CAPO: TOWARDS ENHANCING LLM REASONING THROUGH GENERATIVE CREDIT ASSIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) has significantly improved the reasoning capabilities of large language models (LLMs). However, existing RLVR methods provide limited control over intermediate reasoning quality. Value-based approaches such as PPO suffer from inaccurate credit assignment due to limited sampling, while process reward model (PRM)-based methods rely on costly human annotations and often produce unreliable process signals. To address these limitations, we propose Credit Assignment Policy Optimization (CAPO), a simple and efficient method. CAPO avoids training auxiliary models and instead leverages an off-the-shelf, general-purpose LLM as a Generative Process Reward Model (LLM-as-GenPRM) to generate step-wise critiques in a single pass based solely on step correctness. This enables explicit localized credit assignment to refine tokens that would otherwise receive identical rule-based rewards. As a result, CAPO significantly simplifies the training pipeline. Extensive experiments on LLaMA and Qwen backbones demonstrate that CAPO consistently outperforms six strong baselines across four mathematical benchmarks and three out-of-domain benchmarks. Further analyses show that CAPO's credit assignment mechanism substantially improves reasoning quality. Our implementation of CAPO is provided in `https://anonymous.4open.science/r/CAPO_Repo-45F1/`.

1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) has demonstrated significant success in enhancing the reasoning capabilities of Large Language Models (LLMs) (Lambert et al., 2024; Guo et al., 2025). Recently, a substantial body of work (Su et al., 2025; Lu, 2025; Sane, 2025; Dao & Vu, 2025; Tang et al., 2025) has extended the application of RLVR to diverse domains beyond mathematics and code, achieving notable progress.

Prevailing RLVR methods typically assign a binary reward to the entire response, propagating it uniformly to all tokens. *This approach suffers from two fundamental issues: it provides no differentiated feedback and ignores the credit assignment problem.* Figure 1 illustrates this limitation: for a single question, responses can vary substantially in quality, even when all are correct or all are incorrect. A correct response may have a fully valid reasoning process or contain a few flawed steps; similarly, an incorrect response may arise from pervasive errors or only a few incorrect steps. Thus, a single binary reward is insufficient to capture such fine-grained differences. **The lack of process supervision will potentially cause the model to learn spurious reasoning patterns.** However, as shown in Table 1, methods that attempt fine-grained credit assignment face their own challenges. Densification-based approaches convert sequence-level rewards into token-level feedback (Chan et al., 2024; Cao et al., 2025), but rely on learned reward models or complex external estimators, making them prone to reward hacking. Process reward models (PRMs) provide step-wise supervision but incur costly annotations (Lightman et al.), multiple inference calls (Chen et al., 2025), and unreliable credit assignment (Zhang et al., 2025). Like value models in PPO, PRMs estimate future returns under limited sampling, and optimizing them with Best-of-N objectives can induce a "process-to-outcome" shift (Zhang et al., 2025), where models prioritize final answers over sound reasoning.

To address these challenges, we introduce **Credit Assignment Policy Optimization (CAPO)**. CAPO discards probabilistic value estimation of process rewards and adopts a step-wise evaluation based solely on intrinsic correctness, such as flawed calculations or logic, rather than predict-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

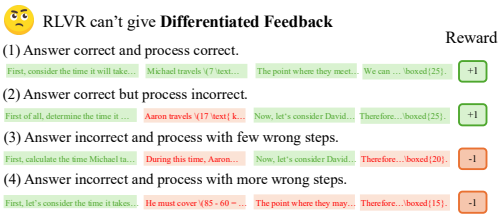

Figure 1: The coarse credit assignment problem in RLVR. RLVR assigns rewards based solely on the outcome, failing to provide differentiated feedback. This lack of granular feedback hinders the model's ability to learn robust reasoning processes.

Table 1: Comparison of existing methods to conduct credit assignment. Here N is larger than 1. CAPO is an efficient, simple, and general method that provides reliable credit assignment.

Method	High-Quality Data	Number of Inference Calls	Providing Reliable Rewards	Test-Time Scaling	Fine-grained Credits
PPO	×	1	×	×	✓
Densification-based RL	×	1	×	×	✓
PRM	✓	N	×	×	✓
CAPO	×	1	✓	✓	✓

ing long-term value. This simple approach reduces the risk of reward hacking. We implement CAPO using an off-the-shelf, general-purpose LLM as a Generative Process Reward Model (LLM-as-GenPRM), prompting it to produce step-wise critiques in a single pass. We then obtain the fine-grained feedback and apply a penalty to the credits of their corresponding tokens that were originally assigned identical rule-based rewards, which shifts learning from outcome- to process-oriented supervision. To balance process and outcome objectives, we analyze reward dynamics and apply reward shaping, while scaling the number of critiques and using a voting mechanism to enhance robustness. The resulting credits are then used for policy optimization.

The above design makes CAPO simple to implement and broadly applicable, as it does not require specialized or additional trained reward models. Moreover, leveraging publicly available LLMs as the GenPRM lowers the barrier for replication and extension by the community. We also find that with the generative credit assignment, CAPO achieves comparable or better performance compared to standard RLVR with around 2.5× efficiency gains. We summarize our main contributions as follows:

- We analyze the limitations of current RLVR approaches, identifying that uniform outcome-based rewards fail to provide differentiated feedback for complex reasoning steps, while existing PRM-based solutions suffer from high costs and efficiency problems.
- We introduce Credit Assignment Policy Optimization (CAPO), a simple and efficient framework that addresses the credit assignment problem by employing off-the-shelf LLMs as Generative Process Reward Models (LLM-as-GenPRM). This design eliminates the need for training specialized reward models, lowering the barrier for adoption and future extension.
- We conduct an in-depth analysis of the shift from outcome- to process-oriented learning (Sec 3.2). Identifying a critical conflict between these signals, we propose an asymmetric reward shaping strategy to mitigate it.
- Extensive experiments on four mathematical and three general reasoning benchmarks with Llama3-1B and Qwen2.5-1.5B/7B/14B show that CAPO consistently outperforms six strong baselines, including RLVR, SFT, and PRM-based methods. Notably, CAPO achieves superior performance with 2.5× efficiency gains compared to standard RLVR (Appendix A.1).

2 Problem Analyses

Reinforcement Learning provides a robust paradigm for fine-tuning Large Language Models (LLMs) beyond standard supervised objectives (Ziegler et al., 2019; Ouyang et al., 2022). In this paradigm, the LLM acts as a policy π_θ that sequentially generates tokens a_t (actions) given the preceding sequence s_t (state), forming a trajectory $\tau = (s_0, a_0, \ldots, s_T, a_T)$ where the state s_t typically represents the sequence of previously generated tokens (e.g., $\boldsymbol{o}_0, \ldots, \boldsymbol{o}_{t-1}$), and an action a_t is the selection of the next token $\boldsymbol{o_t}$ from the vocabulary $\mathcal{V}$. The goal is to optimize π_θ to maximize the expected cumulative reward $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \gamma^t r_t]$.

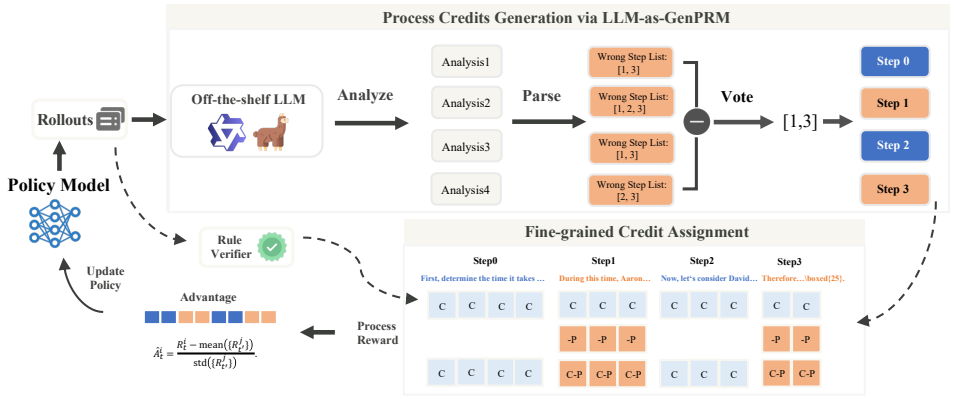

Figure 2: An overview of our method Credit Assignment Policy Optimization (CAPO). We utilize an LLM as a GenPRM to identify all incorrect steps within a model's generated rollout in a single pass. During credit assignment, we then suppress these erroneous steps, which prevents the correct portions of the sequence from being unfairly penalized. We denote W_{whole} and W_{process} as C and P for short.

