate PROGRS variants with 4 and 8 rollouts on six mathematical reasoning benchmarks, and against DAPO baselines using 8 and 16 rollouts. We analyze accuracy, sample efficiency, computational cost, ablations, and Pass@K.

A. Experimental Setup

Training data and evaluation benchmarks. Following prior findings that harder examples provide stronger learning signals in RL fine-tuning [24], we train all models on Level 5 problems from the Hendrycks MATH dataset [25]. A preliminary study over three 2,000-sample subsets confirmed that Level 5 training yields the best downstream performance on MATH-500 (73.7% Pass@1), outperforming Level 4 (73.01%) and Level 3 (72.39%). This setup emphasizes cross-difficulty generalization while remaining computationally feasible. We evaluate on MATH-500 [25], MinervaMath [26], OlympiadBench [27], AMC 2023, AIME 2024, and AIME 2025, covering both in-distribution and out-of-distribution mathematical reasoning.

Models and reward functions. We fine-tune Qwen2.5-Math-1.5B-base [23]. Process-level signals are obtained from a frozen Qwen2.5-Math-PRM-7B with quantile-regression heads from a recent work [12]. We shall note that we used them without additional fine-tuning. All performance gains therefore are the direct impact of PROGRS reward formulation. We compare against outcome-only RLVR baselines using DAPO [20] with group sizes $B = 8$ and $B = 16$, under identical optimization settings.

We report Pass@1 accuracy using greedy decoding (temperature $= 0.0$, max length $= 4096$). Results are averaged over ten random seeds. We also report the average number of generated tokens per problem as a measure of computational cost.

B. Performance Comparison

Table I reports Pass@1 accuracy (mean $\pm$ std over 10 runs) on six mathematical reasoning benchmarks spanning in-distribution evaluation (MATH-500), distribution-shifted settings (AMC/AIME), and harder problem sets (MinervaMath, OlympiadBench). Overall, PROGRS improves accuracy over outcome-only baselines across most benchmarks while using fewer training rollouts (PROGRS-4/8 vs. DAPO-8/16), indicating a more favorable accuracy–budget trade-off. Surprisingly, the results for AIME 24 in no-centering and PROGRS-8 are the same. However, PROGRS-8 achieves substantial token efficiency gains (1082 ± 15 vs. 1148 ± 22 tokens/problem) with improved numerical stability.

Distribution shift vs. in-distribution. The clearest gains under shift appear on AMC-2023, where PROGRS-8 outperforms DAPO-16 (59.0% vs. 52.0%). On AIME-2024/2025, absolute accuracies are low and variance is higher, but PROGRS-8 remains above DAPO-16. We interpret these as directional evidence rather than definitive improvements. In contrast, on the in-distribution MATH-500 benchmark, gaps are smaller but consistent (e.g., PROGRS-8 at 74.9% vs. DAPO-16 at 69.7%), and PROGRS-4 matches or exceeds larger-rollout baselines, supporting improved sample efficiency.

Hardness and stability. On MinervaMath, PROGRS yields sizable gains over DAPO (e.g., PROGRS-4 at 23.6% vs. DAPO-16 at 18.8%), consistent with the intuition that many problems admit structured partial progress even when the final answer is wrong, making within-group process preferences

TABLE I: Results and ablations across mathematical reasoning benchmarks. We report Pass@1 accuracy (upper block, in %) and average generated tokens per problem (lower block; lower is better), both as mean ± standard deviation over ten independent runs. DAPO-K denotes the outcome-only baseline trained with K rollouts per prompt; PROGRS-K denotes our method with the same rollout budget. Ablation rows remove individual components while keeping all other settings fixed: $\alpha_{\text{coh}} = 0$ disables the coherence penalty (centering retained), and *No Centering* removes outcome-conditioned centering (coherence retained).

