HYBRID REINFORCEMENT: WHEN REWARD IS SPARSE, BETTER TO BE DENSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training for reasoning of Large language models (LLMs) increasingly rely on verifiable rewards: deterministic checkers that provide 0–1 correctness signals. While reliable, such binary feedback is brittle—many tasks admit partially correct or alternative answers that verifiers under-credit, and the resulting all-or-nothing supervision limits learning. Reward models offer richer, continuous feedback, which can serve as a complementary supervisory signal to verifiers. We introduce HERO (**H**ybrid **E**nsemble **R**eward **O**ptimization), a reinforcement learning framework that integrates verifier signals with reward-model scores in a structured way. HERO employs stratified normalization to bound reward-model scores within verifier-defined groups, preserving correctness while refining quality distinctions, and variance-aware weighting to emphasize challenging prompts where dense signals matter most. Across diverse mathematical reasoning benchmarks, HERO consistently outperforms RM-only and verifier-only baselines, with strong gains on both verifiable and hard-to-verify tasks. Our results show that hybrid reward design retains the stability of verifiers while leveraging the nuance of reward models to advance reasoning.

1 INTRODUCTION

Reasoning lies at the heart of human intelligence, and increasingly, at the frontier of large language model (LLM) capabilities (Zhang et al., 2025b). In tasks such as mathematical problems or generating proofs, reliable reasoning requires models not only to produce fluent text but also to generate logically consistent multi-step solutions that culminate in a verifiably correct outcome. Verifiable rewards operationalize this by running a deterministic checker—e.g., exact numeric/string match, unit tests, or symbolic equivalence—on a candidate solution y for input x; the checker accepts or rejects the output, yielding a sparse but unambiguous signal $r(x, y) \in \{0, 1\}$ that reinforcement learning can broadcast to the whole trajectory. Building on this principle, reinforcement learning from verifiable rewards (RLVR) (Chen et al., 2025b) uses these binary signals to train policies toward solutions that pass the checker. Recent systems—including OpenAI o1 and DeepSeek-R1—have advanced this paradigm at scale, leveraging verifier-grounded feedback to improve reasoning (Jaech et al., 2024; Guo et al., 2025; Zeng et al., 2025; Luo et al., 2025; Yang et al., 2024a).

However, strict 0–1 verification is inherently coarse and brittle: many reasoning tasks allow for partially correct solutions, equivalent answers in alternative formats, or open-ended outputs that resist exact matching. In such cases, symbolic verifiers may under-credit valid solutions (false negatives) or fail to provide any useful signal. Even when applicable, binary rewards induce sparsity: if all rollouts for a prompt receive the same label (all 0s or 1s), group-relative methods such as GRPO (Shao et al., 2024) yield zero relative advantage and thus no useful policy gradient, stalling policy improvement. Our motivating analysis in Section 3.1 further highlights this limitation: on samples where answers are hard to verify, rule-based verifiers frequently fail to capture correctness. Figure 1 illustrates this tradeoff: while reward models offer smooth but misaligned signals, rule-based verifiers enforce correctness but lack nuance. HERO integrates both to provide reliable yet informative supervision. This brittleness not only reduces sample efficiency but also skews optimization toward easier, strictly verifiable cases—leaving the hardest and most informative prompts underutilized.

Reward-based models, in contrast, offer dense supervision by scoring responses on a continuum (Yang et al., 2024b; Liu et al., 2024; Zhang et al., 2025c; Lyu et al., 2025; Liu et al., 2025).

Figure 1: **Comparison of reward signals from different supervision sources.** Green dots represent correct responses and red dots represent incorrect ones. The Reward Model (left) provides smooth but sometimes misaligned scores, as it can assign high values to incorrect responses. The Rule-based Rewards (middle) give strict binary signals but lack nuance and introduce true negative sometimes. HERO (right) integrates both, leveraging the verifier's correctness guarantees while refining gradients with reward model scores. This combination corrects for cases where the reward model alone may be wrong, leading to more reliable and informative supervision.

Rather than collapsing all incorrect answers into the same category, they can capture nuanced quality differences such as partial correctness, clarity of reasoning steps, or proximity to the ground truth. This graded feedback enriches training signals, helping policies learn from partially correct reasoning paths and better allocate credit across diverse rollouts. However, naively combining these dense reward model signals with a binary verifier output often destabilizes training. Specifically, when the reward model's continuous signals are naively blended with binary correctness checks, the resulting reward can become noisy or misaligned with the expected semantics of correctness. Thus, it remains an open question of *how to design an effective hybrid framework that preserves the reliability of verifiers while harnessing the richness of reward models*?

To address this challenge, we propose HERO (**H**ybrid **E**nsemble **R**eward **O**ptimization), a reinforcement learning framework that integrates verifier-anchored and dense reward-model signals in a structured way. HERO tackles the instability of naive blending through two key innovations. First, it introduces a stratified normalization scheme that bounds reward-model scores within verifier-defined correctness groups. This ensures that dense feedback refines learning only within the set of responses deemed correct by the verifier, preserving correctness guarantees while exploiting nuanced distinctions. Second, HERO employs a difficulty-aware weighting mechanism that adaptively adjusts the contribution of different prompts during training. Easy prompts, where most responses are uniformly correct or incorrect, contribute little additional learning signal and are down-weighted. In contrast, harder prompts—where candidate responses vary widely and reward-model scores provide valuable discrimination—are emphasized. These components allow HERO to overcome the brittleness of purely binary rewards and the noisiness of dense signals.

We evaluate HERO on diverse math reasoning benchmarks that span three regimes: verifiable tasks where exact final-answer checking is possible, hard-to-verify tasks with partially correct or format-sensitive solutions, and mixed settings combining both. Across different LLM backbones, HERO consistently outperforms both RM-only and verifier-only baselines, in all three regimes. Notably, on hard-to-verify tasks, HERO achieves 66.3, which surpasses RM-only (54.6) by +11.7 points and verifier-only (57.1) by a dramatic +9.2 points. Ablations further confirm that anchoring dense signals to verifiable correctness and adaptively reweighting difficult prompts are both critical for stability and efficiency.

2 PRELIMINARIES

Dense reward via reward modeling. Reward modeling learns a scalar function $r(x, y)$ that evaluates the quality of a response y given a prompt x. Based on the Bradley–Terry model (Bradley & Terry, 1952), the reward function is trained on pairwise preference data by minimizing

$$\mathcal{L}_R = -\mathbb{E}_{(x,y_c,y_r)\in\mathcal{D}}[\log \sigma(r(x, y_c) - r(x, y_r))], \tag{1}$$

where σ denotes the sigmoid function, y_c is the response that is considered preferred in a comparison, and y_r is the response considered less preferred. Once learned, r can guide reinforcement learning to align the model with human preferences.

Sparse reward via verifier. Reinforcement learning with verifiable rewards (RLVR) leverages a deterministic function $r(x, y)$ to assess correctness, assigning a sparse reward (e.g., $+1$ for correct, -1 for incorrect). All tokens in a response share the same reward, providing unambiguous supervision for tasks with objective ground truth. In mathematical problem solving, the reward function is based on a verifier that checks whether the model's solution matches the ground-truth reference under equivalence transformations. Specifically, a math verifier typically parses the predicted solution into a structured form (e.g., a symbolic expression, final numeric answer, or proof step), simplifies it, and compares it against the reference solution using symbolic algebra tools or logical equivalence checks. The reward function is based on the verifier:

$$\psi\big(x, y_i, y_{\text{ref}}\big) = \begin{cases} 1, & \text{if } y_i \text{ is equivalent to } y_{\text{ref}} \text{ given } x, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Group Relative Policy Optimization. GRPO (Shao et al., 2024) extends RLVR by optimizing over multiple responses per prompt rather than treating them independently. Instead of relying on a single trajectory, GRPO compares groups of candidate solutions and assigns relative advantages, which stabilizes learning and improves exploration. It also incorporates clipping (as in PPO) to prevent unstable updates and adds a KL penalty to keep the policy close to a reference model. This group-based formulation alleviates the gradient sparsity problem of pure verifier rewards and makes optimization more sample-efficient than standard PPO (Yu et al., 2025).

