VerifyBench: Benchmarking Reference-based Reward Systems for Large Language Models

Anonymous ACL submission

Abstract

Large reasoning models such as OpenAI o1 and DeepSeek-R1 have demonstrated remarkable performance in complex reasoning tasks. A critical component of their training is the incorporation of reference-based reward systems within reinforcement learning (RL), where model outputs are evaluated against ground truth references. However, existing reward benchmarks focus on preference comparisons between responses rather than evaluating verification against ground truth references, leaving a critical gap in our ability to evaluate verification systems used in reasoning model training. In this paper, we introduce VerifyBench and its challenging variant VerifyBench-Hard, two benchmarks specifically designed to assess reference-based reward systems. These benchmarks are constructed through meticulous data collection and curation, followed by careful human annotation to ensure high quality. Our comprehensive evaluation reveals that while larger model-based verifiers show promise on standard cases, all current systems demonstrate substantial room for improvement on challenging instances. Through systematic analysis of performance patterns across reasoning tasks and error categories, we provide insights for advancing reference-based reward systems. These benchmarks establish a standardized framework for improving verification accuracy, ultimately enhancing reasoning capabilities in models trained via RL.

1 Introduction

In recent years, large language models (LLMs) have exhibited remarkable capabilities, significantly assisting humans across diverse practical domains (DeepSeek-AI et al., 2025b; Grattafiori et al., 2024; Yang et al., 2025). Reinforcement learning from human feedback (RLHF) has been crucial to this progress, with reward models playing a central role by evaluating and scoring model-generated responses to guide training. This approach has led to the development of numerous benchmarks (Lambert et al., 2024; Liu et al., 2024; Zhou et al., 2025) for systematic reward model evaluation, focusing primarily on pairwise preference judgments between competing responses.

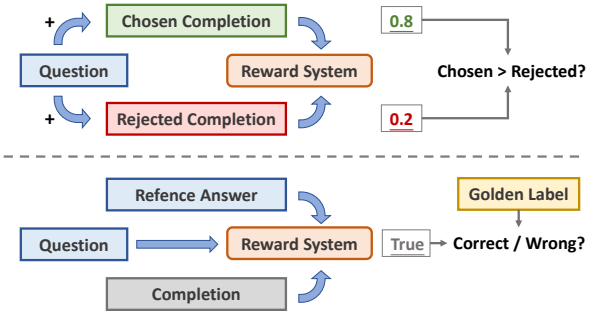

Figure 1: The core distinction between VerifyBench and existing reward benchmarks (Lambert et al., 2024; Liu et al., 2024) is illustrated as follows. **Upper panel:** Existing reward benchmarks assess the accuracy of a reward system by comparing the ranking of two completions for the same question. **Lower panel:** In contrast, our proposed VerifyBench evaluates the accuracy of a reward system by determining the correctness of a single completion using a reference answer.

The emergence of specialized large reasoning models (LRMs) (DeepSeek-AI et al., 2025a; Qwen Team, 2024; Kimi Team et al., 2025) such as OpenAI's o1(OpenAI, 2024) and DeepSeek-R1(DeepSeek-AI et al., 2025a) has fundamentally changed this landscape. These models achieve unprecedented performance on reasoning tasks through specialized reinforcement learning techniques that differ from standard RLHF approaches. A key distinction in training methodologies for LRMs is their reliance on reference-based reward systems, where rewards are assigned based on alignment between model-generated responses and authoritative reference answers. This approach has been implemented variously across leading models, with Deepseek-R1 (DeepSeek-AI et al., 2025a) em-

ploys a rule-based reward to prevent reward hacking, whereas models like Seed1.5-Thinking (Seed et al., 2025) adopt model-based reward systems to generate more precise and robust signals.

Despite the widespread adoption of reference-based reward systems in training state-of-the-art reasoning models, a significant gap exists in our ability to evaluate these systems systematically. Current benchmarks focus almost exclusively on preference-based evaluation, assessing rewards on their ability to rank competing responses correctly. This approach fails to capture the requirements of reference-based verification, where responses must be judged against objective ground truths rather than relative preferences. The absence of dedicated benchmarks for reference-based reward systems has limited researchers' ability to assess, compare, and improve their verification methodologies effectively, potentially impeding progress in reasoning model development.