While the framework is well-defined, a critical and unresolved challenge lies in credit assignment: determining which specific actions within a long trajectory are responsible for the final outcome and to what extent. Below, we analyze the limitations of prevailing credit assignment strategies, which motivates our work.

The Coarse-Grained Feedback Dilemma in Outcome-Based RL. A typical strategy in RLVR like Group Relative Policy Optimization (GRPO) (Shao et al., 2024), is to assign an identical reward to all tokens based solely on the final outcome as $\hat{A}_t^i = \frac{R^i - \text{mean}(\{R^i\}_{i=1}^G)}{\text{std}(\{R^i\}_{i=1}^G)}$ for any token $\boldsymbol{o_t}$, $t = \{1, \ldots, T\}$, where R^i is the response-level rule-based reward. The absence of granular feedback in GRPO means the model cannot differentiate between partially correct and entirely flawed reasoning. The lack of process supervision will potentially cause the model to learn spurious reasoning patterns.

The Vulnerability to Reward Hacking in Fine-Grained Approaches. We argue that the primary challenge in achieving fine-grained credit assignment lies in *providing reliable process credits while avoiding reward hacking*. A key factor in the success of methods like GRPO is their simple and direct reward design, which minimizes dependence on external sources (such as auxiliary models) and is thus robust against reward hacking. In contrast, while methods utilizing an auxiliary model—like a reward model (Chan et al., 2024) or a value model (Schulman et al., 2017)—can offer fine-grained credit assignment, they are inherently vulnerable to being hacked by this reward source. This vulnerability also extends to approaches based on PRMs. Beyond suffering from the high costs of online RL application (Chen et al., 2025) and manual annotations (Lightman et al.), PRMs face some more fundamental issues like **noisy reward modeling using Monte Carlo** (Zhang et al., 2025) and **reward hacking by summation of process reward** (Cheng et al., 2025).

3 CAPO: The Proposed Approach

To address the above challenges, we propose an approach called Credit Assignment Policy Optimization (CAPO), which aims to achieve fine-grained and reliable credit assignment efficiently, with an overview in Figure 2. We prompt the off-the-shelf LLM to obtain the critique for each step in a single pass. Then, penalties are applied to tokens in erroneous steps. Finally, we use these credits to form the advantages and update the policy.

3.1 Process Credits Generation via LLM-as-GenPRM

To avoid the complex modeling and dependencies of existing approaches, we propose to model the process credits in an explicit localized manner. Our method focuses on the intrinsic correctness of each step, rather than its estimated future returns. To achieve this, we introduce an LLM as a GenPRM (LLM-as-GenPRM) to identify erroneous steps in the responses produced by the policy model rollouts during RL. A key advantage of this framework is its immediate applicability using

existing open-source models without any domain-specific fine-tuning of the GenPRM itself. This not only validates the general reasoning capabilities of modern LLMs but also opens up a promising avenue for future work, where the GenPRM could be further fine-tuned on specific verification data for even greater accuracy and efficiency. We use the LLM to generate the rewards in a single pass for all steps efficiently. Given the training data $(\boldsymbol{q}, \boldsymbol{a}) \in \mathcal{D}$, we sample n rollouts from the policy model $\pi_{\theta_{\text{old}}}$: $\{y_i\}_{i=1}^{n} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \boldsymbol{q})$. For each rollout, we utilize LLM-as-GenPRM $\pi_{\text{LLM-as-GenPRM}}$ to generate k critiques:

$$\{c_j\}_{j=1}^{k} \sim \pi_{(\text{LLM}-\text{as}-\text{GenPRM})}(\cdot \mid I, (\boldsymbol{q}, \boldsymbol{a}), y_i),$$

where I is the critic prompt for LLM-as-GenPRM. Next, we extract the step indices of wrong steps from critiques: $\mathcal{S}_{i,j} = \text{ExtractIndices}(c_j), \forall j \in \{1, \ldots, k\}$, where $\mathcal{S}_{i,j}$ represents the set of wrong step indices identified by the j-th critique c_j for the i-th rollout y_i. For each rollout y_i, this process yields k sets of indices, denoted as $\{\mathcal{S}_{i,j}\}_{j=1}^{k}$. **Although a single sampling result from the judge already works well when $k = 1$, we can also optionally incorporate a voting mechanism to aggregate multiple sampled results and obtain more robust wrong step indices.**

These sets of indices are then aggregated through a *voting mechanism* to produce a final set $\mathcal{S}_i^*$ of erroneous step indices. The two typical voting mechanisms are:

- **Majority Vote:** A balanced strategy that declares a step as erroneous if it is flagged by at least half of the critiques. Let $\text{count}(s)$ be the number of times step index s appears across all sets $\{\mathcal{S}_{i,j}\}_{j=1}^{k}$. The final set is $\mathcal{S}_i^{\text{maj}} = \{s \mid \text{count}(s) \geqslant k/2\}$. This offers a compromise between intersection and union.
- **Union Vote:** This approach maximizes recall by flagging any step identified as erroneous by at least one critique: $\mathcal{S}_i^{\cup} = \bigcup_{j=1}^{k} \mathcal{S}_{i,j}$. It is comprehensive in capturing potential errors but is more susceptible to false positives, as a single faulty critique can introduce an error into the final set.

3.2 Fine-Grained Credit Assignment

For a given rollout $y_i = (o_1, o_2, \ldots, o_{L_i})$, where o_t is the t-th token and $L_i = |y_i|$ is the total number of tokens in the response, we aim to assign credit more precisely.

Localizing Contributions with Token-level Rewards. We initialize all the tokens in y_i with the score of response-level reward r_v from the rule-based verifier, scaled by a factor W_{whole}. Let $\mathcal{S}_i^*$ be the set of identified erroneous step indices for response y_i. We define $\mathcal{T}_{\text{err}}^i$ as the set of token indices belonging to these erroneous steps:

$$\mathcal{T}_{\text{err}}^i = \bigcup_{s \in \mathcal{S}_i^*} \{t \mid \text{token } o_t \text{ is part of step } s\}.$$

Then the token reward R_t^i for token o_t in y_i is :

$$R_t^i = r_v \cdot W_{\text{whole}} - \mathbb{I}(t \in \mathcal{T}_{\text{err}}^i) \cdot W_{\text{process}}, \tag{1}$$

where $\mathbb{I}(t \in \mathcal{T}_{\text{err}}^i)$ is an indicator function that equals 1 if token o_t is part of an erroneous step, and 0 otherwise. W_{process} is the factor controlling the magnitude of the penalty. Equation 1 integrates both process and outcome rewards into a single formulation. However, these two signals can empirically conflict, leading to challenging optimization dynamics. We therefore analyze their interaction and apply *reward shaping* to balance their influence.

Importantly, we do not accumulate rewards across steps. After the subsequent group-wise normalization, R directly serves as the final advantage. **This differs fundamentally from conventional PRM-based RL, where accumulated process rewards can easily lead to reward hacking** Cheng et al. (2025): maximizing accumulated process rewards is not necessarily aligned with the true optimization objective, often encouraging the model to generate unnecessarily verbose and useless responses.

The Shift from Outcome-Oriented Learning to Process-Oriented Learning. The introduction of a penalty for process errors, as formulated in Equation 1, transforms the optimization *from a single-objective setting into a multi-objective one*. This necessitates a careful investigation into how these two reward signals interact and how their balance can be calibrated for optimal performance.