3 Methodology

3.1 Motivation: Delving into Rule-based vs. RM-based Verifiers

Building on the preliminaries, we now examine how the two supervision paradigms – rule-based verifiers that provide sparse but precise correctness signals, and reward models that offer dense but potentially noisy preferences – behave on tasks where correctness is difficult to verify. Since the reliability of training hinges on the quality of supervision, understanding their respective strengths and weaknesses is crucial. To this end, we use the `HardVerify_Math` benchmark (Xu et al., 2025) as prompts, generate three responses per problem from Llama3.1-8B, Llama3.3-70B, and Qwen3-8B, and then evaluate verifier and reward model performance on these samples.

Limitations of rule-based verifiers. To better understand the trade-offs among different verification approaches, we compare several representative verifiers. For rule-based verifiers, we consider `math_reward.py` from the verl library, `math_verify` module from verl, and the parse and verify functions from the `Math-Verify` library. In addition, we include more general verifiers that utilize a binary classifier trained to judge the correctness of answers, such as the `TIGER-Lab/general-verifier` (Ma et al., 2025).

Results in Table 1 highlight clear precision–recall trade-offs. *Function-based rules offer high precision but low recall*. For example, the `math_reward.py` checker is highly conservative: it almost never produces false positives (FPR=0.3%) but fails to recognize many correct answers, resulting in very low recall (10.1%). A more advanced variant, `math_verify.py (in verl)`, achieves the best balance—near-zero false positives with substantially higher recall. The `math_verify` library extends coverage with normalization heuristics (e.g., handling formatting differences or units) but remains brittle for mismatched orderings such as lists vs. sets, yielding only 38.6% recall.

Reward modeling can generalize to hard-to-verify samples. We further examine how reward models behave on hard-to-verify samples. Since correctness is directly checkable, most reward models for mathematical reasoning are trained on verifiable samples (Yang et al., 2024b; Liu et al., 2024; Zhang et al., 2025c; Lyu et al., 2025; Liu et al., 2025). This raises the question: to what extent can such models generalize to tasks where correctness cannot be directly verified? Here, we investigate this issue by analyzing the performance of a math-focused reward model (AceMath-7B-RM) on hard-to-verify tasks. We evaluate the model under varying thresholds of scores produced

Table 1: **Rule-based vs. RM-based verification performance.**

Type	Verifier	Recall ↑	Precision ↑	FPR ↓	Acc. ↑
Rule-based	`math_reward (verl)`	10.1	97.5	0.3	53.6
	`math_verify (verl)`	68.4	100.0	0.0	83.7
	`math_verify (library)`	38.6	96.1	1.6	67.6
	general-verifier	49.5	89.3	6.3	70.9
RM-based	AceMath-7B-RM w threshold 1	91.7	67.7	46.4	73.2
	AceMath-7B-RM w threshold 3	84.2	72.7	33.5	75.6
	AceMath-7B-RM w threshold 5	73.8	76.6	23.9	74.9
	AceMath-7B-RM w threshold 7	62.4	78.5	18.1	71.9

by the reward model. As shown in Table 1, at RM ≥ 1, the model achieves a high recall of 91.7%, and stronger overall coverage, significantly outperforming the rule-based verifiers. However, the precision is notably lower. Higher thresholds improve precision but reduce recall.

The need for hybrid reward design. Our analysis underscores a key tension: *neither rule-based verification nor reward models alone is sufficient*. Purely binary verifiable rewards can be brittle and overly conservative, especially on hard-to-verify samples. This not only reduces sample efficiency but also skews optimization toward easier, strictly verifiable cases—leaving the hardest and most informative prompts underutilized. Reward-based models, in contrast, offer dense supervision by scoring responses on a continuum and can capture nuanced quality differences such as partial correctness or proximity to the ground truth. These complementary strengths and weaknesses motivate a hybrid approach: anchoring supervision in symbolic verifiers to preserve correctness, while enriching it with the dense signal of reward models to drive effective policy learning. In the next subsection, we describe our proposed approach in detail.

3.2 HERO: Hybrid Ensemble Reward Optimization

Motivated by these findings, our design principle is that rule-based rewards should continue to guide the overall reasoning dynamics, while reward models serve as supplementary signals to enrich training. We therefore propose a *hybrid reward framework* that (i) augments binary correctness with dense reward-model scores and (ii) scales supervision according to prompt difficulty. We describe both components in detail below.

Dense signals anchored to verifiable correctness. As argued in the motivation, binary verifiers alone provide stable but overly coarse supervision, while reward models offer nuanced distinctions that are easily corrupted if left unconstrained. However, we found that a naive combination of rule-based verification and reward modeling signals tends to undermine the stability of training and render the hybrid approach ineffective (see Appendix A.3). Specifically, when the reward model's continuous signals are naively blended with binary correctness checks, the resulting reward can become misaligned with the expected semantics of correctness.

To address this, we propose *stratified normalization*, which explicitly bounds the continuous scores of RM within the symbolic structure imposed by the verifier. Formally, let $r_{\text{rule}} \in \{0, 1\}$ denote the verifier output and $r_{\text{RM}} \in \mathbb{R}$ the reward-model score. We partition responses by r_{rule} and apply group-wise min–max normalization to r_{RM}, yielding:

$$\hat{r}(x, y) = \begin{cases} -\alpha + 2\alpha \cdot \dfrac{r_{\text{RM}} - \min r_{\text{RM}}}{\max r_{\text{RM}} - \min r_{\text{RM}} + \epsilon}, & r_{\text{rule}} = 0, \\ (1-\beta) + 2\beta \cdot \dfrac{r_{\text{RM}} - \min r_{\text{RM}}}{\max r_{\text{RM}} - \min r_{\text{RM}} + \epsilon}, & r_{\text{rule}} = 1. \end{cases} \tag{3}$$

Here $\alpha, \beta \in (0, 1]$ control the allowable ranges for incorrect and correct groups, with $\epsilon > 0$ preventing division by zero. Technically, we set this value relatively small so that the training dynamic

is primarily led by rule-based rewards, and the reward from reward modeling is only supplementary. This design differs from traditional pure verifiable reward in the hard-to-verify samples and all-positive and all-negative groups, which do not provide the advantage over different rollouts.

This stratified normalization effectively embodies the hybrid approach: verifiers ensure the preservation of correctness semantics by constraining the score ranges, while reward models enhance the supervision by introducing gradations within each group. Incorrect responses are clearly distinguished from correct ones, and correct responses are prioritized based on their relative quality. In this manner, dense signals are anchored to symbolic correctness, mitigating the sparsity observed in pure RLVR.