To address this critical gap, we introduce VerifyBench, a benchmark specifically designed to evaluate the accuracy of reference-based reward systems. VerifyBench differs fundamentally from existing reward benchmarks by focusing on absolute correctness judgments rather than relative preference assessments. While traditional benchmarks ask reward models to determine which of two responses is better, VerifyBench challenges systems to verify whether a single response correctly aligns with a reference answer, more accurately reflecting the actual use case in reasoning model training.

In this paper, we present VerifyBench, a benchmark specifically designed to evaluate the accuracy of reference-based reward systems. To create VerifyBench, we curated a diverse collection of instructions paired with reference answers sourced from existing open datasets. Responses to these instructions were generated by multiple open-source and proprietary LLMs. The correctness of each response was assessed using both automated model judgments and human evaluations. Each instance in VerifyBench was verified by at least two human annotators to ensure label consistency and reliability, thereby producing a high-quality benchmark for the evaluation of reward systems.

Recognizing the need to differentiate between various verification techniques and to push the boundaries of current capabilities, we further developed VerifyBench-Hard, a more challenging variant of our benchmark. This dataset focuses on contentious cases where leading models produce highly conflicting judgments, providing a more stringent test for reward system accuracy. VerifyBench-Hard samples were carefully selected based on disagreement patterns among high-performing models, then subjected to thorough human annotation to ensure label quality.

Our contributions are summarized as follows:

1. To better reflect realistic reinforcement learning (RL) scenarios for reasoning models, we construct VerifyBench, a benchmark derived from existing models and datasets, to provide an objective evaluation of the accuracy of reference-based reward systems.

2. We further develop VerifyBench-Hard, a more challenging benchmark curated from cases exhibiting high disagreement among multiple models. This dataset contains a larger proportion of difficult-to-verify samples, highlighting substantial potential for improvement in current models.

3. We conduct a comprehensive empirical analysis of model performance on both VerifyBench and VerifyBench-Hard, offering actionable insights to advance the accuracy of reference-based reward systems and enhance RL training in reasoning tasks.

2 Preliminaries

Reference-free Reward Models In reinforcement learning (RL) for large language models (LLMs), the reward model plays a crucial role by approximating real-world reward signals associated with model-generated outputs. A typical reward model takes as input a user's query q along with the corresponding LLM-generated response r, and produces a reward signal, formally defined as:

$$r = R_\varphi(q, r) \tag{1}$$

where q represents the user's query, r denotes the response generated by the LLM, and φ encapsulates either the learned parameters of the reward model or the heuristic criteria used to evaluate the quality of the response given q and r.

Evaluation of Reference-free Reward Models Generally, reward models produce scalar outputs whose scales can vary significantly across different implementations, complicating direct numerical comparisons. Consequently, current benchmarks evaluate reward models using a pairwise

comparative approach. Formally, given a dataset D comprising tuples (q, r_w, r_l), where q represents a user's query, and r_w and r_l denote two candidate responses with r_w considered superior to r_l, the accuracy of a reward model is quantified as the proportion of instances in which the model correctly assigns a higher score to r_w than to r_l. Mathematically, this accuracy metric is defined as:

$$\text{Acc} = \frac{1}{|D|} \sum_{(q,r_w,r_l)\in D} \mathbb{I}\left[R_\varphi(q, r_w) > R_\varphi(q, r_l)\right] \tag{2}$$

where $\mathbb{I}(\cdot)$ is the indicator function, and R_φ denotes the reward model parameterized by φ.

Reference-based Reward Models With the emergence of advanced reasoning models such as DeepSeek-R1, reference-based reward systems have been integrated into reinforcement learning (RL) frameworks for large reasoning models (LRMs). These models require training on extensive datasets, which typically include authoritative reference answers. Consequently, the reward assignment task shifts towards evaluating the alignment between the model-generated outputs and their corresponding reference answers. Formally, this reward calculation can be expressed as:

$$r = R_\varphi(q, gt, r) \tag{3}$$

where q denotes the user-issued query, gt denotes the ground-truth reference answer, r represents the model-generated response, and φ encapsulates either the learned parameters of the reward model or the established evaluation criteria used to assess the alignment among q, gt, and r.