To analyze this interaction, we first consider the mode where process-based rewards dominate. Empirically, we observe that when the process penalty is too high, the response length and per-token reward increase, while accuracy declines. We find that the model will overfit to a certain behavior pattern, optimizing for easily attainable process-level rewards (e.g., generating long, simple, but correct steps) at the expense of the final goal. In contrast, a dominant outcome-based reward acts as a crucial corrective force, anchoring the policy back to the primary objective. We summarize our key insights about the mechanism of interaction of these two signals below, and *the detailed discussion and validation can be found in the experiments and Appendix.*

Observation 1: *Learning from the process is a more challenging objective. In early optimization, accumulating correct process steps might not lead to a correct result, and even slow down the training.*

Observation 2: *The process-oriented signal becomes critically important in the later stages of optimization. It helps to differentiate and prioritize samples with better reasoning among numerous responses that achieve the same outcome.*

These observations lead to our *reward shaping* design, which can be framed in the context of Multi-Objective Reinforcement Learning (MORL) (Mossalam et al., 2016; Wu et al., 2023). Our approach employs a *linear scalarization* of the two objectives. Based on the observations of the learning process, we adopt an *asymmetric configuration* ($W_{\text{whole}} > W_{\text{process}}$) in Equation 1 to construct the reward formulation. By setting $W_{\text{whole}} > W_{\text{process}}$, we ensure that any trajectory leading to a correct answer is *always* preferred over any trajectory leading to a wrong one, regardless of process errors. Within this primary constraint, the model is then secondarily encouraged to refine its reasoning process.

Group-Relative Per-Token Normalization. After obtaining the per-token reward R_t^i, the advantage $\hat{A}_t^i$ for token o_t in rollout y_i is calculated by normalizing $R_{i,t}$ using the mean and standard deviation of all per-token rewards across a batch of n rollouts. Let $\{(y_j, \{R_{t'}^j\}_{t'=1}^{L_j})\}_{j=1}^n$ be the set of rollouts and the per-token rewards in the batch. The fine-grained advantage estimate is:

$$\hat{A}_t^i = \frac{R_t^i - \text{mean}(\{R_{t'}^j\}_{j=1,\ldots,n,\ t'=1,\ldots,L_j})}{\text{std}(\{R_{t'}^j\}_{j=1,\ldots,n,\ t'=1,\ldots,L_j})}. \tag{2}$$

This formulation ensures that *the advantage estimates directly reflect the reliable, fine-grained and token-level feedback*, guiding the policy update to specifically suppress the generation of erroneous segments while appropriately rewarding correct ones. Then, we derive the final objective function below and subsequently use this objective function to update the policy model:

$$\mathcal{J}_{\text{CAPO}}(\theta) = \mathbb{E}_{q\sim Q,\ \{y_i\}_{i=1}^n \sim \pi_{\theta_{\text{old}}}(o|q)}$$
$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{L_i}\sum_{t=1}^{L_i}\min\left\{\rho_{i,t}(\theta)\hat{A}_t^i, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t^i\right\} - \beta \mathrm{D}_{KL}\left[\pi_\theta \| \pi_{\text{ref}}\right].$$

where the ratio is given by $\rho_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}$.

4 Experiments

4.1 Experiments Setup

Implementation Details. We utilize Llama-3-1B-Instruct (Dubey et al., 2024) and Qwen2.5-Math series (1.5B, 7B, and 14B) (Team, 2024) for our experiments. Regarding the off-the-shelf LLM as GenPRM, we employ a lightweight instruction-following model Qwen3-4B-Instruct-2507 as the GenPRM for the Qwen and Llama experiments, without any task-specific fine-tuning. And we further utilize Qwen2.5-14B-Instruct to show the performance of CAPO when using larger or more capable judge models. PURE Cheng et al. (2025) uses the well-trained PURE-PRM-7B or Qwen2.5-Math-PRM-72B to guide the RL. We follow the Zero-RL setting proposed by Zeng et al. (2025), utilizing their 8k difficulty-stratified dataset for RL training. All experiments are performed on NVIDIA H20 96G GPUs, with detailed hyperparameters provided in Table 8. For step segmentation, we adopt the standard delimiter $\backslash n\backslash n$, following widely accepted conventions in process

Table 2: Performance comparison across different methods. We evaluate Qwen2.5-Math-1.5B/7B and Llama-3.2-1B-Instruct and report the pass@1 in percentages. CAPO($k = 1$) indicates that only a single critic is sampled from the judge, whereas the majority-voting and union voting mechanisms use $k = 4$ samples.

Method	General Reasoning Tasks (OOD)			Math Reasoning Tasks (ID)				Math Mean	All Mean
	GPQA	ARC-C	MMLU Pro	MATH	Olympiad Bench	AIME	AMC		
Qwen2.5-Math-7B									
Base Model	18.18	30.32	16.71	64.73	19.80	11.35	42.28	34.54	29.05
SimpleRL-Zoo	20.71	54.10	43.32	79.60	41.48	31.98	55.05	52.03	46.60
PURE (7B PRM)	23.23	57.81	42.72	**83.80**	45.19	25.42	56.48	52.72	47.81
PURE (72B PRM)	25.25	51.62	40.96	81.60	43.41	30.52	59.11	53.66	47.50
RLOO	26.26	55.55	**43.51**	77.20	42.96	25.73	60.58	51.62	47.40
Eurus-2-7B-PRIME	16.16	**71.33**	35.31	81.20	43.70	26.35	58.89	52.54	47.56
Qwen2.5-7B-DPO	**29.80**	67.15	33.70	82.80	**46.96**	23.02	51.24	51.01	47.81
Qwen2.5-7B-RAFT-Zero	16.16	63.31	35.20	78.20	40.89	16.88	53.69	47.41	43.47
CAPO (Judge 4B, k=1)	26.77	58.02	41.76	79.80	41.78	30.94	58.51	52.76	48.22
CAPO-Major (Judge 4B)	25.76	55.03	43.31	79.80	44.74	27.40	**63.22**	53.79	48.46
CAPO-Union (Judge 4B)	20.20	56.57	42.75	81.20	42.07	**35.21**	58.21	54.17	48.03
CAPO (Judge 14B, k=1)	21.89	64.85	41.20	81.67	45.14	35.00	59.10	**55.23**	**49.83**
Qwen2.5-Math-1.5B									
Base Model	5.05	18.20	10.30	63.13	29.43	9.97	41.57	36.02	25.38
SimpleRL-Zoo	11.11	27.99	18.48	73.20	33.33	10.00	46.99	40.88	31.58
PURE (7B PRM)	**14.14**	39.16	20.96	71.80	34.22	10.00	48.19	41.05	34.07
PURE (72B PRM)	13.64	34.13	19.09	70.60	34.67	10.00	**53.01**	42.07	33.59
RLOO	13.64	25.09	17.31	69.40	32.30	6.67	48.19	39.14	30.37
CAPO (Judge 4B, k=1)	13.13	32.42	18.49	72.20	35.70	10.00	50.60	42.13	33.22
CAPO-Major (Judge 4B)	12.63	34.90	19.84	71.00	35.70	13.33	50.60	42.66	34.00
CAPO-Union (Judge 4B)	10.10	37.80	20.03	72.00	34.22	16.67	51.81	43.67	34.66
CAPO (Judge 14B, k=1)	13.13	**43.34**	**24.28**	**76.80**	**40.20**	**18.19**	44.73	**44.98**	**37.24**
Llama-3.2-1B-Instruct									
Base Model	18.69	6.91	1.30	22.60	4.89	5.65	1.77	8.73	8.83
SimpleRL-Zoo	20.71	54.69	17.00	28.60	5.48	3.33	11.60	12.25	20.20
PURE (7B PRM)	23.74	52.56	**17.46**	29.80	5.48	0.21	12.54	12.01	20.26
PURE (72B PRM)	**26.77**	31.66	10.02	29.60	6.52	3.12	11.78	12.76	17.07
RLOO	24.24	26.71	11.64	30.80	6.22	0.52	10.58	12.03	15.82
CAPO (Judge 4B, k=1)	20.71	53.07	17.07	29.40	6.96	5.00	10.20	12.89	20.35
CAPO-Major (Judge 4B)	23.74	51.45	16.71	30.20	6.22	5.83	14.23	14.12	21.20
CAPO-Union (Judge 4B)	19.70	50.85	16.10	30.80	6.96	4.58	11.56	13.48	20.08
CAPO (Judge 14B, k=1)	17.00	**54.78**	16.56	**33.13**	**8.30**	**6.67**	**17.41**	**16.38**	**21.98**

reward modeling (Zhang et al., 2025; Yang et al., 2024; Zheng et al., 2025). Regarding the voting mechanism, we employ $k = 4$ critiques in our main experiments to balance computational efficiency with performance. We also only provide the ground truth answer to the judge.