Variance-aware advantage reweighting. In the motivation, we argued that not all prompts are equally informative: trivial ones provide little learning signal, while challenging prompts better reveal differences across candidate solutions. A shortcoming of the original GRPO algorithm is that it treats all prompts uniformly, ignoring this variability. The consequence is inefficient use of training capacity—easy prompts dominate optimization even though they provide little additional guidance, while difficult prompts that expose meaningful distinctions are underutilized. To realign training effort with informativeness, we introduce a *variance-aware weighting* scheme. For each prompt, let σ_u denote the standard deviation of reward-model scores across candidate responses, with $\bar{\sigma}$ as a running mean. This variance reflects uncertainty: higher values suggest greater disagreement and thus a richer training signal. We define a bounded monotone weighting function:

$$w_{\text{difficulty}}(\sigma_u) \;=\; w_{\min} + (w_{\max} - w_{\min}) \cdot \frac{1}{1 + \exp\big(-k(\sigma_u - \bar{\sigma})\big)}, \tag{4}$$

where $w_{\min}$ and $w_{\max}$ set the minimum and maximum weights, and k controls the slope of the transition. In practice, we treat these as tunable hyperparameters; unless otherwise stated, we use $w_{\min} = 0.5$, $w_{\max} = 2.0$, and $k = 5$, ensuring that difficult prompts are up-weighted by at most $2\times$ while trivial prompts retain at least half weight. The final shaped reward is

$$r_{\text{final}}(x, y) \;=\; w_{\text{difficulty}}(\sigma_u) \cdot \hat{r}(x, y). \tag{5}$$

This design operationalizes our intuition: ambiguous, high-variance prompts are emphasized because they reveal more about model weaknesses and reward-model sensitivity, while trivial, low-variance prompts are down-weighted to avoid wasting capacity. In doing so, the training process not only remains anchored to verifiable correctness through $\hat{r}$, but also allocates learning effort to the most challenging and informative parts of the data.

4 Experiments

4.1 Experimental setup

Training datasets. A central question raised is whether reasoning skills acquired through RLVR on verifiable data can generalize to tasks whose correctness cannot be mechanically checked. To empirically examine this, we design three evaluation regimes: verifiable-only, hard-to-verify-only, and mixed. To evaluate learning under different types of supervision, we construct three training regimes based on subsets of the OpenMathReasoning (Moshkov et al., 2025) benchmark. For the verifiable-only regime, we sample 2,000 problems whose final answers can be deterministically validated using a rule-based *math_verifier*. For the hard-to-verify-only regime, we likewise sample 2,000 problems from OpenMathReasoning, which consists of the correct answer whose format is very complex (see Appendix A.2.2 for how do we filter as well as some qualitative examples). These tasks supply dense preference signals and model-based verifier scores, but lack reliable binary labels from exact checking. Finally, in the mixed regime, we combine 1,000 verifiable and 1,000 hard-to-verify problems per epoch, enabling the policy to benefit simultaneously from robust exact-check supervision and nuanced feedback from unverifiable cases. Unless otherwise stated, mini-batches are stratified so that each epoch preserves the designated regime's composition, and variants or prompts are randomly resampled to reduce overfitting to a single rendition.

Model. To evaluate the generalizability of our method across different backbone models, we conduct experiments using the following models of various model families and sizes: we use Qwen3-4B-Base (Yang et al., 2025) and Octothinker Hybrid 8B base mode (Wang et al., 2025). Motivated

Table 2: **The results of Qwen-3-4B-Base on both verifiable and hard-to-verify reasoning tasks.** The first block shows results on verifiable tasks (MATH500, AMC, Minerva, Olympiad; with Avg.), and the second block shows results on hard-to-verify tasks (HVM, TBR).

	Verifiable tasks					Hard-to-verify tasks		
	MATH500	**AMC**	**Minerva**	**Olympiad**	**Avg. ↑**	**HVM**	**TBR**	**Avg. ↑**
Qwen3-4B-Base	67.5	44.1	29.4	32.1	43.3	45.2	40.2	42.7
SFT model	69.1	50.3	39.1	34.3	48.2	50.8	43.3	47.1
Training with verifiable samples								
Reward model	80.2	61.6	40.6	43.3	56.4	57.2	52.0	54.6
math_verify (verl)	82.3	61.3	44.0	45.5	58.3	61.0	53.1	57.1
General Reasoner	82.8	62.8	43.8	45.0	58.6	62.8	54.0	58.4
Qwen2.5-7B-It	83.7	58.1	43.1	47.4	58.1	68.0	57.1	62.5
HERO *(Ours)*	**85.4**	**69.4**	**44.5**	**48.9**	**62.0**	**73.2**	**59.3**	**66.3**
Training with hard-to-verify samples								
Reward model	79.6	58.8	39.9	42.1	54.1	59.2	48.2	53.7
math_verify (verl)	81.3	61.3	38.0	43.9	42.6	58.4	50	54.2
General Reasoner	78.6	56.3	38.7	41.5	53.8	**59.6**	48.4	54
Qwen2.5-7B-It	78.2	60.5	41.8	41.7	55.6	57.2	51.7	54.5
HERO *(Ours)*	80.0	63.4	40.7	43.1	**56.8**	59.0	**54**	**56.5**
Training with mixed samples								
Reward model	79.6	58.8	39.9	42.1	55.0	58.4	49.6	54.0
math_verify (verl)	81.3	61.3	38.0	43.9	56.1	62.4	55.3	58.9
General Reasoner	81.4	61.2	43.2	46.5	58.1	64.0	54.0	59.0
Qwen2.5-7B-It	80.4	63.1	40.5	48.0	58.0	68.8	57.7	63.3
HERO *(Ours)*	**81.6**	**64.4**	**42.1**	**47.0**	**58.8**	**71.4**	56.7	**64.1**

by stabilizing the RL training dynamic, we perform the cold start SFT on the base model (see Appendix A.2.1). All of the experiments of RL training start from the same SFT model.

Baselines. To provide a comprehensive comparison, we benchmark our hybrid-reward framework against both standard RL paradigms and stronger model-based references. As preliminary points of reference, we also report the performance of the base model and a cold-start SFT model, which serve to contextualize the impact of reinforcement learning itself. The main baselines are: (1) Reward model (RM)-only RL, which uses the AceMath-RM-7B reward model (Liu et al., 2024) to provide dense supervision; (2) Math verifier, which relies on binary, rule-based rewards, marking a sample as correct only if the normalized final answer matches the ground truth via *math_verifier* in the VERL repo—this emphasizes stability and reliability (3) General Reasoner, a frozen, well-trained 1.5B verifier model (Ma et al., 2025) that delivers binary correctness judgments, illustrating the potential of lightweight, task-agnostic model-based evaluation; and (4) Qwen2.5-7B-IT, which uses the Qwen2.5-7B-instruct verifier (Yang et al., 2024b). The proposed HERO combines rule-based verification when exact correctness is checkable with continuous reward-model signals otherwise, further enhanced by variance-aware advantage reweighting.

Evaluation for verifiable tasks. We report pass@1 averaged over 8 seeds in Table 2. Following a standard decoding protocol, we use temperature 0.6 and top-p 0.95, generate $N = 8$ candidates per problem, and evaluate the first decoded output (pass@1). Reported numbers are means over seeds. Correctness is decided by *math_verifier* (normalized numeric/string match with task-specific post-processing). Benchmarks include MATH500 (Hendrycks et al., 2021), AMC (Li et al., 2024), Minerva (Lewkowycz et al., 2022), and Olympiad (He et al., 2024).