Evaluation of Reference-based Reward Models In this paper, we propose a reference-based reward benchmark designed to systematically evaluate reward models within reinforcement learning (RL) frameworks for large reasoning models (LRMs). Unlike traditional reward evaluation benchmarks, which rely on pairwise comparisons, our approach leverages explicit reference answers to directly assess the correctness of individual model-generated responses. Concretely, given a dataset D consisting of instances (q, gt, r, y), where q denotes the user-issued query, gt represents the ground-truth reference answer, r is the model-generated response, and y is the binary correctness label assigned to the response, we evaluate the reward model by measuring its accuracy in correctly predicting these labels. Formally, the accuracy metric is defined as:

$$\text{Acc} = \frac{1}{|D|} \sum_{(q,gt,r,y)\in D} \mathbb{I}\left[\mathbb{E}(R_\varphi(q, gt, r)) = y\right] \tag{4}$$

where $R_\varphi(q, gt, r)$ denotes the reward model parameterized by φ or defined by heuristic verification rules, producing predictions indicative of the correctness of response r relative to the provided reference answer gt. The function $\mathbb{E}(\cdot)$ represents an operation (e.g., thresholding or discretization) mapping continuous reward scores into discrete correctness predictions suitable for direct comparison with the ground-truth labels y.

3 Benchmark Construction

In this paper, we introduce two benchmarks, VerifyBench and VerifyBench-Hard, to evaluate reference-based reward systems. The VerifyBench benchmark is designed to reflect naturally distributed data, whereas VerifyBench-Hard comprises samples exhibiting high levels of disagreement among models, thereby assessing a model's ability to provide reliable judgments in ambiguous or challenging scenarios.

3.1 Construction of VerifyBench

Query Curation To emulate realistic reinforcement learning (RL) scenarios involving reference-based reward systems, we curate a comprehensive collection of open-source reasoning problems paired with corresponding reference answers. These problems encompass three primary categories, general reasoning, logical reasoning and mathematical reasoning, and are aggregated from 41 distinct sources. A complete list of these data sources is provided in Appendix B.

Answer Type Labeling To comprehensively evaluate model performance across diverse answer formats, we define four canonical answer types: numerical values, algebraic expressions, multiple-choice selections, and free-form strings. Utilizing a general-purpose LLM Llama-3.3-70B-Instruct (Grattafiori et al., 2024), we performed automatic answer-type classification with a prompt(Appendix C.1). Questions that fall outside these categories, such as proof-based or open-ended prompts, were excluded from further analysis. Following classification, we randomly sampled 2,000 instances per answer type, resulting in a final candidate pool of 8,000 questions.