Metric	Model / Setting	AIME 24	AIME 25	AMC 23	MATH-500	MinervaMath	OlympiadBench
Pass@1 (%)	DAPO-16	8.67 ± 4.00	8.00 ± 1.63	52.00 ± 3.67	69.66 ± 0.70	18.79 ± 1.47	32.03 ± 0.73
	DAPO-8	8.67 ± 1.63	$\mathbf{12.33 \pm 2.13}$	48.75 ± 5.62	68.40 ± 0.80	14.63 ± 0.59	31.34 ± 0.87
	PROGRS-4	10.67 ± 2.00	12.00 ± 1.63	50.25 ± 2.61	74.14 ± 0.98	$\mathbf{23.64 \pm 0.66}$	$\mathbf{35.74 \pm 0.89}$
	PROGRS-8	$\mathbf{12.00 \pm 3.06}$	10.67 ± 2.91	$\mathbf{59.00 \pm 5.27}$	$\mathbf{74.92 \pm 0.82}$	22.13 ± 0.49	35.06 ± 0.61
	Ablation: $\alpha_{\text{coh}} = 0$	9.33 ± 3.59	9.67 ± 2.33	50.50 ± 2.18	71.82 ± 1.07	20.18 ± 0.80	33.52 ± 0.58
	Ablation: No Centering	12.00 ± 3.06	8.33 ± 1.67	54.00 ± 3.74	67.78 ± 0.95	20.15 ± 0.75	30.77 ± 0.64
Avg tokens (↓)	DAPO-16	1202.22 ± 20.86	1128.88 ± 18.16	873.95 ± 12.23	$\mathbf{671.04 \pm 2.42}$	1110.73 ± 4.48	$\mathbf{910.65 \pm 4.83}$
	DAPO-8	1245.09 ± 22.52	1149.21 ± 15.21	$\mathbf{831.90 \pm 19.82}$	673.86 ± 2.21	1168.00 ± 3.03	944.48 ± 2.94
	PROGRS-4	1201.29 ± 14.99	$\mathbf{1065.38 \pm 12.05}$	950.66 ± 7.24	694.70 ± 2.62	1081.73 ± 5.61	990.17 ± 3.24
	PROGRS-8	$\mathbf{1082.42 \pm 15.47}$	1156.79 ± 15.73	918.64 ± 13.92	683.74 ± 2.67	$\mathbf{1043.08 \pm 4.99}$	958.38 ± 3.53
	Ablation: $\alpha_{\text{coh}} = 0$	1018.21 ± 30.21	1027.08 ± 11.72	858.91 ± 5.18	669.70 ± 3.20	1012.28 ± 3.37	934.80 ± 3.35
	Ablation: No Centering	1147.96 ± 21.58	1273.70 ± 17.86	911.98 ± 19.47	699.68 ± 2.10	1016.18 ± 3.70	975.54 ± 4.35

useful. On OlympiadBench, improvements persist but narrow (e.g., PROGRS-4 at 35.7% vs. DAPO-16 at 32.0%), suggesting diminishing returns from reward-structure refinements near model capacity. Across several benchmarks, PROGRS also exhibits lower run-to-run variance, consistent with outcome-conditioned centering reducing systematic PRM bias on incorrect trajectories and the coherence aggregation reducing sensitivity to volatile step scores.

C. Computational Efficiency

Our efficiency analysis focuses on rollout and token budgets and does not fully account for PRM inference costs. Since PRM scoring adds a forward pass at each reasoning step, training can be substantially more expensive than outcome-only RL. Thus, the reported gains reflect relative improvements under matched rollout settings, rather than absolute end-to-end compute efficiency. We evaluate the accuracy–compute trade-off along two axes: **training-time rollout budget** (group size K) and **inference-time generation length** (tokens per problem). Table I reports average tokens per problem, and Fig. 1 summarizes accuracy relative to compute budgets.

Rollout-budget efficiency. PROGRS is designed to improve the learning signal per rollout by using centered process preferences when outcome advantages are uninformative. Empirically, PROGRS-4 matches or exceeds DAPO baselines that use substantially larger rollout budgets (e.g., PROGRS-4 vs. DAPO-16 on MATH-500), and PROGRS-8 is generally competitive with or better than DAPO-16, indicating a more favorable accuracy–budget frontier.

Token usage. Token counts vary by benchmark (Table I). PROGRS does not uniformly increase generation length: it is shorter on some datasets and longer on others. Taken together with the Pass@1 improvements, this suggests gains are not explained solely by producing longer solutions, but by changes in solution quality induced by the training objective.