Evaluation Setups. We evaluate the performance on four mathematical reasoning benchmarks: MATH (Hendrycks et al., 2021), Olympiadbench (He et al., 2024), AMC2023 (Mathematical Association of America, 2023), AIME2024 (Mathematical Association of America, 2024). We also include evaluation on three out-of-distribution benchmarks ARC-c (Clark et al., 2018),MMLU-Pro (Wang et al., 2024),GPQA-diamond (Rein et al., 2024), focusing on open-domain and general reasoning. To avoid contamination, we follow Yan et al. (2025) to shuffle the multiple-choice options. During evaluation, we use greedy decoding, and the final pass@1 results are averaged over three runs.

Baselines. We compare our method against 6 strong and widely-used baselines ranging from RL-based methods to SFT approaches. (i) **PURE** Cheng et al. (2025): A technique using PRMs to conduct min-form credit assignment for RL. (ii) **PRIME** Cui et al. (2025): This method employs implicit PRMs for process supervision; we utilize the *Eurus-2-7B-PRIME* checkpoint based on Qwen2.5-Math-7B. (iii) **SimpleRL-ZOO** Zeng et al. (2025): A reimplementation of the DeepSeek-R1 training recipe across various base models. (iv) **Iterative DPO** Xiong et al. (2025): A method that iteratively constructs preference pairs from the current policy for optimization via DPO loss. (v) **RLOO** Ahmadian et al. (2024): Utilizes leave-one-out baselines for group-level advantage calculation. (vi) **RAFT** Dong et al. (2023): Uses rejection sampling to generate synthetic data for supervised fine-tuning.

Table 3: Comparison of **Step Error Ratio (SER)** across different methods using Llama-3.2-1B, Qwen2.5-Math-1.5B, and Qwen2.5-Math-7B. Lower is better.

Model	Method	General Reasoning Tasks (OOD)				Math Reasoning Tasks (ID)				
		GPQA	ARC-C	MMLU Pro	OOD Mean	MATH	Olympiad Bench	AIME	AMC	Math Mean
Llama-3.2-1B	SimpleRL-Zoo	71.88	34.23	57.38	54.50	44.89	59.79	75.30	57.06	59.26
	CAPO	**65.77**	**31.06**	**52.57**	**49.80** (↓4.70%)	**39.65**	**55.55**	**58.83**	**49.19**	**50.81** (↓8.45%)
Qwen2.5-Math-1.5B	SimpleRL-Zoo	50.47	29.03	38.59	39.36	13.58	27.74	40.23	27.40	27.24
	CAPO	**47.36**	**27.55**	**34.90**	**36.60** (↓2.76%)	**10.69**	**22.71**	**31.74**	**22.49**	**21.91** (↓5.33%)
Qwen2.5-Math-7B	SimpleRL-Zoo	42.42	16.07	37.16	31.88	8.32	22.49	35.74	19.65	21.55
	CAPO	**38.44**	**10.85**	**27.60**	**25.63** (↓6.25%)	**8.07**	**20.42**	**27.99**	**15.09**	**17.89** (↓3.66%)

Table 4: Comparison of **Token Error Ratio (TER)** across different methods using Llama-3.2-1B, Qwen2.5-Math-1.5B, and Qwen2.5-Math-7B. Lower is better.

Model	Method	General Reasoning Tasks (OOD)				Math Reasoning Tasks (ID)				
		GPQA	ARC-C	MMLU Pro	OOD Mean	MATH	Olympiad Bench	AIME	AMC	Math Mean
Llama-3.2-1B	SimpleRL-Zoo	71.42	35.65	57.22	54.77	41.90	57.89	72.79	56.24	57.21
	CAPO	**63.51**	**31.96**	**50.88**	**48.78** (↓5.99%)	**36.88**	**52.53**	**54.70**	**47.95**	**48.02** (↓9.19%)
Qwen2.5-Math-1.5B	SimpleRL-Zoo	60.79	31.22	40.96	44.32	15.94	32.92	54.92	31.32	33.78
	CAPO	**55.12**	**29.19**	**36.97**	**40.43** (↓3.89%)	**11.83**	**26.71**	**35.19**	**24.88**	**24.65** (↓9.13%)
Qwen2.5-Math-7B	SimpleRL-Zoo	46.67	16.41	37.47	33.51	**9.16**	24.77	39.18	22.44	23.89
	CAPO	**43.84**	**10.97**	**27.78**	**27.53** (↓5.98%)	9.52	**23.54**	**31.05**	**17.35**	**20.36** (↓3.53%)

4.2 Main Results

As shown in Table 2, CAPO consistently outperforms six baselines across all evaluated model scales and families.

Efficiency of Generative Credit Assignment. A notable finding is the efficacy of our LLM-as-GenPRM approach without the need for auxiliary model training. Remarkably, the base CAPO variant utilizing merely an off-the-shelf 4B judge model achieves performance comparable to, and in many cases exceeding, the strongest baseline PURE, which relies on a heavily trained 7B process reward model (PRM). This demonstrates that zero-shot, step-wise critiques generated by a general-purpose LLM can effectively replace costly PRM training pipelines while providing highly accurate credit assignment.

Robustness through Voting Mechanisms. We further explore majority and union aggregation over multiple judge outputs. Majority voting generally improves the overall performance over single-sample CAPO, while union voting exhibits a different precision–recall trade-off. These results motivate majority voting as our default aggregation strategy.

Scaling the Judge Model. Scaling up the judge model leads to substantial performance boosts. A stronger judge inherently provides more nuanced and accurate signals for identifying logical fallacies in intermediate steps. On Qwen2.5-Math-1.5B, for example, CAPO with a larger judge model achieves the highest overall performance, outperforming PURE by a notable margin of +3.93% on math benchmarks. **Interestingly, we also observe that replacing the 7B PRM with the massively scaled Qwen2.5-Math-PRM-72B yields only marginal performance gains**, an observation that remains consistent across various temperature settings. Because PURE-PRM and Qwen-PRM rely on different training data, they exhibit inconsistent behaviors during RL. Thus, simply scaling the PRM to 72B does not strictly yield positive overall gains; rather, as shown in Table 2, while Qwen2.5-Math-PRM-72B improves mathematical reasoning, it comes at the cost of degrading general reasoning capabilities. Conversely, scaling our GenPRM from 4B to 14B results in a substantial performance boost. This contrast suggests that **the efficacy of CAPO does not solely stem from the capability of the judge model**. Rather, it is deeply rooted in our holistic credit assignment framework. Through the integration of a process penalty (as opposed to the advantage accumulation used in PURE), generative reward modeling, robust voting mechanisms, and an asymmetric reward scheme, the careful design allows CAPO to efficiently utilize a lightweight, off-the-shelf judge to outperform PRM-based RL methods that rely on much larger trained PRMs.

Generalization and Training Efficiency. The strong capability of CAPO generalizes seamlessly to other model architectures, as evidenced by similar substantial leads on the Llama-3.2-1B-Instruct

Table 5: Performance comparison across different methods using Qwen2.5-14B. CAPO uses Qwen2.5-32B-Instruct as a judge.

Method	General Reasoning Tasks (OOD)				Math Reasoning Tasks (ID)						All Mean
	GPQA	ARC-C	MMLU Pro	OOD Mean	MATH	Olympiad Bench	AIME24	AMC	AIME25	Math Mean	
Base	25.76	76.54	54.89	52.39	61.60	29.19	10.00	33.85	5.21	27.97	37.13
SimpleRL-Zoo	30.30	86.18	61.96	59.48	**81.40**	**44.59**	16.67	50.45	5.63	39.75	47.15
CAPO	**30.81**	**90.19**	**62.06**	**61.02**	80.00	42.96	**22.81**	**52.97**	**13.44**	**42.44** (+2.69%)	**49.41** (+2.26%)

Table 6: Performance comparison of CAPO with different reward configurations of W_{whole}(C) and W_{process}(P) on Qwen2.5-1.5B. The asymmetric setting leads to the best performance.