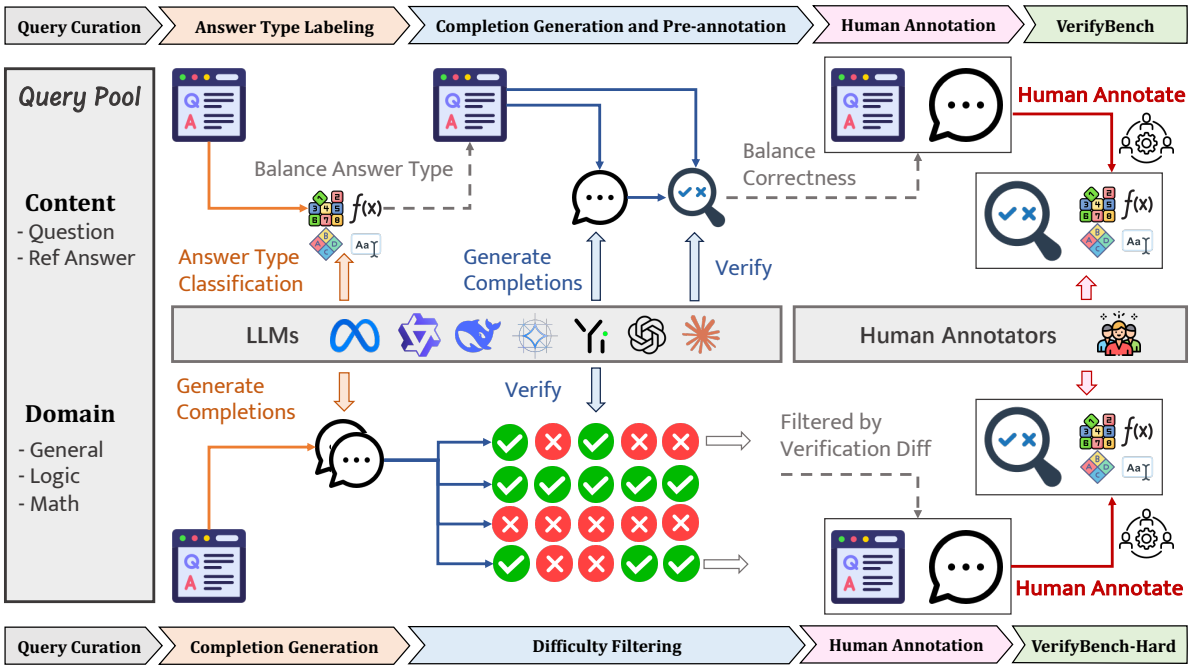

Figure 2: Overview of the benchmark construction process. The upper section outlines the pipeline used to construct VerifyBench, whereas the lower section details the pipeline for VerifyBench-Hard. The components highlighted by black boxes denote the final entries included in the benchmark.

Completion Generation and Pre-annotation We subsequently employed 22 widely used open-source and proprietary models (see Appendix E) to generate single-shot completions for the curated set of 8,000 questions, resulting in a total of 176,000 completions. To assign initial correctness labels, we utilized Llama-3.3-70B-Instruct (Grattafiori et al., 2024) within a prompt-based judgment framework. For each question, we randomly selected four completions, two labeled as correct and two labeled as incorrect by the model, and retained them for subsequent human annotation.

Human Annotation We conducted human annotation for the aforementioned questions and their associated completions. The annotation procedure comprised two primary tasks: (1) identifying the most appropriate answer type for each question based on its formulation and corresponding ground-truth answer, and (2) evaluating the correctness of each of the four completions. Each question was independently annotated by at least two annotators. If their annotations were consistent, the labeling was finalized; otherwise, a third annotator resolved disagreements to ensure consistency and finalize the labels.

Benchmark Construction Following human annotation, we identified notable biases in the models' predictions regarding both answer types and completion correctness, leading to imbalanced data distributions. To mitigate this issue, we performed controlled downsampling to ensure uniform category-level representation and balanced correctness labels. Specifically, we retained 250 questions per answer type, resulting in a total of 1,000 questions. Each question is paired with exactly two completions, one correct and one incorrect. The resulting dataset, VerifyBench, thus comprises 2,000 well-balanced question-answer-completion-correctness tuples. Detailed statistics for VerifyBench are provided in Table 1.

3.2 Construction of VerifyBench-Hard

To construct VerifyBench-Hard, we employed a specialized data generation pipeline consisting of the following key steps:

Completion Generation To construct the dataset, we first generated single-turn completions for the queries described in Section 3.1 using a collection of 18 open-source models. Due to the substantial volume of generations and the associated computational costs, closed-source models were excluded from this stage. In total, we produced approximately 1.45 million completions.

Statistics for VerifyBench and VerifyBench-Hard		
Statistics	**VerifyBench**	**VerifyBench-Hard**
# of unique questions	1000	945
# of unique completions	2000	1000
# of correct completions	1000	291
# of wrong completions	1000	709
Statistics of Answer Type		
# of Numeric Values	500	252
# of Expressions	500	88
# of Multi-choice	500	430
# of String	500	230
Statistics of Domain		
# of General Reasoing	404	303
# of Logic Reasoning	498	315
# of Math Reasoning	1098	382

Table 1: Benchmark statistics of VerifyBench and VerifyBench-Hard.

Difficulty Filtering Next, we employed five top-performing large models on VerifyBench (Llama-3.3-70B-Instruct (Grattafiori et al., 2024), Llama-4-Scout-17B-16E-Instruct (Meta AI, 2025), Qwen2.5-72B-Instruct (Qwen et al., 2025), Qwen3-30B-A3B, and Qwen3-32B (Yang et al., 2025)), which span a diverse range of model sizes and architectures, to evaluate the correctness of the generated completions. Based on their judgments, we identified question-answer-completion tuples exhibiting model disagreement, specifically those for which two models' assessments diverged from the other three. To ensure balanced and comprehensive representation, we applied stratified sampling across data domains and sources, ultimately selecting 2,000 examples for human annotation.