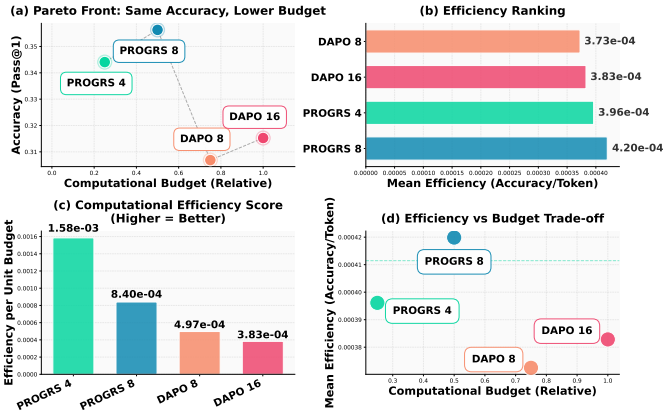

Fig. 1: Computational efficiency across models. (a) Pareto frontier: Accuracy vs. budget; PROGRS-4 matches DAPO baselines with $\sim 50\%$ budget, PROGRS-8 matches/exceeds DAPO-16. (b) Efficiency ranking: Accuracy per token; PROGRS-8 highest, PROGRS-4 next. (c) Efficiency score: Accuracy normalized by budget; PROGRS-4 best. (d) Efficiency-budget surface: PROGRS-4 optimal for $\leq 50\%$ budget, PROGRS-8 for $50\% - 100\%$.

Efficiency across rollout and token budgets. Fig. 1 summarizes the accuracy–compute trade-off under different budget views. **Accuracy vs. rollout budget** (Fig. 1a) shows that PROGRS-4 reaches accuracy comparable to DAPO baselines with substantially fewer rollouts, while PROGRS-8 matches or exceeds DAPO-16 at similar budget levels, indicating a more favorable accuracy–computation frontier. **Token efficiency** (Fig. 1b), together with Table I, suggests that PROGRS achieves higher accuracy per generated token on average, and that gains are not explained solely by longer generations. **Aggregate metrics** (Fig. 1c–d) further highlight the budget dependence: PROGRS-4 is most effective in constrained

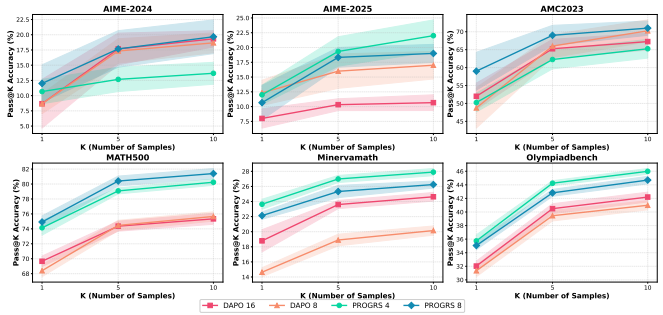

Fig. 2: Pass@K accuracy across benchmarks. Each panel shows Pass@1, 5, 10 for DAPO-16 (red), DAPO-8 (orange), PROGRS-4 (green), and PROGRS-8 (blue). Shaded regions indicate standard deviation across evaluation runs.

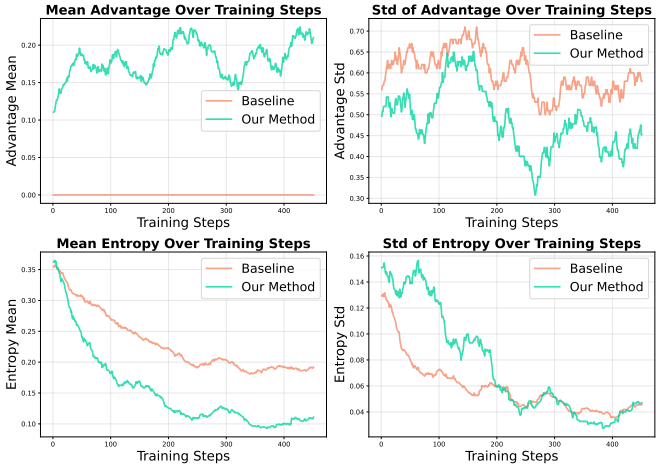

Fig. 3: Training dynamics of DAPO vs. PROGRS: mean and std. of advantages and per-token entropy over training (smoothed with a 50-step moving average).