Evaluation for hard-to-verify tasks. Since symbolic checkers cannot reliably provide binary labels for open-ended solutions, we adopt an *LLM-as-a-judge* protocol. Specifically, we use GPT-4o to compare model outputs against ground-truth answers. We evaluate using the HardVerify-Math benchmark (Xu et al., 2025), which consists of 250 samples. Based on the results in Section 3.1, we find that HardVerify-Math is not a particularly challenging filter, as using math_verify yields rel-

Table 3: **The results of OctoThinker-8B-Hybrid-Base on both verifiable and hard-to-verify reasoning tasks.** We report pass@1 averaged over 8 seeds. The first block shows results on verifiable tasks (MATH500, AMC, Minerva, Olympiad; with Avg.), and the second block shows results on hard-to-verify tasks (HVM, TBR).

	Verifiable tasks					Hard-to-verify tasks		
	MATH500	**AMC**	**Minerva**	**Olympiad**	**Avg. ↑**	**HVM**	**TBR**	**Avg. ↑**
OctoThinker-8B-Hybrid-Base	32.0	15.3	9.10	11.0	16.9	26.0	21.1	23.6
SFT cold start model	56.0	35.9	19.7	21.6	33.3	27.6	26.4	27.0
Verifiable only								
Reward model	62.3	38.4	26.2	25.5	38.1	29.6	27.8	28.7
math_verify (verl)	60.1	39.4	26.7	24.1	37.6	**31.6**	28.9	30.3
HERO *(Ours)*	**63.0**	**40.6**	**30.1**	**26.7**	**40.1**	28.4	**36.7**	**32.6**
Hard-to-verify only								
Reward model	60.7	33.8	22.4	24.9	35.4	**32.0**	29.8	30.9
math_verify (verl)	60.0	29.7	23.9	24.8	34.6	28.8	26.7	27.8
HERO *(Ours)*	**64.9**	**41.6**	**27.9**	**29.6**	**41.0**	32.4	**36.7**	**34.6**
Mixed samples								
Reward model	60.2	34.4	24.0	23.8	35.6	30.8	29.3	30.1
math_verify (verl)	59.3	33.7	24.7	24.0	35.4	27.6	28.7	28.2
HERO *(Ours)*	**65.2**	**38.1**	**28.1**	**29.3**	**40.2**	**34.8**	**31.6**	**33.2**

atively good results. Therefore, to further evaluate performance on hard-to-verify reasoning tasks, we additionally collect the TextBookReasoning dataset (Fan et al., 2025) (see Appendix A.2.3 for more details).

4.2 Main Results

Hybrid reward consistently improves performance across all regimes. Table 2 shows that HERO outperforms all baselines—including RM-only, rule-based verifiers, and LLM-as-verifiers—across verifiable, hard-to-verify, and mixed settings. On verifiable tasks, HERO achieves the best average (62.0), exceeding RM-only (56.4) and rule-based training (58.3). The key advantage is that stratified normalization allows HERO to fully exploit both positive and negative groups: while verifier-only training collapses all-correct or all-incorrect batches (yielding zero relative advantage), HERO preserves learning signal within each group via dense intra-group rewards. Model-based verifiers such as Qwen2.5-7B-IT and General Reasoner further expand coverage, achieving around 58.4–62.5, but still lag behind HERO since they provide only coarse binary labels and miss the fine-grained calibration offered by hybrid reward and could still not handle with the zero advantage for all positive/negative rollouts. On hard-to-verify tasks, HERO shows the largest margin, reaching 73.2 compared to 59.2 for RM-only and 42.6 for verifier-only. Here, rule-based verifiers fail because most responses collapse into the same label, while RM-only suffers from noise and reward drift. By anchoring dense signals to correctness groups, HERO achieves both stability and stronger supervision than either signal alone. In the mixed regime, HERO again secures the best average (58.8), surpassing RM-only (55.0) and rule-based verifiers (56.1). It is also worth noting that although LLM-as-verifier approaches can improve coverage, they are computationally expensive; in contrast, reward models are lightweight to deploy and far more efficient. HERO thus achieves superior accuracy while retaining the efficiency advantages of reward modeling, explaining why it consistently generalizes better than both symbolic and LLM-based verifier baselines.

Hybrid reward generalizes across backbones. A key observation is that hybrid training scales across models of very different capacities and starting strengths. Qwen3-4B already shows strong SFT results (48.2 on verifiable tasks) but gains large boosts from HERO, particularly on more challenging benchmarks such as AMC (+7.8 points over RM-only) and Olympiad (+3.4 points over verifier-only). In contrast, OctoThinker-8B begins with very low base performance (16.9 on verifiable and 23.6 on hard-to-verify), yet hybrid training raises its averages to 40.1/32.6 (verifiable-only) and 41.0/34.6 (hard-to-verify-only). The relative improvement is most pronounced in difficult settings, showing that hybrid reward is not tailored to a single architecture but rather captures a general

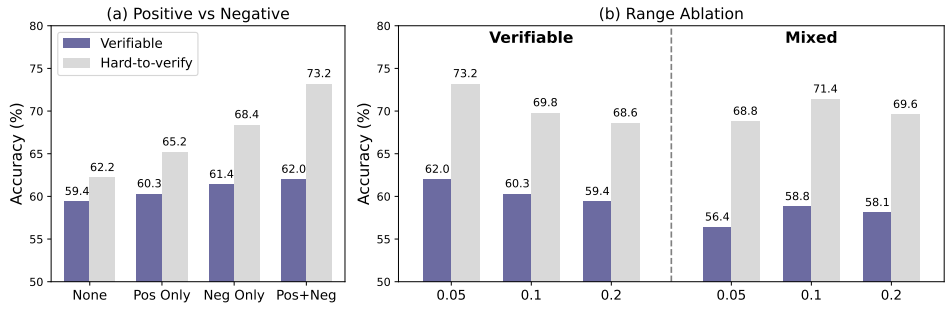

Figure 2: (a)Impact of using positive and negative dense ranges. (b)Effect of varying reward ranges under different training regimes. Left: Verifiable setting. Right: Mixed setting.

principle: exploiting the verifier's precision while refining gradients with reward-model scores produces robust supervision independent of scale.

Verifier-only training struggles on hard-to-verify tasks. The limitations of symbolic supervision become clear in the hard-to-verify regime. On Qwen3-4B, verifier-only training achieves just 42.6, far below both RM-only (59.2) and HERO (56.5) and is even worse than the cold start SFT model. OctoThinker-8B shows the same failure mode, with verifier-only reaching 34.6 compared to HERO's 41.0. The underlying issue is structural: group-relative optimization collapses when all candidate responses receive the same binary 0 label, producing no gradient. As a result, verifier-only approaches cannot differentiate between nearly correct and completely incorrect reasoning paths. Hybrid reward avoids this collapse by anchoring dense reward-model signals within correctness groups. This ensures that progress continues even when binary labels saturate, while still retaining the verifier's strict guarantees. Consequently, HERO is able to generalize better in precisely those regimes where purely symbolic feedback is least reliable.

4.3 Additional Ablations

Dense negative ranges are more important than positive samples. We found that dense negative rewards play a more critical role in stabilizing training and improving learning efficiency than positive samples. While positives signal correctness, negatives provide richer supervision by penalizing diverse reasoning errors. Notably, using only negative rewards boosts performance on verifiable tasks from 59.4 to 61.4, and even more on hard-to-verify tasks from 62.2 to 68.4. This demonstrates that well-calibrated negative ranges are essential: they provide broader feedback, enabling the model to detect subtle errors and generalize better on complex cases.