Method	General Reasoning Tasks (OOD)			Math Reasoning Tasks (ID)				Math Mean	All Mean
	GPQA Diamond	ARC_c	MMLU Pro	MATH 500	Olympiad Bench	AIME	AMC		
CAPO (C=1, P=2)	10.61	33.96	18.35	69.40	33.48	10.00	44.58	39.36	31.48
CAPO (C=1, P=1)	11.11	**36.77**	**19.17**	69.20	33.19	**13.33**	44.58	40.07	32.48
CAPO (C=1, P=0.1)	**13.13**	32.42	18.49	**72.20**	**35.70**	10.00	**50.60**	**42.13**	**33.22**

backbone. Even when applying our method to the much larger **14B base model** (evaluated in Appendix Table 5), CAPO consistently achieves a +2.69% over the rule-based SimpleRL-Zoo, highlighting its strong scalability and robustness. Furthermore, as detailed in Appendix A.1, our generative credit assignment achieves these remarkable gains with approximately **2.5× training efficiency** compared to traditional rule-based RL methods, underscoring CAPO as a highly practical and efficient solution for LLM alignment.

4.3 CAPO Indeed Improves Reasoning Quality

Since CAPO is designed to provide fine-grained credit assignment, we examine whether it leads to improved reasoning patterns and higher-quality reasoning processes. We use GPT-4o to serve as the evaluator to assess the process quality of different methods. To rigorously quantify the quality of the reasoning process, we define two metrics: **Step Error Ratio (SER)**: The percentage of reasoning steps identified as incorrect and **Token Error Ratio (TER)**: The percentage of total tokens belonging to incorrect steps. Let dataset $\mathcal{D}$ contain N generated responses. For the i-th response Y_i, let it consist of K_i steps $(s_{i,1}, \ldots, s_{i,K_i})$, where $|s_{i,k}|$ denotes the token count of step $s_{i,k}$, and $\mathbb{I}(s_{i,k})$ is the error indicator (1 if incorrect, 0 otherwise). We calculate the metrics by averaging the error rates per response: $\text{SER} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\sum_{k=1}^{K_i}\mathbb{I}(s_{i,k})}{K_i}\right)$, and $\text{TER} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\sum_{k=1}^{K_i}\mathbb{I}(s_{i,k})\cdot|s_{i,k}|}{\sum_{k=1}^{K_i}|s_{i,k}|}\right)$.

As shown in Table 3 and Table 4, CAPO demonstrates a **significant reduction in process errors compared to SimpleRL-Zoo** across both In-Domain (Math) and Out-Of-Domain benchmarks and achieves reductions ranging from 2.76% to 8.45%. The lower Token Error Ratio confirms that CAPO generates reasoning chains with fewer logic or calculation errors. These findings directly validate the motivation of CAPO.

4.4 Further Validation on 14B Base Model

As for the experiments applying CAPO to the Qwen2.5-14B model shown in Table 5, CAPO consistently outperforms the SimpleRL-Zoo baseline on the 14B scale. CAPO improves over SimpleRL-Zoo by 2.69 percentage points on the Math Mean and 1.54 percentage points on the OOD Mean. By demonstrating consistent gains across 1.5B, 7B, and 14B parameter scales, we empirically verify that CAPO is a generalized credit assignment framework capable of improving models across a wide range of scales, including larger models.

4.5 Ablation Study on Reward Formulation

Our key insight is that an *asymmetric* reward setup ($C > P$) is essential for establishing a clear learning hierarchy. A *symmetric* setting ($C = P$) confuses the model with ambiguous reward signals, while an extreme process-focus ($C < P$) breaks the hierarchy by penalizing minor process flaws more heavily than completely failed attempts. The results in Table 6 validate this analysis.

5 Related Work

Improving LLM Reasoning Capabilities Using RL Reinforcement learning has been widely applied to enhance LLMs' reasoning capabilities, but many challenges remain-such as optimization difficulties introduced by long reasoning (Yuan et al., 2025), how to better balance and improve the exploration–exploitation trade-off in the RL process (Yue et al., 2025), clipping-ratio issues (Yu et al., 2025), entropy decrease during RL (Wang et al., 2025), and so on. A range of algorithms aimed at improving RL have since emerged in the community, including GRPO (Shao et al., 2024), Reinforce++ (Hu, 2025), VC-PPO (Yuan et al., 2025), VAPO (Yue et al., 2025), DAPO (Yu et al., 2025). These works enhance existing RL algorithms from an algorithmic-optimization perspective, and we address the problem of imprecise or non-verifiable credit assignment in current approaches.

Credit Assignment *Credit assignment* (Sutton et al., 1998; Zhou et al., 2020) is a fundamental challenge in RL. Recent work mainly improves credit assignment through reward densification or PRMs. Reward-densification methods decompose sequence-level rewards into token- or step-level feedback (Chan et al., 2024; Cao et al., 2025). However, these methods rely on unverifiable reward signals and often suffer from inaccurate credit estimation. PRM-based methods Cheng et al. (2025) provide step-level rewards in RL, but their effectiveness is limited by unreliable annotations (Zhang et al., 2025) and mismatch with completion models (Setlur et al., 2024), leading to inaccurate rewards and suboptimal policies.

Generative Reward Model Recently, several works focused on using capabilities of language modeling in LLMs to improve reward modeling (Mahan et al., 2024; Zhang et al., 2024) and further scale it with COT reasoning (Zhang et al., 2024; Liu et al., 2025). And this technique is similar to previous concept "LLM as a judge" (Zheng et al., 2023) using LLMs to judge the response. Zhang et al. (2024) propose Generative Verifiers, which recast reward modeling as a next-token prediction task. Recently, Liu et al. (2025) introduced Generative Reward Modeling for flexible, inference-time scalable reward estimation and proposed to use principles to further scale the performance of GenRMs.

6 Conclusion

In this work, we identify the limitations of coarse-grained credit assignment in current RLVR approaches. To address this, we propose Credit Assignment Policy Optimization (CAPO), a simple and efficient framework, leveraging off-the-shelf, general-purpose LLMs as Generative Process Reward Models (GenPRMs) to produce step-wise credit assignment in a single pass. We therefore avoid the costly and complex pipeline of training auxiliary reward models. We further enhance the robustness of these signals through a voting mechanism and employ an asymmetric reward shaping strategy to effectively balance process and outcome supervision. Extensive experiments demonstrate that CAPO consistently outperforms strong baselines across a wide range of mathematical and general reasoning benchmarks. Qualitative analyses further confirm that CAPO effectively guides the model to learn correct reasoning pathways rather than exploiting spurious patterns and improves the reasoning quality. Crucially, the simplicity and generality of our framework make it a practical and accessible tool, lowering the barrier for extension and paving the way for future innovations in LLM reasoning.

AI Use Statement

Generative AI tools were used solely to aid and polish the writing of this paper, including improving grammar and readability. All AI-assisted edits were carefully reviewed by the authors, and we take full responsibility for the final content of this work.

Reproducibility Statement

We provide the algorithmic details of CAPO in Section 3, the experimental settings in Section 4 and Appendix B, the training hyperparameters in Table 8, the prompts used by the GenPRM in Appendix D, and additional analyses in Appendices A–C. An anonymized implementation of CAPO is also provided at `https://anonymous.4open.science/r/CAPO_Repo-45F1/` to facilitate reproduction of our experiments.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267. Association for Computational Linguistics, August 2024.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Meng Cao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Scar: Shapley credit assignment for more efficient rlhf. *arXiv preprint arXiv:2505.20417*, 2025.

Alex J Chan, Hao Sun, Samuel Holt, and Mihaela Van Der Schaar. Dense reward for free in reinforcement learning from human feedback. *arXiv preprint arXiv:2402.00782*, 2024.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, et al. Rm-r1: Reward modeling as reasoning. *arXiv preprint arXiv:2505.02387*, 2025.