Human Annotation We subsequently subjected the selected samples to human annotation, focusing on two key aspects: identifying the answer type and determining the correctness of each completion. Each instance was annotated independently by at least two annotators. In cases where both annotators agreed, the annotation was finalized; when disagreement occurred, a third annotator was consulted to resolve the conflict.

Benchmark Construction Following human annotation, we excluded samples identified as unsuitable for inclusion in our benchmark. This filtering resulted in a final set of 1,000 question-answer-completion-correctness tuples. In contrast to VerifyBench, which enforces a balanced structure with one correct and one incorrect completion per question, VerifyBench-Hard is derived through natural sampling. We observed that larger models are more likely to erroneously accept incorrect answers as correct, resulting in a natural skew towards incorrect completions within the dataset. Detailed statistics for VerifyBench-Hard are provided in Table 1.

4 Evaluation Results

This section presents the evaluation results and analyses of our proposed benchmark. Section 4.1 reports the primary evaluation outcomes. In Section 4.2, we investigate the impact of reference answers on the verification process. Section 4.3 provides a comparative analysis between our benchmark and existing reward benchmarks, as well as the performance of several general-purpose reward models on VerifyBench and VerifyBench-Hard.

4.1 Overall Performance

We evaluate the performance of various verification approaches on both VerifyBench and VerifyBench-Hard. For rule-based baselines, we adopt the widely used *math-verify* (Kydlíček, 2025) method. In the LLM-as-a-judge setting, we prompt LLMs to perform verification; detailed prompt templates are provided in Appendix C.2. Our evaluation yields several key findings and insights:

Existing models perform well on VerifyBench: The primary objective in constructing VerifyBench is to establish a benchmark for the objective evaluation of reference-based reward systems. To this end, we designed the dataset with a balanced distribution across diverse domains and answer types, pairing each question with both a correct and an incorrect completion. This structure facilitates a rigorous and fair assessment of reward model performance. Notably, state-of-the-art LLMs already demonstrate strong performance on this benchmark: GPT-4o-mini achieves an average accuracy of 92.85%, while Qwen3-32B reaches 95.8%, highlighting the high reliability of LLMs as verifiers in this context.

VerifyBench-Hard is challenging: To more effectively differentiate the performance of various models, we constructed VerifyBench-Hard by selecting cases in which multiple LLMs exhibited substantial disagreement in their verification outputs. Evaluation results demonstrate that model performance on VerifyBench-Hard is significantly lower than on VerifyBench. The highest accuracy achieved is 72.4%, representing a 20% decrease compared to performance on VerifyBench. This