Variance-aware reweighting improves model's reasoning ability. We evaluated variance-aware reweighting based on reward-model score variance, which emphasizes ambiguous, high-variance prompts while down-weighting trivial ones to reduce overfitting. This dynamic adjustment yields consistent gains, particularly on hard-to-verify tasks where dense signals are most informative. As shown in Table 4, reweighting improves accuracy on both verifiable and hard-to-verify benchmarks, with larger gains in the latter (+3.8), confirming that focusing capacity on uncertain samples leads to more robust and generalizable improvements.

Table 4: Variance-aware reweighting improves performance on both verifiable and hard-to-verify samples.

Methods	Verifiable	Hard-to-verify
w/o reweighting	60.8	69.4
w reweighting	**62.0**	**73.2**

Reward range selection is crucial for balancing stability and performance. We conducted ablation studies to investigate the impact of varying reward ranges on model performance, as shown in Figure 2(b). For **verifiable tasks**, smaller reward ranges (e.g., $\alpha = 0.05$) yielded the best results, as the rule-based verifier's precision benefits from a tighter range that minimizes noise and maintains stability. Expanding the range beyond this threshold led to diminishing returns and increased

noise. In contrast, for mixed tasks, where many samples fail the rule-based verifier, the learned reward model plays a larger role. Here, larger reward ranges (e.g., $\alpha = 0.1$ or $\alpha = 0.2$) provided richer signals, allowing the model to learn more effectively from harder tasks. However, expanding the range beyond a certain point caused a slight performance drop due to overfitting or excessive noise. Overall, careful tuning of the reward range, particularly for the negative rewards, is crucial to balancing stability and performance, depending on the task type.

5 Related Work

Reinforcement learning from verifiable rewards. Reinforcement learning from verifiable rewards (RLVR) leverages deterministic correctness checks—such as passing unit tests or matching reference answers—to enhance model learning (Shao et al., 2024). Early program synthesis work demonstrated that agent-generated trajectories validated against ground truth outperform supervised approaches (Bunel et al., 2018; Chen et al., 2021). In the context of LLMs, rule-based verification plays a crucial role in filtering, providing training signals, and supporting benchmark evaluations (Xiong et al., 2025; Yu et al., 2025; Shao et al., 2024). Recent extensions include: outcome-driven RL (GRPO) for grounding and citation fidelity in QA tasks (Sim et al., 2025); rubric-anchored RL, which introduces structured rubrics for open-ended response evaluation (Huang et al., 2025b); verifier-free RL strategies like VeriFree, which bypass explicit checking while achieving performance on par with verifier-based methods (Zhou et al., 2025); and cross-domain RLVR, which employs LLM-derived scoring for domains lacking clear reference answers (Su et al., 2025). Despite these advancements, rule-based methods still struggle with semantically correct but textually divergent outputs, motivating the use of model-based verifiers (Chen et al., 2025a; Ma et al., 2025; Huang et al., 2025a; Xu et al., 2025). However, the coverage of LLM-based verifiers remains limited for the generalization (Li et al., 2025), and the rewards they provide are still sparse, often consisting of binary labels. In contrast to previous work, we propose a hybrid approach that combines rule-based verification with continuous, dense reward signals from learned models, allowing us to maintain the stability of verifiers while addressing their sparsity. By anchoring dense signals to symbolic correctness and introducing a variance-aware weighting mechanism, our method enables more informative, stable, and sample-efficient learning on both verifiable and hard-to-verify tasks.

Reasoning on hard-to-verify tasks. As the reasoning capabilities of large language models (LLMs) have reached new heights, increasingly challenging reasoning benchmarks have been proposed (Phan et al., 2025; Zhang et al., 2025a). These problems often involve complex outputs, such as natural language representations and intricate mathematical or physical formulas. In such cases, rule-based verification methods, while effective for well-defined problems, struggle to capture the nuances of these tasks. Recent work has focused on the use of LLMs as judges, where LLMs assess the quality of generated responses (Chen et al., 2025a; Ma et al., 2025; Huang et al., 2025a; Xu et al., 2025; Li et al., 2025), enabling more nuanced evaluations. However, despite its conceptual simplicity, LLM-as-judge may not always produce reliable assessments for domain-specific or long-form data. Some recent work proposes going beyond binary labels from verifiers for hard-to-verify tasks. For example, Gurung & Lapata (2025) applies reasoning traces in Next-Chapter Prediction (NCP) for long-form story generation via likelihood estimation, while Tang et al. (2025) uses Jensen's evidence lower bound to treat chain-of-thought reasoning steps as latent variables in the generative process. They directly get rid of the verifier component. In contrast, our work retains the use of verifiable rewards, but enhances supervision through the introduction of a reward model.

6 Conclusion

We introduced HERO, which anchors reward-model signals to verifier-defined correctness via stratified normalization and emphasizes informative prompts with variance-aware weighting. This hybrid design preserves the stability of verifiers while supplying dense, trajectory-sensitive feedback, mitigating gradient sparsity and RM-only drift. Empirically, HERO consistently outperforms RM-only and verifier-only baselines across verifiable, hard-to-verify, and mixed regimes and across backbones. Future work includes stronger difficulty estimators, process-level rewards, and extension beyond math reasoning.

REFERENCES

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Rudy Bunel, Matthew Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. Leveraging grammar and reinforcement learning for neural program synthesis. In *ICLR*, 2018.

Ding Chen, Qingchen Yu, Pengyuan Wang, Wentao Zhang, Bo Tang, Feiyu Xiong, Xinchi Li, Minchuan Yang, and Zhiyu Li. xverify: Efficient answer verifier for reasoning model evaluations. *arXiv preprint arXiv:2504.10481*, 2025a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Run-Ze Fan, Zengzhi Wang, and Pengfei Liu. Megascience: Pushing the frontiers of post-training datasets for science reasoning. *arXiv preprint arXiv:2507.16812*, 2025.

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.

Alexander Gurung and Mirella Lapata. Learning to reason for long-form story generation. *arXiv preprint arXiv:2503.22828*, 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. *arXiv preprint arXiv:2103.03874*, 2021.

Yuzhen Huang, Weihao Zeng, Xingshan Zeng, Qi Zhu, and Junxian He. Pitfalls of rule-and model-based verifiers–a case study on mathematical reasoning. *arXiv preprint arXiv:2505.22203*, 2025a.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Xiaomeng Hu, Yanmei Gu, Yuanyuan Wang, Zhengkai Yang, Jianguo Li, and Junbo Zhao. Reinforcement learning with rubric anchors. *arXiv preprint*, 2025b. arXiv:2508.12790.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

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

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

Xuzhao Li, Xuchen Li, Shiyu Hu, Yongzhen Guo, and Wentao Zhang. Verifybench: A systematic benchmark for evaluating reasoning verifiers across domains. *arXiv preprint arXiv:2507.09884*, 2025.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, et al. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*, 2025.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acemath: Advancing frontier math reasoning with post-training and reward modeling. *arXiv preprint arXiv:2412.15084*, 2024.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Chengqi Lyu, Songyang Gao, Yuzhe Gu, Wenwei Zhang, Jianfei Gao, Kuikun Liu, Ziyi Wang, Shuaibin Li, Qian Zhao, Haian Huang, et al. Exploring the limit of outcome reward for learning mathematical reasoning. *arXiv preprint arXiv:2502.06781*, 2025.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing llm reasoning across all domains. *arXiv preprint arXiv:2505.14652*, 2025.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *arXiv preprint arXiv:2504.16891*, 2025.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

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

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.