Jie Cheng, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Gang Xiong, Yisheng Lv, and Fei-Yue Wang. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. *NeurIPS*, 2025.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

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

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Alan Dao and Dinh Bach Vu. Alphamaze: Enhancing large language models' spatial intelligence via grpo. *arXiv preprint arXiv:2502.14669*, 2025.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

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. *NeurIPS*, 2021.

Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Accurate credit assignment in rl for llm mathematical reasoning. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

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

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025.

Xun Lu. Writing-zero: Bridge the gap between non-verifiable problems and verifiable rewards. *arXiv preprint arXiv:2506.00103*, 2025.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models. *arXiv preprint arXiv:2410.12832*, 2024.

Mathematical Association of America. American mathematics competitions, 2023. AMC contest series.

Mathematical Association of America. American invitational mathematics examination, 2024. AIME contest series.

Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*, 2016.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Soham Sane. Hybrid group relative policy optimization: A multi-sample approach to enhancing policy optimization. *arXiv preprint arXiv:2502.01652*, 2025.

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

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding rl with verifiable rewards across diverse domains. *arXiv preprint arXiv:2503.23829*, 2025.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Weiliang Tang, Dong Jing, Jia-Hui Pan, Zhiwu Lu, Yun-Hui Liu, Li Erran Li, Mingyu Ding, and Chi-Wing Fu. Incentivizing multimodal reasoning in large models for direct robot manipulation. *arXiv preprint arXiv:2505.12744*, 2025.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

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

Xueru Wen, Jie Lou, Xinyu Lu, Junjie Yang, Yanjiang Liu, Yaojie Lu, Debing Zhang, and Xing Yu. Scalable oversight for superhuman ai via recursive self-critiquing. *arXiv preprint arXiv:2502.04675*, 2025.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 36:59008–59033, 2023.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning. *arXiv preprint arXiv:2502.19613*, 2025.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025.

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

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

Yufeng Yuan, Yu Yue, Ruofei Zhu, Tiantian Fan, and Lin Yan. What's behind ppo's collapse in long-cot? value optimization holds the secret. *arXiv preprint arXiv:2503.01491*, 2025.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun MA, and Junxian He. SimpleRL-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=vSMCBUgrQj.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:11853–11864, 2020.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Table 7: Performance comparison of different steps using Qwen2.5-Math-7B.The **gains** are computed using the results of SimpleRL-Zoo in the same step.

Method	Step	General Reasoning Tasks (OOD)				Math Reasoning Tasks (ID)					All Mean
		GPQA	ARC-C	MMLU Pro	OOD Mean	MATH	Olympiad Bench	AIME	AMC	Math Mean	
Base	–	18.18	30.32	16.71	21.74	64.73	19.80	11.35	42.28	34.54	29.05
SimpleRL-Zoo	60	23.23	59.44	43.69	42.12	80.53	41.53	26.67	**61.38**	52.53	48.07
	90	22.90	65.22	**44.70**	44.27	79.33	40.89	28.96	50.62	49.95	47.52
	120	24.75	60.75	44.69	43.40	81.93	43.56	24.17	60.17	52.46	48.57
	176	**24.57**	63.52	43.19	43.76	81.40	43.73	30.00	56.83	52.99	49.03
CAPO	60	22.89	58.06	43.94	41.63	81.33	43.17	28.11	60.49	53.28 (**+0.75%**)	48.28 (**+0.21%**)
	90	20.01	60.89	43.94	41.61	**82.27**	43.11	30.00	60.99	54.09 (**+4.14%**)	48.74 (**+1.22%**)
	120	23.89	**65.85**	43.20	**44.31**	81.67	**45.14**	**35.00**	59.10	**55.23** (**+2.77%**)	**50.55** (**+1.98%**)

A Discussion

A.1 Computational Efficiency and Convergence Analysis

Although integrating an external LLM-as-GenPRM introduces inference latency during the reward generation phase, a holistic analysis of the training budget reveals that CAPO is significantly more efficient than rule-based baselines. The core insight is that CAPO's fine-grained signal accelerates learning dramatically, allowing the model to reach optimal performance with far fewer training steps. Consequently, the *total GPU hours* required by CAPO **are substantially lower than those for sparse-reward methods** like SimpleRL-Zoo(or said GRPO).

(1) Efficiency via Faster Convergence. The primary advantage of CAPO is its ability to reach superior performance rapidly. As detailed in Table 7, we compared the training trajectory of CAPO against a fully trained SimpleRL-Zoo baseline (176 steps). Remarkably, at just **Step 60**, CAPO achieves a Math Mean of **53.28**, already surpassing the final performance of SimpleRL-Zoo at Step 176 (52.99). Converting this to computational cost (using 96 GPUs), CAPO requires only $\sim$**88 GPU hours** to beat the benchmark that SimpleRL-Zoo requires $\sim$**220.5 GPU hours** to establish. This implies that CAPO delivers superior reasoning capabilities with roughly **2.5$\times$ efficiency gains**. If we extend training to **Step 120**, CAPO achieves a dominant Math Mean of **55.23**—a **+2.2%** improvement over the best SimpleRL-Zoo result—while still consuming $\sim$**20% fewer total GPU hours** (approx. 175 hours) than the SimpleRL-Zoo baseline. This rapid convergence allows for early stopping strategies that effectively offset per-step overhead, resulting in a net reduction in the computational budget.

(2) Per-step Latency vs. Total Budget. Regarding the specific overhead, training Qwen2.5-Math-7B with a locally deployed Qwen2.5-14B-Instruct judge increases the per-step time from $\approx$47s (SimpleRL-Zoo) to $\approx$55s (CAPO). This moderate increase ($\approx$17%) is primarily due to the generative reward evaluation (12.1s) compared to the rule-based verification (4.5s) and **is further mitigated by using 4B judge with an increase of $\approx$10%**. However, as demonstrated by the convergence analysis above, this trade-off is highly favorable: our method yields high-quality signals that reduce the required training steps by half or more, making CAPO the more time-efficient option in practice.

A.2 The Reliability and the Capability of the LLM-as-a-GenPRM

The experiments above have already shown the judge's reliability across various sizes, including those smaller than the PRMs used. A key design choice in CAPO is the use of a general-purpose LLM as the GenPRM. The success of this approach hinges on the verifier's ability to accurately identify reasoning errors. One might question the reliability of such a model. However, we argue this design is robust for the following reasons. First of all, verification is easier than generation as has been discussed in Shinn et al. (2023); Bai et al. (2022); Wen et al. (2025). The solution provided transforms the task from abstract, ungrounded verification to a more constrained problem of error localization. The grounded context simplifies the verification task and enhances its reliability, as the model can cross-reference the intermediate steps with the known correct outcome.

A.3 Step Segmentation

The current implementation of CAPO assigns credit at the step level, using `\n \n` as a delimiter following *the widely accepted convention in the Process Reward Model (PRM) literature* (Zhang et al., 2025; Yang et al., 2024; Zheng et al., 2025). While most of the public open-source models like Qwen-series use this to separate the steps, we here still discuss the granularity and the intrinsic

advantage of our LLM-as-GenPRM approach: its inherent flexibility in handling task segmentation. Unlike rigid, rule-based systems that require predefined step structures, our framework can leverage the powerful multi-tasking and generalization capabilities of the LLMs (Brown et al., 2020; Ouyang et al., 2022; Chung et al., 2024). For our experiments, we wrapped the `\n \n` -delimited segments in `<step_i>` tags to construct the prompt. This was a deliberate choice for simplicity and reproducibility, especially since the Qwen models we used naturally structure their reasoning with this delimiter. However, this is not a fundamental requirement of CAPO. *It is important to clarify that CAPO is a general framework for credit assignment, and the use of* `\n \n` *is merely a domain-specific implementation for mathematical reasoning.* CAPO is agnostic to how steps are defined and can be easily adapted to other scenarios. In an Agentic workflow, the natural segmentation might be the turn of interaction or a tool call rather than a newline character. CAPO can be directly applied to these segments without modification to the core algorithm.