regimes, whereas PROGRS-8 provides additional gains primarily at mid-range budgets, consistent with diminishing returns at higher rollout counts. **Pass@K Performance Analysis** Pass@K results (Fig. 2) show that PROGRS consistently outperforms DAPO across benchmarks and sampling budgets. In many cases, PROGRS-4 matches or exceeds DAPO-16, indicating better sample efficiency. Although all methods improve as K increases, the gap between PROGRS and DAPO remains, suggesting gains come not only from better exploration or sample ranking, but also from stronger base policy quality. Variance is generally lower or comparable under PROGRS, indicating similar or improved stability. Gains are largest on moderate- and high-accuracy benchmarks such as MATH-500 and AMC, and smaller in low-accuracy settings like AIME.

Training Dynamics Analysis. We analyze optimization behavior by tracking advantage statistics and policy entropy during training (Fig. 3). Compared to DAPO, PROGRS exhibits more stable dynamics. It maintains a more consistent advantage signal with lower variance, indicating reduced sensitivity to outlier trajectories and more stable gradient updates. In contrast, the baseline shows larger fluctuations. Entropy under PROGRS decreases more smoothly with smaller variance, suggesting a more gradual and stable transition from exploration to exploitation. Overall, these results indicate improved optimization stability under PROGRS.

D. Ablation Studies

PROGRS introduces two mechanisms designed to make process-reward guidance useful without undermining verifiable outcome supervision: (i) *outcome-conditioned centering* (Eq. 2), which removes any systematic PRM-driven advantage offset on incorrect trajectories while preserving relative preferences within the incorrect set; and (ii) a *coherence penalty*, which downweights trajectories whose step-level PRM scores exhibit abrupt local fluctuations. To isolate the contribution of each mechanism, we ablate them one at a time while holding fixed the policy model, PRM, rollout budget, decoding/evaluation protocol, and all other hyperparameters.

Effect of the coherence penalty ($\alpha_{\text{coh}} = 0$). We first remove the coherence term while *retaining* outcome-conditioned centering, to test whether coherence contributes beyond the core safeguard against PRM misalignment. Disabling coherence reduces accuracy relative to the full PROGRS configuration on most benchmarks, with particularly clear drops on AMC-2023 (from 59.00 to 50.50) and MATH-500 (from 74.92 to 71.82), and smaller but consistent decreases on MinervaMath and OlympiadBench. In parallel, average generation length typically decreases (e.g., MATH-500 from 683.74 to 669.70 tokens), indicating that coherence removal tends to shorten outputs but at the cost of correctness. This pattern is consistent with the intended role of coherence: it does not change which solutions are ultimately rewarded by verifiable outcomes, but it improves the *quality* of process guidance by suppressing volatile step-level PRM signals that can otherwise encourage locally confident but unstable reasoning segments. The changes in run-to-run variability are mixed across datasets, but several tasks show increased variability when coherence is removed (e.g., AIME-24 and MATH-500), suggesting coherence can also act as a stabilizer for the PRM-derived component of the advantage.

Effect of outcome-conditioned centering (No Centering). We next remove outcome-conditioned centering while keeping the PRM signal and all other settings unchanged, to directly test the failure mode that motivates PROGRS: absolute PRM scores may systematically over-reward trajectories that appear locally coherent even when the final answer is incorrect. This ablation produces the largest performance degradation overall, including a marked drop on MATH-500 (74.92 to 67.78) and a consistent decrease on OlympiadBench (35.06 to 30.77). Crucially, removing centering also tends to *increase* generation length on harder or out-of-distribution settings (e.g., AIME-25 from 1156.79 to 1273.70 tokens; OlympiadBench from 958.38 to 975.54 tokens), indicating that without centering the model is incentivized to produce longer trajectories that score well

TABLE II: Comparison with prior PRM-based methods under a shared base model. Avg@8/16 (%).