Shang Hong Sim, Tej Deep Pala, Vernon Toh, Hai Leong Chieu, Amir Zadeh, Chuan Li, Navonil Majumder, and Soujanya Poria. Lessons from training grounded llms with verifiable rewards. *arXiv preprint*, 2025. arXiv:2506.15522.

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

Yunhao Tang, Sid Wang, Lovish Madaan, and Rémi Munos. Beyond verifiable rewards: Scaling reinforcement learning for language models to unverifiable data. *arXiv preprint arXiv:2503.19618*, 2025.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025.

Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, et al. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*, 2025.

Zhangchen Xu, Yuetai Li, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. Tinyv: Reducing false negatives in verification improves rl for llm reasoning. *arXiv preprint arXiv:2505.14625*, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024a.

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*, 2024b.

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

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

Jie Zhang, Cezara Petrui, Kristina Nikolić, and Florian Tramèr. Realmath: A continuous benchmark for evaluating language models on research-level mathematics. *arXiv preprint arXiv:2505.12575*, 2025a.

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

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*, 2025c.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers. *arXiv preprint*, 2025. arXiv:2505.21493.

Appendix

A Experiments **13**

A.1 Experimental setup 13

A.2 Data preparation 15

A.3 More experiments 17

B Qualitative analysis **18**

B.1 Reward model qualification ability 18

B.2 Qualitative analysis of rule-based verifiers 19

C Limitations and Future Work **19**

D The Use of Large Language Models(LLM) **20**

A Experiments

A.1 Experimental setup

Category	Hyperparameter	Value
Data	Train file	OpenMathReasoning
	Max prompt length	1024
	Max response length	8192
	Filter overlong prompts	True
Actor Model	Base model 1	`Qwen3-4B-Base`
	LR	1×10^{-6}
	KL loss coefficient β	0
	Entropy loss	0
	Use dynamic batch size	True
Rollout	Rollout engine	vllm
	GPU mem utilization	0.6
	Train rollout n	8
	Temperature	1.0
Reward	Rule Based	`Math_Verify`
	Reward Model Based	`AceMath-RM-7B`
Trainer	Mini Batch size	128
	Full Batch size	512 (4 step off-policy)
	Critic Warmup	0
	GPUs/node	4
	Nodes	8
	Total epochs	20
	Clip Ratio	(0.2, 0.28)

Table 5: Key hyperparameters used for GRPO training on OpenMathReasoning (Moshkov et al., 2025) in the verl (Sheng et al., 2025) framework for the Qwen-4B-Base.

Category	Hyperparameter	Value
Data	Train file	OPENMATHREASONING
	Max prompt length	1024
	Max response length	4096
	Filter overlong prompts	True
Actor Model	Base model 1	`OctoThinker-8B`
	LR	1×10^{-6}
	KL loss coefficient β	0.001
	Entropy loss	0
	Use dynamic batch size	True
Rollout	Rollout engine	vllm
	GPU mem utilization	0.6
	Train rollout n	16
	Temperature	1.0
Reward	Rule Based	`Math_Verify`
	Reward Model Based	`AceMath-RM-7B`
Trainer	Mini Batch size	128
	Full Batch size	512 (4 step off-policy)
	Critic Warmup	0
	GPUs/node	4
	Nodes	8
	Total epochs	20
	Clip Ratio	(0.2, 0.28)

Table 6: Key hyperparameters used for GRPO training on OPENMATHREASONING (Moshkov et al., 2025) in the verl (Sheng et al., 2025) framework for the OctoThinker-8B.

HERO hyper-parameters. For hybrid reward training for both Qwen-4B-Base and OctoThinker-8B-Base, we set the range parameters α and β depending on the task type. For verifiable tasks, we adopt a tighter setting $\alpha = \beta = 0.05$ to exploit the high precision of rule-based verifiers while minimizing noise. For mixed and hard-to-verify tasks, where the reward model contributes more substantially to supervision, we relax the range to $\alpha = \beta = 0.1$ to provide richer feedback. For variance-aware reweighting, we fix the weighting bounds as $w_{\min} = 0.4$ and $w_{\max} = 3.0$, with a steepness parameter $k = 6$ in the logistic weighting function. These values ensure that trivial prompts are down-weighted, while highly uncertain prompts—where reward-model scores vary widely—receive stronger emphasis without destabilizing training.

Training hyper-parameters. ables 5 and 6 provide an overview of the hyperparameter configurations used in our GRPO training runs with Qwen3-4B-Base and OctoThinker-8B. The tables cover settings across data preparation, actor model optimization, rollout generation, reward specification, and trainer configuration. They highlight the consistent use of OPENMATHREASONING as the training corpus, the integration of both rule-based and reward-model signals, and the adoption of scalable rollout and training strategies within the verl framework. Together, these summaries document the experimental setup and ensure reproducibility across different backbone models. In addition, we employ the HuggingFace `math_verify` library to provide standardized rule-based verification of responses against ground-truth answers, which guarantees consistency in supervision across all experiments.

A.2 Data preparation

A.2.1 Supervised Fine-Tuning dataset preparation

We found that initiating RL training directly from the base model often resulted in instability, particularly in the absence of a cold start. For instance, the qwen3-4b-base model frequently produced mixed-language outputs and generated irrelevant content during the early stages of training. Similarly, the octothinker base model demonstrated multi-turn behavior, leading to highly variable response lengths. To mitigate these issues and enhance the stability of RL training, we first conducted two epochs of cold-start supervised fine-tuning (SFT) before beginning RL. To avoid unintentional distillation from more capable models, we used the base model itself to generate responses. These outputs were then filtered, retaining only samples that satisfied the following criteria: the response contained the correct final answer, was entirely in English, and did not exhibit any unstop issues. For cold start training, we ultimately used only 2,000 SFT samples.

A.2.2 Training data filter from OpenMathReasoning

In this paper, we focus on reasoning questions that have extractable answers. To this end, we exclusively utilize data from the OpenMathReasoning dataset, selecting only those examples where the problem_type is set to has_answer_extracted. From the CoT split, we extracted 40k examples. For each example, we generated solutions and extracted the predicted answers, which were then verified using math_verifier (verl). We randomly sampled 2k examples that passed the verifier to serve as verifiable training data, and another 2k examples that failed verification as hard-to-verify training samples. These two sets were combined to create a mixed training dataset for reinforcement learning (RL) training. We use math_verifier (verl) to filter all the samples

A.2.3 Hard-to-verify evaluation benchmark from TextBookReasoning

GPT-4o filter prompt for TextBookReasoning.

```
"I am looking for math questions that are suitable for evaluating a math model. Please
     help me select questions that meet the following criteria:

1. The question must be clear and unambiguous.

2. The question must have a specific, factual, and answerable solution (not open-ended or
     subjective).

3. The question must NOT require a proof or explanation of reasoning.

4. The question must NOT be a statement; it should be a direct question.

For each question I provide, please respond with:
- \"Conclusion: Suitable\" in the end if the question meets all the criteria above.

- \"Conclusion: Not Suitable\"

if the question does not meet the criteria, and briefly explain why."
```

Figure 3: GPT-4o filter prompt for TextBookReasoning.

To construct a more challenging and reliable benchmark for hard-to-verify tasks, we employ the **TextBookReasoning** benchmark. The following criteria were used to filter and refine the dataset for the evaluation:

1. **Pass-through Math Verification Filter**
   The initial step in filtering was to ensure that the answers in the dataset did not pass the

Prompt Template for hard-to-verify tasks evaluation via GPT-4o.

```
User: ### Question: {question}

### Ground Truth Answer: {ground_truth}

### Student Answer: {student_answer}

For the above question, please verify if the student's answer is equivalent to the ground
     truth answer.
Do not solve the question by yourself; just check if the student's answer is equivalent to
      the ground truth answer.
If the student's answer is correct, output "Final Decision: Yes". If the student's answer
     is incorrect, output "Final Decision: No".

Assistant:
```

Figure 4: Prompt Template for hard-to-verify tasks evaluation via GPT-4o.