A.4 Prompt Sensitivity of the GenPRM

A potential concern for any method leveraging large language models is its sensitivity to the specific phrasing of the prompt. In CAPO, the GenPRM's performance could theoretically be influenced by the structure and wording of the verification prompt. While we found our chosen prompt (as detailed in Appendix D) to be effective and robust across different models and tasks, we acknowledge that prompt engineering can play a role in optimizing performance. A well-designed and well-considered prompt for GenPRM can probably improve the performance. We did not heavily tune the prompt used by LLM-as-a-GenPRM, but we indeed include some intuitive designs into the prompt, like asking the LLM to "*verify each step of the solution independently, without relying on the correctness of the context before. This means that the current step cannot be considered as incorrect simply because it used the conclusion derived from the previous erroneous reasoning*". We want the LLM to focus more on the quality or correctness of the currently verified step itself, rather than being influenced by the previous step. Preliminary experiments indicated that the model benefits from this design.

B Implementation Details

To avoid contamination, we follow Yan et al. (2025) to shuffle the multiple-choice options. During evaluation, we use greedy decoding and the final pass@1 results are averaged over three runs. As for the rule-based verifier, we use *Math-verify* to conduct an automatic judgment. We use verl (Sheng et al., 2025) to train the models.

We follow Kazemnejad et al. for automatic step segmentation. Specifically, we partition the model-generated response into steps using `\n \n` as the delimiter. We then enclose each step with the markers `<step_k>` and `</step_k>` to denote the k-th step. Subsequently, we construct the prompt for LLM-as-GenPRM following the structure shown in the Appendix D. For each evaluation sample, we first constructed a prompt containing the **[Question]**, the **[Ground Truth Solution]**, and the **[Solution To Judge]** using the prompt template D. The Case Study section provides a concrete example of a LLM-as-GenPRM generation.

All the hyperparameters during RL and SFT training can be found in Table 8. We follow the hyper-parameters set in (Hugging Face, 2025) as a widely accepted setting.

Table 8: Hyperparameters of RL settings.

Hyperparameter	Zero-RL
RL Stage	
Global Batch Size	192
PPO Epochs	1
PPO Mini Batch Size	192
KL Beta (`kl_beta`)	1×10^{-3}
Sampling Instances	8
LR Decay Style	Const
Rollout Top-p	0.9
Rollout Temperature	0.7
Judge Setting	
Temperature	0.7
Top-p	0.9

Learning Rates during RL Stage			
Model	**Qwen 1.5B**	**Qwen 7B**	**Llama 1B**
Learning Rate	1×10^{-5}	5×10^{-6}	2×10^{-6}

C Additional Experiments

C.1 GenPRM error-detection quality

We evaluated the GenPRM on ProcessBench Zheng et al. (2025), which contains high-quality human annotations of the first erroneous step and assigns -1 to a fully correct solution. We use the following metrics. **Error Acc** is the proportion of erroneous solutions for which the gold first-error index is predicted. **Correct Acc** is the proportion of correct solutions for which -1 is predicted. **PB-F1** is the harmonic mean of Error Acc and Correct Acc, following the ProcessBench evaluation perspective. **Loc-P/R/F1** measure micro precision, recall, and F1 over predicted step-index sets, with the gold set defined as $\{\text{label}\}$. Precision penalizes extra predicted indices, while recall penalizes a missed gold index.

For a single-output verifier, every prediction and every gold set contains exactly one index. Consequently, for them, Loc-P = Loc-R = Loc-F1 by construction. Their distinction becomes meaningful for ensemble variants, where majority/union aggregation may return multiple indices or an empty set. For these set-valued predictions, PB-F1 uses the relaxed "gold label in predicted set" criterion, whereas Loc-P/R/F1 reveal whether a higher hit rate is obtained simply by predicting more indices.

C.1.1 Evaluation Results

Overall detection quality. The off-the-shelf 4B GenPRM obtains 77.2 average Loc-F1. This matches the trained 72B PRM (77.1) and exceeds PURE-PRM-7B (58.5). Thus, the GenPRM provides reliable first-error localization rather than merely producing useful downstream rewards.

Judge-model size. Increasing the size of GenPRM from 4B to 14B improves Loc-P/R/F1 from 77.2 to 82.2 and PB-F1 from 76.9 to 83.2. Loc-F1 improves on all four subsets. This shows that stronger judges provide more reliable localization, especially on difficult reasoning trajectories, while the competitive 4B result shows that CAPO does not require a very large verifier.

Sampling frequency. For the 4B GenPRM, increasing the critique budget from k=1 to k=4 with majority voting improves every dataset's Loc-F1 and raises the average from 77.2 to 80.4. Recall increases more strongly than precision (77.2 to 82.7 versus 77.2 to 78.4), indicating that repeated sampling recovers errors missed by an individual stochastic critique without introducing many additional false positives. PB-F1 similarly improves from 76.9 to 82.1.

Voting strategy. Majority and union implement different operating points. Majority@4 achieves the best balanced localization result among the 4B settings (P/R/F1 = 78.4/82.7/80.4). Union@4 maximizes recall at 88.8 but lowers precision to 69.7, because any error proposed by one critique

is retained. Its high PB-F1 of 88.3 should therefore be interpreted together with Loc-P/R/F1: the relaxed ProcessBench hit rule rewards coverage, whereas localization precision exposes set expansion. Since a false positive in CAPO would penalize an otherwise correct step, majority voting is the safer default for balanced credit assignment; union voting is useful when missing an error is considered more costly.

Importantly, the trained 72B PRM has a higher single-sample PB-F1 than the 4B GenPRM (78.2 vs. 76.9), yet PURE with the 72B PRM achieves a lower downstream All Mean than the best 4B CAPO variant. This separates verifier quality from credit-assignment quality: even with a smaller verifier, CAPO's localized process penalty uses the detected errors more effectively than PURE's min-form credit assignment.

Table 9: Comparison of different verifier settings on localization and process-based verification benchmarks.

Verifier	Setting	Localization F1 (Loc-F1)				Average Metrics			
		GSM8K Loc-F1	MATH Loc-F1	OlympiadBench Loc-F1	Omni-MATH Loc-F1	Avg. Loc-P	Avg. Loc-R	Avg. Loc-F1	Avg. PB-F1
PURE-PRM-7B	single	73.5	67.2	49.7	43.5	58.5	58.5	58.5	56.9
Qwen2.5-Math-PRM-72B	single	88.0	79.6	72.4	68.3	77.1	77.1	77.1	78.2
Qwen3-4B-Instruct	single ($k = 1$)	81.8	82.8	72.5	71.7	77.2	77.2	77.2	76.9
Qwen3-4B-Instruct	majority@4	84.9	85.1	76.7	75.0	78.4	82.7	**80.4**	82.1
Qwen3-4B-Instruct	union@4	83.2	83.0	74.3	71.4	69.7	**88.8**	78.0	88.3
Qwen-14B	single ($k = 1$)	87.5	86.2	79.9	74.1	**82.2**	82.2	**82.2**	83.2

C.2 Case Study of Different Methods

This case in Figure 4 demonstrates that CAPO accurately interprets the geometric relationships in the diagram, correctly identifying EF as the side opposite angle D and DF as the hypotenuse. It then applies the sine definition and the Pythagorean theorem to obtain the correct answer, $\sqrt{51}$. In contrast, GRPO misreads the diagram at the first step, treating DE as the opposite side and incorrectly assigning the given length 7 to DF, which leads to the erroneous answer 4.9. This comparison highlights CAPO's stronger ability to preserve geometric consistency throughout its reasoning.

C.3 Case Study of Internal Variations in Rollout Responses

To concretely illustrate *the inadequacy of binary, outcome-based rewards*, we present a case study comparing four distinct model-generated responses during RL training to the same mathematical problem. These examples highlight how trajectories with identical final outcomes can possess vastly different internal reasoning qualities, a nuance that coarse-grained reward signals fail to capture.