Model/Method	VerifyBench					VerifyBench-Hard				
	Num	Exp	MC	Str	AVG	Num	Exp	MC	Str	AVG
rule-based functions										
math-verify	83.60	72.00	19.40	8.60	45.90	**76.19**	**82.95**	8.37	10.43	32.50
LLM-as-a-judge										
OpenAI/gpt-4o-2024-11-20	94.80	90.20	96.80	90.80	93.15	71.43	65.91	75.35	71.30	**72.60**
OpenAI/gpt-4o-mini	95.80	89.80	95.80	90.00	92.85	69.05	72.73	74.19	72.17	72.30
meta-llama/Llama-4-Scout-17B-16E-Instruct	94.20	86.80	89.80	89.25	90.01	48.02	39.77	46.98	55.22	48.50
meta-llama/Llama-3.3-70B-Instruct	88.80	77.80	88.40	78.00	83.25	54.37	45.45	60.70	47.39	54.70
meta-llama/Llama-3.1-8B-Instruct	72.20	70.60	77.00	72.40	73.05	51.19	35.23	45.12	33.91	43.20
meta-llama/Llama-3.2-3B-Instruct	65.80	63.60	56.80	57.60	60.95	33.33	28.41	38.84	27.39	33.90
meta-llama/Llama-3.2-1B-Instruct	44.40	41.00	37.60	53.60	44.15	22.22	13.64	29.07	27.39	25.60
Qwen/Qwen3-235B-A22B	96.40	92.40	97.00	89.40	93.80	70.24	72.73	70.93	69.57	70.60
Qwen/Qwen3-30B-A3B	96.60	91.80	97.40	90.20	94.00	64.68	70.45	69.53	56.52	65.40
Qwen/Qwen2.5-72B-Instruct	95.40	89.80	95.60	88.60	92.35	70.63	60.23	61.40	56.09	62.40
Qwen/Qwen3-32B	**97.60**	**94.00**	**99.00**	**92.60**	**95.80**	69.05	81.82	68.14	**77.83**	71.80
Qwen/Qwen3-8B	96.40	93.00	96.20	90.40	94.00	68.65	78.41	73.02	66.52	70.90
Qwen/Qwen3-4B	95.20	91.60	93.60	87.60	92.00	71.03	62.50	**75.58**	71.74	72.40
Qwen/Qwen3-1.7B	83.20	81.00	80.60	79.60	81.10	48.81	38.64	60.93	41.74	51.50
microsoft/phi-4	92.60	86.40	93.00	85.40	89.35	59.52	57.95	54.19	57.39	56.60
01-ai/Yi-1.5-9B-Chat-16K	90.40	87.40	88.00	85.00	87.70	65.48	63.64	62.09	54.78	61.40
google/gemma-3-1b-it	55.40	56.20	43.00	56.00	52.65	32.14	19.32	33.72	40.87	33.70

Table 2: Overall performance(%) of VerifyBench and VerifyBench-Hard. **Num** stands for Numeric Values, **Exp** stands for Expressions, **MC** stands for Multi-choice and **Str** stands for String.

performance gap underscores substantial opportunities for improvement in the precise verification capabilities of current LLMs.

Small models still have potential for development: In real-world reinforcement learning scenarios, the inference efficiency of the reward system critically impacts the overall training speed. Since verification typically involves generative computation, its computational cost is often comparable to that of generating rollouts. This raises an important practical question: *Can smaller models effectively serve as verifiers?* Our results indicate that small-parameter models perform significantly worse on VerifyBench. For instance, Qwen3-1.7B achieves an accuracy of 81.10%, whereas Llama-3.2-3B-Instruct lags behind at 60.95%. These findings suggest that enhancing the verification capabilities of compact models constitutes a promising and necessary research direction.

4.2 Reference-answers Play an Important Role in Verification

The benchmark proposed in this work fundamentally differs from existing reward benchmarks by explicitly incorporating reference answers, thereby aligning more closely with the training setups of contemporary reasoning LLMs. To isolate the impact of reference answers on verification performance, we conduct an ablation study in which models are evaluated without reference answers provided in the prompt; the prompt format used is detailed in Appendix C.3.

Model	VerifyBench	
	w/ Ref	w/o Ref
Llama-4-Scout-17B-16E-Instruct	90.01	73.95-16.06
Llama-3.3-70B-Instruct	83.25	75.00-8.25
Llama-3.1-8B-Instruct	73.05	64.10-8.95
Llama-3.2-3B-Instruct	60.95	55.35-5.60
Llama-3.2-1B-Instruct	44.15	44.50+0.35
Qwen3-235B-A22B	93.80	80.15-13.65
Qwen3-30B-A3B	94.00	78.25-15.75
Qwen2.5-72B-Instruct	92.35	77.30-15.05
Qwen3-32B	95.80	78.90-16.90
Qwen3-8B	94.00	75.75-18.25
Qwen3-4B	92.00	74.40-17.60
Qwen3-1.7B	81.10	62.10-19.00

Table 3: Evaluation results(%) about how including the reference answer in the prompt influences the performance of LLM-as-a-judge.

Experimental results, summarized in Table 3, reveal a performance degradation of approximately 5–18% when reference answers are excluded. These findings underscore the crucial role of reference answers in reasoning-oriented RL, suggesting they provide a more reliable and informative supervision signal during reward modeling.