Dataset	PROGRS Avg@8	PROGRS Avg@16	PROF-GRPO Avg@16	PRL Avg@8
AMC23	58.75	60.47	49.1	45.94
MATH-500	75.42	74.82	73.2	71.00
Minerva	22.15	22.36	30.0	28.63
Olympiad	37.91	38.02	36.1	33.57
AIME24	10.83	10.21	9.6	9.17

under the PRM but do not translate into higher correctness. This directly matches the hypothesized misalignment: when incorrect samples can receive a net positive PRM offset, the learning signal can drift toward verbose, PRM-favored reasoning that competes with outcome supervision rather than complementing it. Together, these ablations support a clear division of labor between the two components. Outcome-conditioned centering is the primary safeguard that preserves *outcome dominance* by preventing systematic reward uplift on incorrect trajectories. Without it, PRM scores can drive reward hacking and reduce both accuracy and efficiency. The coherence penalty provides a secondary but meaningful benefit by refining the process signal within the safe regime enforced by centering: it tends to improve Pass@1 while also preventing pathological short/unstable reasoning patterns that arise when step-level PRM confidence is locally erratic. In addition, centering alone remains competitive with (and often stronger than) the outcome-only baselines. However, the full method achieves the most robust improvements across benchmarks when both mechanisms are combined.

E. Comparison with Prior PRM-Based Methods

We compare PROGRS with recent PRM-based RL methods, PROF-GRPO [11] and PRL [10], using a shared Qwen2.5-Math-1.5B-base model. For PROGRS-8, results are evaluated using Avg@8/16 metrics over rollouts.

Results for PROF-GRPO and PRL are taken from their original papers. Although all methods use the same base model, differences in training and optimization make the comparison indicative rather than strictly controlled.

PROGRS outperforms prior methods on AMC23, MATH-500, OlympiadBench, and AIME24, while remaining competitive on MinervaMath, showing robust gains under distribution shift.

Comparison with Alternative PRM Integrations. Beyond these ablations, we compare PROGRS against several alternative ways of incorporating process-level supervision into RLVR under a matched rollout budget ($K = 4$). Specifically, we evaluate: *Global Centering*, which subtracts a single group-wise mean across all samples; *Rank Shaping*, which converts PRM scores into pairwise Bradley–Terry style logits; *Step Shaping*, which applies per-step rewards without coherence modulation; and *PRM Filtering*, which discards low-coherence trajectories before optimization. As shown in Table III, PROGRS-4 achieves the highest Pass@1 on all six benchmarks, outperforming these alternatives on both in-distribution (MATH-500) and out-of-distribution settings (AMC/AIME, MinervaMath, OlympiadBench). This supports the view that outcome-conditioned centering, combined with coherence-aware aggregation, provides a more reliable way to use PRM signals than purely global normalization, rank-based shaping, or heuristic trajectory filtering.

TABLE III: Rollout-matched baselines ($K = 4$). Pass@1 (%, higher better), mean±std over 10 runs.

	Model	AIME24	AIME25	AMC23	MATH-500	Minerva	Olympiad
Pass@1 (%)	Global Centering	13.3±2.6	3.0±3.1	50.8±3.5	63.8±0.7	14.1±0.6	30.4±1.0
	Rank Shaping	3.0±2.3	8.3±1.7	35.0±4.0	60.9±1.0	12.2±0.6	24.4±1.0
	Step Shaping	5.7±3.4	4.3±3.7	44.0±3.0	65.7±0.9	18.3±0.7	28.9±0.8
	PRM Filtering	6.7±2.6	3.0±1.8	47.0±4.2	63.0±0.7	15.7±0.7	27.4±1.1
	PROGRS-4	10.7±2.0	12.0±1.6	50.3±2.6	74.1±1.0	23.6±0.7	35.7±0.9

TABLE IV: Scaling to a 7B policy. Pass@1 (%), mean±std over 10 runs.