We first examine two responses that arrive at the correct answer of 20. The correct response (top left) in Figure 3 demonstrates a flawless, direct, and correct reasoning process. In contrast, the response reaches the same correct answer through a spurious and convoluted line of reasoning, as highlighted in red. Under a standard RLVR framework, both responses would receive an identical positive reward, rendering the model unable to distinguish between sound logic and fortunate coincidence.

Next, we compare two responses that fail to produce the correct answer. The bottom left response in Figure 3 demonstrates a strong understanding of the core logic (i.e., that the numbers must be equal), and its failure is isolated to a minor off-by-one calculation error at the end. Conversely, the response in bottom right is fundamentally flawed, employing a completely incorrect and irrelevant problem-solving strategy from the outset. A binary reward scheme would assign the same penalty to both, failing to credit the largely correct reasoning in the third case while not sufficiently penalizing the complete conceptual failure in the fourth.

This case study underscores that a single binary reward signal obscures critical, process-level information. This lack of differentiated feedback prevents the model from learning to favor robust reasoning pathways over flawed or lucky ones, motivating the need for a more granular credit assignment mechanism.

D Prompt

Prompt for Generative Reward model. Below is the prompt for the generative reward model.

Prompt for Generative Reward Model

You are a math teacher. Use **[Ground Truth Solution]** to find any erroneous step in **[Solution To Judge]**. And the solution to be checked is separated with steps like "<step_i>...</step_i>" where i is the index of the step.
First, determine whether the solution is correct based on **[Ground Truth Solution]** and write it in the form "Judgement: Is the solution correct (Yes/No)? X", where X is either Yes or No.
If the solution is incorrect, review, analyze, and verify each step of the solution independently, without relying on the correctness of the context before. This means that the current step cannot be considered as incorrect simply because it used the conclusion derived from the previous erroneous reasoning.
Then provide the list of the indices of error steps separated by a comma. And the format is as follows:
<incorrect_steps>
...(the index of the incorrect steps)...
</incorrect_steps>.
Please give your answer directly.
[Question]
{problem}
[Ground Truth Solution]
{ground_truth solution}
[Solution To Judge]
{solution}

Training Prompt. We use the following prompt to conduct SFT and RL training.

Training Prompt

System
Please reason step by step, and put your final answer within \boxed{}.
User
{Question Content}

E CASE STUDY

The formal analysis of the case study is in Appendix C.2. The following are the Question, the response of CAPO, and the response of GRPO of the case study.

Taxonomy of Reasoning Outcomes

Problem: Mary and Jenna are playing a game in which each girl rolls a twenty-sided die with numbers from 1 to 20 on the faces. If the number on the die that Mary rolls is a factor of the number on the die that Jenna rolls, Mary wins. If the number on the die that Jenna rolls is a factor of the number on the die that Mary rolls, Jenna wins. For how many possible rolls would both girls win?

Case 1: Correct Ans & Correct Reasoning

To solve this problem, we need to find the number of pairs (m, j) where m is the number rolled by Mary and j is the number rolled by Jenna, such that m is a factor of j and j is a factor of m.

This implies that m and j must be equal, since m being a factor of j and j being a factor of m can only happen if $m = j$. **[✓ Valid Logic]**

Given that both Mary and Jenna are rolling a twenty-sided die, the possible values for m and j range from 1 to 20. Therefore, there are 20 possible outcomes where both m and j are equal.
Hence, the number of possible rolls where both girls win is:

$\boxed{20}$

Case 2: Correct Ans & Incorrect Reasoning

For Mary to win, her number must be a factor of Jenna's. For Jenna to win, her number must be a factor of Mary's. The only way both can win simultaneously is if the number on both dice is the same.

Given that each die has numbers from 1 to 20, we need to find how many numbers have at least one **other** number that is a factor of them... The number 1 is a factor of every number, so it cannot be the number... The numbers 2, 3... 20 each have at least one other number that is a factor of them. **[✗ Error: Flawed logical shift]**

Therefore, there are 20 numbers in the range from 1 to 20 that have at least one other number... Since both girls can roll any of these numbers and win, there are:

$\boxed{20}$

Case 3: Wrong Ans & Minor Reasoning Error

The winning conditions are as follows: Mary wins if $M|J$. Jenna wins if $J|M$. For both girls to win, these two conditions must be met simultaneously. For positive integers, if $M|J$ and $J|M$, the only possibility is that $M = J$.
Our task is to count how many pairs (M, J) satisfy $M = J$. The faces of the die are numbered from 1 to 20.

To count the number of integers in this range, we can subtract the minimum value from the maximum value. Therefore, the number of possible outcomes is: $20 - 1 = 19$. **[✗ Error: Off-by-one counting error]**

Since there are a total of 19 such possible numbers, there are 19 satisfying combinations.
Therefore, the total number of possible rolls is:

$\boxed{19}$

Case 4: Wrong Ans & Totally Wrong Reasoning

To solve this problem, we need to count the number of ways in which both can win...

Case 1: Jenna rolls a prime number.
(8 primes). Mary can only win if she rolls a 1. $\Rightarrow$ 8 ways.
Case 2: Jenna rolls a composite (not square).
(9 numbers). Mary has 2 choices. $\Rightarrow 9 \times 2 = 18$ ways.
Case 3: Jenna rolls a perfect square.
(1, 4, 9, 16). If $1 \Rightarrow 20 - 1 = 19$ ways. If $4 \Rightarrow 2 - 1 = 1$ way. If $9 \Rightarrow 2 - 1 = 1$ way. If $16 \Rightarrow 4 - 1 = 3$ ways. $\Rightarrow 24$ ways.
Total: $8+18+24 = 50$ ways. **[✗ Error: Hallucinated conditions and completely misunderstood game rules]**

Therefore, the total number of possible rolls where both girls win is:

$\boxed{50}$

Figure 3: **Four Types of Reasoning Outcomes.** A 2x2 comparison of different reasoning behaviors generated for the same problem. We categorize the outputs based on the correctness of the final answer and the validity of the underlying reasoning chain. **Top Left:** Perfect logic and answer. **Top Right:** Reaches the correct answer but hallucinates a flawed logical bridge. **Bottom Left:** Valid logic ruined by a simple off-by-one calculation error. **Bottom Right:** Completely misunderstands the problem constraints, leading to a complex but entirely false reasoning chain.

Case Study: CAPO vs. GRPO

Problem: In right triangle DEF, the right angle is at E, $EF = 7$, and $\sin D = 0.7$. What is the length of DE?

CAPO (Ours)

Since the right angle is at E, DF is the hypotenuse. Relative to angle D, the opposite side is EF. Therefore,

$$\sin D = \frac{EF}{DF}.$$

Substituting the given values gives

$$0.7 = \frac{7}{DF}.$$

Solving for the hypotenuse,

$$DF = \frac{7}{0.7} = 10.$$

[✓Correctly Identifies the Opposite Side and Hypotenuse]

By the Pythagorean theorem,

$$DF^2 = DE^2 + EF^2,$$
$$10^2 = DE^2 + 7^2,$$
$$DE^2 = 100 - 49 = 51.$$

Since a side length is positive,

$$\boxed{DE = \sqrt{51}}.$$

GRPO (Baseline)

GRPO begins by stating that

$$\sin D = \frac{DE}{DF}.$$

It then treats the side of length 7 as the hypotenuse:

$$DF = 7.$$

Using these assignments, GRPO writes

$$0.7 = \frac{DE}{7}.$$

[✗ Error: Misidentifies Both the Opposite Side and the Hypotenuse]

It consequently obtains

$$DE = 0.7 \times 7$$
$$= 4.9.$$

Therefore, GRPO concludes

$$\boxed{DE = 4.9}.$$

However, EF, rather than DE, is opposite angle D, and the given side of length 7 is EF, not the hypotenuse DF.

Figure 4: **Comparison of Geometric Reasoning. Left (CAPO):** Correctly identifies EF as the side opposite angle D and DF as the hypotenuse. It first derives $DF = 10$ from the sine ratio and then applies the Pythagorean theorem to obtain the correct answer $\sqrt{51}$. **Right (GRPO):** Misreads the triangle by treating DE as the opposite side and the given length 7 as the hypotenuse. This incorrect geometric interpretation leads to the erroneous answer 4.9.