Model	RM-Bench	Reward Bench	VerifyBench				
			Num	Exp	MC	Str	AVG
General Reward Models							
Skywork/Skywork-Reward-Llama-3.1-8B	72.29	**93.33**	60.80	64.80	59.60	68.80	63.48
internlm/internlm2-20b-reward	72.06	92.16	65.60	64.80	61.20	70.00	65.40
Ray2333/GRM-llama3-8B-sftreg	71.33	88.50	64.80	58.40	58.80	67.60	62.40
internlm/internlm2-7b-reward	72.42	90.02	73.20	68.00	66.80	70.40	69.60
Domain-specific Reward Models							
Qwen/Qwen2.5-Math-RM-72B	**76.28**	82.11	**83.60**	**79.20**	**73.60**	**75.60**	**78.00**
Qwen/Qwen2-Math-RM-72B	62.61	75.54	79.20	78.40	73.20	72.80	75.90

Table 4: The performance(%) of existing reward models on VerifyBench without access to reference answers, as well as a comparison with existing reward benchmarks.

4.3 Performance of Vanilla Reward Models

To enable a more comprehensive evaluation of existing reward models, we additionally assessed several reference-free reward models and benchmarked their performance on conventional pairwise reward evaluation datasets for comparison. Notably, each question in our proposed VerifyBench consists of one correct and one incorrect completion, enabling straightforward reformulation into standard pairwise evaluation instances. The experimental results are summarized in Table 4.

Our experimental results show that VerifyBench introduces a level of challenge comparable to existing reward benchmarks, with the absence of reference answers. Reference-free reward models achieve sub-80% accuracy on VerifyBench, highlighting its difficulty. Furthermore, domain-specific reward models exhibit inferior performance on general reward benchmarks compared to VerifyBench, validating the benchmark's design objectives.

5 Analysis

5.1 Error Analysis

To gain deeper insights from VerifyBench, we introduce a more fine-grained taxonomy for each answer type and analyze model performance across these subcategories. This detailed analysis helps identify specific reasoning tasks or answer formats where models are particularly error-prone. We subdivide the Numeric Values category into 8 subtypes, Expressions into 5 subtypes, Multi-choice into 3 subtypes, and String into 2 subtypes. Table 5 presents the comparative performance of different models across these detailed categories.

We further analyze subcategories within each major category that exhibit below-average accuracy.

Answer Type	Q32B	g4o	L70B	L3B
Numeric Values	97.60	94.80	88.80	65.80
Integer	96.88-0.72	96.88	93.75	65.62-0.18
Constant	96.88-0.72	95.31	92.19	70.31
Float Number	98.39	96.77	90.32	61.29-4.51
Radical	98.39	95.16	87.10-1.70	75.81
Complex Number	96.77-0.83	96.77	85.48-3.32	59.68-6.12
Angle	96.77-0.83	96.77	93.55	66.13
Non-decimal number	100	93.55-1.25	88.71-0.09	64.52-1.28
Multiple Values	96.77-0.83	87.10-7.70	79.03-9.77	62.90-2.90
Expressions	94.00	90.20	77.80	63.60
Algebraic formula	91.54-2.46	84.62-5.58	67.69-10.11	56.92-6.68
Equation	87.50-6.5	78.12-12.08	70.31-7.49	60.94-2.66
Interval	96.09	94.53	82.81	60.94-2.66
Set	98.00	98.00	78.00	60.00-3.60
Matrix	96.09	94.53	86.72	75.78
Multi-choice	99.00	96.80	88.40	56.80
Single-choice	99.39	98.17	92.07	59.15
Multiple-choice	98.21-0.79	94.05-2.75	77.98-10.42	49.40-7.40
Finite state selection	99.40	98.21	95.24	61.90
String	92.60	90.80	78.00	57.6
Specific	93.60	93.31	81.69	59.01
Semantic	90.38-2.22	85.26-5.54	69.87-8.13	54.49-3.11

Table 5: Model performance(%) across the fine-grained taxonomy on VerifyBench. **Q32B** stands for Qwen3-32B, **g4o** stands for gpt-4o-2024-11-20, **L70B** stands for Llama-3.3-70B-Instruct and **L3B** stands for Llama-3.2-3B-Instruct.