	Model	AIME24	AIME25	AMC23	MATH-500	Minerva	Olympiad
Pass@1 (%)	DAPO16-7B	24.0±1.3	13.3±2.1	66.0±4.1	77.3±0.5	36.0±0.9	38.8±0.9
	DAPO8-7B	20.0±2.1	10.7±4.4	58.0±6.0	68.2±1.1	26.0±1.2	34.5±0.6
	PROGRS4-7B	19.3±1.3	10.7±4.9	53.0±1.9	70.9±0.9	29.2±0.4	36.3±0.6
	PROGRS8-7B	22.0±2.7	12.7±2.5	57.0±1.9	**81.5±0.5**	**37.5±0.5**	**41.6±0.6**

F. Additional Analysis

Scaling to Larger Policy Models. To assess whether the benefits of PROGRS persist at larger model scales, we replicate our comparison with DAPO using Qwen2.5-Math-7B as the policy model while keeping the PRM architecture and training protocol unchanged. Table IV reports Pass@1 accuracy on all benchmarks for DAPO baselines (8 and 16 rollouts) and PROGRS variants (4 and 8 rollouts). Across datasets, PROGRS-8-7B matches or exceeds DAPO-16-7B while using only half the rollout budget, and achieves the strongest performance on MATH-500, MinervaMath, and OlympiadBench. These results show that the proposed advantage construction remains effective at larger model scales and preserves a favorable accuracy–budget trade-off.

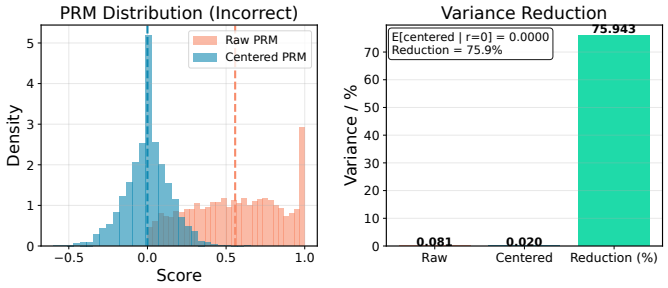

Fig. 4: Effect of outcome-conditioned centering on incorrect trajectories. Left: histogram of raw PRM scores vs. centered scores for $r = 0$ samples. Right: variance comparison; centering reduces variance by 75.9% while enforcing zero mean on incorrect bonuses ($\mathbb{E}[S^{\text{PRM}}|r=0] \approx 0.558$, $\text{Var}[S^{\text{PRM}}|r=0] \approx 0.081$; $\mathbb{E}[\tilde{S}|r=0] = 0$, $\text{Var}[\tilde{S}|r=0] \approx 0.020$).

Variance Reduction from Outcome-Conditioned Centering. Finally, we highlight the variance-reduction effect of outcome-conditioned centering on incorrect trajectories. Fig. 4 compares the distribution of raw PRM scores against the centered scores for samples with $r = 0$, and quantifies their empirical variance. Centering enforces a zero-mean constraint on incorrect bonuses and removes a dominant shared noise mode, reducing the variance of PRM-derived contributions by roughly three quarters.

V. Conclusion

We introduced PROGRS, a simple method for incorporating process reward models (PRMs) into outcome-verifiable RL for mathematical reasoning. It addresses a central failure mode of PRM-based training: absolute process scores can be misaligned with final correctness and may reward fluent but incorrect trajectories. PROGRS combines outcome-conditioned centering, which removes systematic PRM bias on incorrect trajectories, with coherence-based weighting, which down-weights unstable process signals. Because it adds no trainable components and uses a frozen PRM only for scoring, it integrates naturally into GRPO/DAPO-style training. Experiments across multiple math benchmarks show consistent Pass@1 gains and, in several settings, improved rollout efficiency, while ablations confirm that these gains come from centering and coherence rather than from simply adding process rewards.

Although PROGRS reduces sensitivity to absolute PRM miscalibration, it still relies on useful relative PRM preferences. Extending it beyond fully verifiable tasks will require alternative signals for defining outcome groups, such as consistency constraints, tool feedback, or learned evaluators. Overall, our results show that process rewards are most effective when used as outcome-constrained relative signals rather than absolute training targets.

Acknowledgment

The authors gratefully acknowledge the computing time made available to them on the high-performance computer Capella at the NHR Center at TU Dresden (ZIH). This center is jointly supported by the Federal Ministry of Research, Technology and Space and the state governments participating in the National High-Performance Computing (NHR) joint funding program (https://www.nhr-verein.de/en/our-partners).