`math_verify` check, ensuring that the questions and answers involved a certain level of complexity or ambiguity that would make them challenging for standard verifiers.

2. **Llama 3.3_70B Instruct Model for Natural Reasoning**
   The dataset was further refined by using the `Llama 3.3_70b_instruct` model to answer natural reasoning prompts. Only the prompts for which Llama could not provide an answer were kept for further evaluation. This step ensured that the dataset included questions that required more advanced reasoning abilities, beyond the capabilities of standard models.
3. **GPT-4 as the Final Filter**
   Finally, GPT-4 was used to filter out questions that still met the criteria of being complex and hard-to-verify. GPT-4's ability to handle nuanced reasoning ensured that only the most challenging prompts remained. The prompt is shown as Figure 3

This process ultimately resulted in a refined set of approximately **750** prompts suitable for hard-to-verify task evaluation.

Prompt Template for Hard-to-Verify Tasks Evaluation The evaluation of student answers to these prompts is based on the following template, which uses GPT-4 to compare the student's answer against the ground truth:

Math Question Selection Criteria The following prompt was used to select math questions suitable for evaluating a math model. The criteria for question selection are outlined below:

A.2.4 HARD-TO-VERIFY PROMPT

We set the hard-to-verify evaluation prompt as shown in Figure 4. This template is designed to assess whether a student's response matches the reference answer without re-solving the question. By explicitly instructing GPT-4o to perform equivalence checking rather than problem solving, the protocol minimizes leakage of additional reasoning and focuses purely on correctness judgment. The structured format, including the question, ground truth, and student answer, ensures consistency across evaluations and reduces prompt sensitivity, making it suitable for benchmarking performance on hard-to-verify tasks.

A.3 MORE EXPERIMENTS

Naively combining rule-based rewards and reward signals from reward modeling does not perform well. A direct integration of rule-based verification and reward signals from reward modeling, without proper structural alignment, often disrupts the stability of training. As shown in Table 7, when the weight of the rule-based reward is varied ($\alpha = 0.1$, 0.5, and 0.9), the combined reward performance remains suboptimal, with scores ranging from 55.9 to 58.7 for verifiable tasks and 60.2 to 61.4 for hard-to-verify tasks. Specifically, when the continuous signals from the reward model are naively combined with binary correctness checks, the resulting reward can become noisy or misaligned with the intended notion of correctness. Without explicitly constraining the continuous scores within the rigid framework of the verifier's correctness criteria, reward-model outputs can be distorted by imperfections in the model, diminishing both interpretability and precision in the feedback. Moreover, the lack of a safeguard to differentiate true positives from noisy results can lead the model to exploit unintended patterns, which may not align with human expectations. As a result, an unrefined fusion of these two reward signals can dilute the benefits of both approaches, destabilizing the learning process.

Table 7: α represents the weight of the rule-based reward.

Methods	Verifiable	Hard-to-verify
Reward combine (α=0.1)	57.6	60.2
Reward combine (α=0.5)	58.7	61.4
Reward combine (α=0.9)	55.9	60.4
HERO (Ours)	62.0	73.2

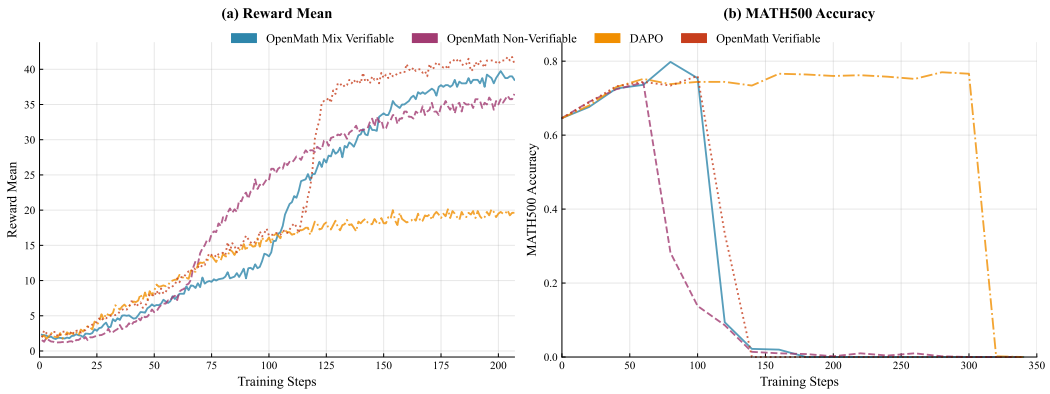

Figure 5: Reward model qualification ability on mixed groups: (a) distribution of AUROC scores, (b) AUROC box plot, (c) cumulative distribution of AUROC, and (d) AUROC performance categories.

Reward models hack faster on hard-to-verify samples. Since the reward model (RM) is trained on outcome-based verifiable samples (Liu et al., 2024), it is important to examine its behavior across datasets with varying levels of verifiability. We evaluate four datasets: DAPO (Yu et al., 2025), which is easy to verify; OpenMath Verifiable, which passes the `math_verifier`; OpenMath Non-Verifiable, which is harder to verify; and OpenMath Mix Verifiable, which combines both. As shown in Figure 5, the RM rapidly increases the reward mean across all datasets, with the sharpest gains on OpenMath Non-Verifiable and OpenMath Mix Verifiable. For example, on Non-Verifiable data, the reward mean climbs steeply from below 5 to over 30 within the first 100 training steps, and peaks above 40 by step 150. However, MATH500 accuracy collapses shortly after, dropping from around 0.75 at step 50 to below 0.2 by step 100, and effectively to zero by step 150. A similar trend appears on Mix Verifiable: accuracy initially rises to about 0.8 at step 100 but then crashes to nearly zero by step 150, despite the reward mean continuing to rise steadily past 35. In contrast, OpenMath Verifiable shows slower but steadier progress: rewards grow more gradually, and accuracy improves to about 0.8 by step 120 before stabilizing without collapse. DAPO also exhibits stable optimization, with accuracy consistently around 0.75–0.78 as rewards increase moderately. These results highlight a clear mismatch: rapid reward gains on hard-to-verify tasks are not evidence of genuine reasoning improvement, but rather reward hacking that leads to catastrophic accuracy collapse. This illustrates

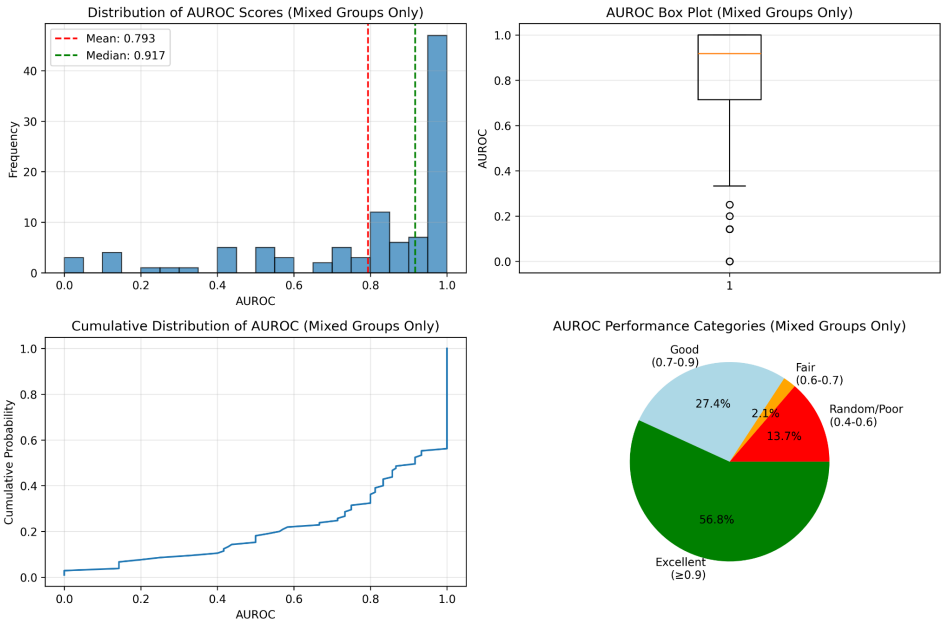

Figure 6: Reward model qualification ability on mixed groups: (a) distribution of AUROC scores, (b) AUROC box plot, (c) cumulative distribution of AUROC, and (d) AUROC performance categories.

s

the brittleness of relying solely on dense reward models and motivates hybrid reward frameworks that combine verifier-anchored reliability with the nuance of dense signals.

The proposed method does not rely on large reward models. A natural question is whether stronger supervision requires scaling up the reward model itself. To test this, we compare HERO with AceMath-RM-7B against a much larger variant, AceMath-RM-72B. As shown in Table 8, the larger reward model does not provide meaningful gains: while it slightly improves performance on verifiable tasks (62.8 vs. 62.0), it underperforms on hard-to-verify tasks (71.4 vs. 73.2). This result suggests that the benefits of our framework stem primarily from hybrid design—stratified normalization and variance-aware weighting—rather than from simply scaling the reward model. In practice, this means HERO can achieve strong results with compact reward models, offering better efficiency and deployability without sacrificing accuracy.

Table 8: Impact of reward model size: a larger RM (72B) provides no remarkable gain over the smaller RM (7B).

Reward model	Verifiable	Hard-to-verify
AceMath-RM-7B	62.0	73.2
AceMath-RM-72B	62.8	71.4

B Qualitative Analysis

B.1 Reward Model Qualification Ability

To better understand the reliability of reward-model supervision, we analyze its ability to approximate the verifier signal as a binary classification task. We random take all the rollout from one step (the 250 for the verifiable samples training) during the training. Specifically, we treat the reward model's raw scores as logits and the verifier's outputs as ground-truth binary labels, then compute AUROC statistics to measure discriminative power.

Figure 6 shows four complementary views. The histogram (top-left) reveals a strong skew toward high AUROC values, with a mean of 0.79 and median of 0.92, indicating that the reward model often ranks correct responses above incorrect ones. The box plot (top-right) highlights robustness but also exposes several low outliers where the model fails to separate classes. The cumulative dis-

Ground truth	Model Prediction	math.py	math_verify.py(verl)	Math_verify library	o3
$f(x) = 2x$	`\boxed {f(x) = 2x}`	✗	✓	✓	✓
$(6, 3), (9, 3), (9, 5), (54, 5)$	`\boxed {(6,3)}, \boxed {(9,3)}, \boxed {(9,5)}, \boxed {(54,5)}`	✗	✓	✓	✓
$(0, 1, 1), (0, -1, -1), (1, 0, 1), (-1, 0, -1), (1, 1, 0), (-1, -1, 0), \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right), \left(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right), \ldots$	`Final Answer: \boxed {(1,1,0)}, \boxed {(-1,-1,0)}, \boxed {(1/\sqrt {3},1/\sqrt {3},1/\sqrt {3})}, \boxed {(-1/\sqrt {3},-1/\sqrt {3},-1/\sqrt {3})},`	✗	✗	✗	✓
$10, 11, 12, 13, 14, -2, -1, 0, 1, 2$	`Final Answer: \boxed {-2,-1,0,1,2} and \boxed {10,11,12,13,14}`	✗	✓	✗	✓
$(1, 7, 103, 105),\ (3, 5, 101, 107)$	`Final Answer: Two possible lists are \boxed {(3,5,101,107)} and \boxed {(1,7,103,105)}`	✗	✓	✗	✓
$f(x) = ax + b$ (where b is an arbitrary integer, and a is an arbitrary positive integer with mho(a)=0)	`Final Answer: \boxed {f(n)=cn+d}, where c has no prime factors > 10^{100} and d is any integer`	✗	✓	✗	✓

Table 9: Examples demonstrating agreement between different math verification tools.

tribution (bottom-left) confirms that roughly 80% of groups achieve AUROC above 0.7. Finally, the performance categorization (bottom-right) shows that 56.8% of groups reach "excellent" AUROC (≥ 0.9), while only 13.7% fall into the "random/poor" range (0.4–0.6).

These results suggest that although the reward model is not perfect, it provides reliable ranking signals in the majority of cases. Importantly, this supports the use of dense reward signals to refine learning within verifier-defined groups: while the verifier anchors correctness, the reward model adds discriminative power that helps differentiate among responses of varying quality. The presence of failure cases further justifies our hybrid framework, which uses stratified normalization to bound reward-model signals within verifier groups, ensuring stability even when AUROC is low.

B.2 Qualitative analysis of rule-based verifiers

Table 9 highlights representative behaviors of rule-based and model-based verifiers. `math.py` is overly strict, failing on minor formatting variations such as boxing or punctuation (Rows 1–2), while `math_verify.py` improves recall through normalization. The `Math-Verify` library handles simple surface mismatches but struggles with structural differences like disjoint ranges or multiple valid tuples (Rows 4–5). In contrast, o3 is the most permissive: it credits partially correct sets (Row 3) and parametric families with renamed symbols (Row 6), which increases coverage but risks over-crediting. These cases illustrate the precision–recall trade-off: rule-based verifiers enforce exact symbolic correctness but miss semantically equivalent or partially correct answers, whereas model judges offer flexibility at the cost of reliability. This motivates our hybrid design: HERO anchors dense reward signals to rule-based correctness, ensuring robustness to format variance, while leveraging model- or RM-derived scores to provide graded feedback on harder cases involving subsets, orderings, or parametric equivalence.

C Limitations and Future Work

While HERO demonstrates clear advantages over RM-only and verifier-only training, several limitations remain. First, the method depends on the availability and reliability of rule-based verifiers: when these are brittle or domain-mismatched, the partitioning into correctness groups may be biased, weakening the benefits of stratified normalization. Second, because the reward model is trained primarily on outcome-based, verifiable data, it can become miscalibrated on harder, non-verifiable formats, and although our framework constrains its scores, residual bias or spurious correlations may still be exploited. Third, HERO introduces sensitivity to hyperparameters such as (α, β) and the weighting slope k, and increases training overhead due to concurrent verifier and RM calls. Fi-

nally, evaluation on non-verifiable tasks often relies on LLM-as-judge protocols, which introduce prompt sensitivity and annotation noise. Future work will focus on improving verifier coverage with hybrid symbolic–learned approaches, incorporating process-level supervision to capture reasoning quality beyond final answers, and developing adaptive range and weighting schemes that calibrate dense signals online. These directions can further strengthen the stability and generality of hybrid reward frameworks for reasoning.

D THE USE OF LARGE LANGUAGE MODELS(LLM)

In our project, we use LLM for writing polishing.