References

[1] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, "Let's verify step by step," in *The Twelfth International Conference on Learning Representations*, 2023.

[2] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv:2501.12948*, 2025.

[3] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu *et al.*, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," *arXiv:2402.03300*, 2024. [Online]. Available: https://arxiv.org/abs/2402.03300

[4] Y. Wang, Q. Yang, Z. Zeng, L. Ren, L. Liu, B. Peng, H. Cheng, X. He, K. Wang, J. Gao *et al.*, "Reinforcement learning for reasoning in large language models with one training example," *arXiv e-prints*, pp. arXiv–2504, 2025.

[5] T. Wang, Z. Jiang, Z. He, S. Tong, W. Yang, Y. Zheng, Z. Li, Z. He, and H. Gong, "Towards hierarchical multi-step reward models for enhanced reasoning in large language models," *arXiv:2503.13551*, 2025.

[6] J. Skalse, N. Howe, D. Krasheninnikov, and D. Krueger, "Defining and characterizing reward gaming," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9460–9471, 2022.

[7] A. Pan, K. Bhatia, and J. Steinhardt, "The effects of reward misspecification: Mapping and mitigating misaligned models," *arXiv preprint arXiv:2201.03544*, 2022.

[8] Y. Miao, S. Zhang, L. Ding, R. Bao, L. Zhang, and D. Tao, "Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling," *Advances in Neural Information Processing Systems*, vol. 37, pp. 134 387–134 429, 2024.

[9] T. He, R. Mu, L. Liao, Y. Cao, M. Liu, and B. Qin, "Good learners think their thinking: Generative prm makes large reasoning model more efficient math learner," *arXiv:2507.23317*, 2025.

[10] J. Yao, R. Wang, and T. Zhang, "Prl: Process reward learning improves llms' reasoning ability and broadens the reasoning boundary," *arXiv:2601.10201*, 2026.

[11] C. Ye, Z. Yu, Z. Zhang, H. Chen, N. Sadagopan, J. Huang, T. Zhang, and A. Beniwal, "Beyond correctness: Harmonizing process and outcome rewards through rl training," *arXiv preprint arXiv:2509.03403*, 2025.

[12] Y. J. Park, K. Greenewald, K. Alimohammadi, H. Wang, and N. Azizan, "Know what you don't know: Uncertainty calibration of process reward models," 2025.

[13] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[14] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv:1506.02438*, 2015.

[15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[17] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[18] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan *et al.*, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv:2204.05862*, 2022.

[19] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *Advances in neural information processing systems*, vol. 36, pp. 53 728–53 741, 2023.

[20] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, W. Dai, T. Fan, G. Liu, L. Liu *et al.*, "Dapo: An open-source llm reinforcement learning system at scale," *arXiv:2503.14476*, 2025.

[21] Z. Yang, C. He, X. Shi, L. Li, Q. Yin, S. Deng, and D. Jiang, "Beyond the first error: Process reward models for reflective mathematical reasoning," *arXiv preprint arXiv:2505.14391*, 2025.

[22] W. Sun, Q. Du, F. Cui, and J. Zhang, "An efficient and precise training data construction framework for process-supervised reward model in mathematical reasoning," pp. 4292–4305, 2025.

[23] A. Yang, B. Zhang, B. Hui, B. Gao, B. Yu, C. Li, D. Liu, J. Tu, J. Zhou, J. Lin *et al.*, "Qwen2.5-math technical report: Toward mathematical expert model via self-improvement," *arXiv:2409.12122*, 2024.

[24] B. Pikus, P. R. Tiwari, and B. Ye, "Hard examples are all you need: Maximizing grpo post-training under annotation budgets," *arXiv preprint arXiv:2508.14094*, 2025.

[25] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the math dataset," *Sort*, vol. 2, no. 4, pp. 0–6.

[26] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo *et al.*, "Solving quantitative reasoning problems with language models," *Advances in neural information processing systems*, vol. 35, pp. 3843–3857, 2022.

[27] C. He, R. Luo, Y. Bai, S. Hu, Z. Thai, J. Shen, J. Hu, X. Han, Y. Huang, Y. Zhang *et al.*, "Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems," pp. 3828–3850, 2024.