The following error-prone subtypes are identified as the most frequent sources of incorrect judgments:

- **Numeric Values:** Complex numbers and answers containing multiple numerical values;
- **Expressions:** Algebraic formulas and equations;
- **Multi-choice:** Multi-answer choice problems;
- **String:** Strings requiring semantic consistency verification.

We analyzed the samples most prone to errors and identified a common underlying issue: models

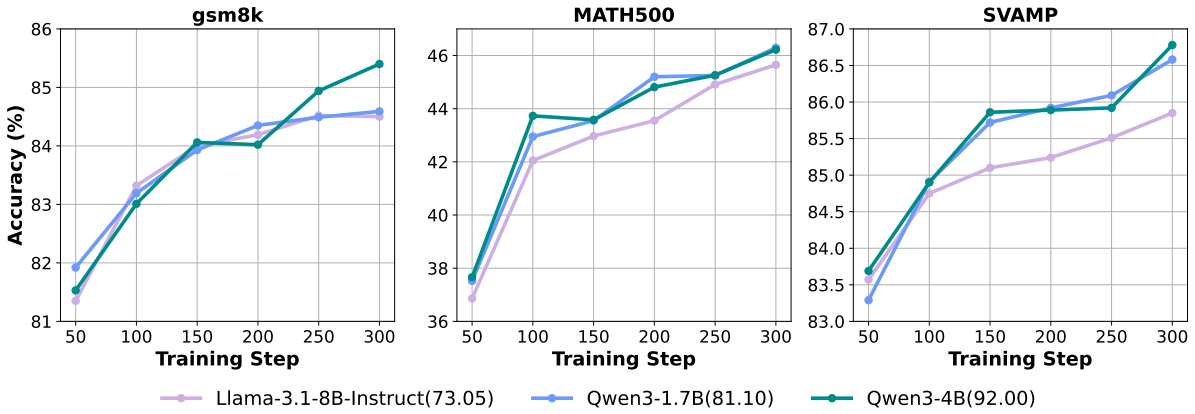

Figure 3: The performance(%) of RFT across different LLM judges which have various performance on VerifyBench.

frequently fail to fully comprehend the question or clearly recognize the intended objective. For instance, in cases involving multi-value answers, the ordering of values is typically irrelevant. However, if the sequence of values in the model's output differs from the golden answer, models often incorrectly classify the response as erroneous. Similarly, errors within the Expressions category, particularly involving algebraic formulas and equations, predominantly result from inadequate mathematical comprehension. Specifically, when a model outputs an unsimplified expression, superficial textual discrepancies compared to the ground-truth answer can be significant. Rather than evaluating whether the expression is mathematically equivalent upon simplification, models prematurely deem the output incorrect, thereby leading to verification failures.

5.2 Correlation Analysis

We constructed VerifyBench and VerifyBench-Hard with the goal of improving the effectiveness of RL for reasoning models by enhancing the accuracy of reference-based reward systems. To evaluate the practical utility of our benchmark, we performed a correlation analysis between VerifyBench and real-world RL performance.

In our experiments, we applied rejection sampling to implement reference-based reward systems. For each question in the GSM8K and MATH training sets, we generated 64 candidate completions using Qwen2.5-Math-7B-Instruct (Yang et al., 2024) with a sampling temperature of 0.7. These responses were subsequently filtered by three verifier models with varying performance levels on VerifyBench: Llama-3.1-8B-Instruct, Qwen3-4B, and Qwen3-1.7B. Only completions consistently verified as correct were retained to form the SFT training data. We conducted independent SFT training runs accordingly, with full hyperparameter configurations provided in the Appendix D.

The resulting models were evaluated on multiple mathematical reasoning benchmarks, including GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021; Lightman et al., 2023), and SVAMP (Patel et al., 2021). As shown in Figure 3, on GSM8K, MATH500, and SVAMP, using Qwen3-4B, a verifier that achieves higher accuracy on VerifyBench, consistently outperforms Llama-3.1-8B-Instruct, a lower-scoring verifier, under the same training steps. This highlights the strong alignment between VerifyBench and practical usage. Our benchmarks serve as reliable tools to guide the development of reward systems, leading to more effective training and improved model performance.

6 Conclusion

In this paper, we present two dedicated benchmarks, VerifyBench and VerifyBench-Hard, to evaluate reference-based reward systems in the context of reasoning-focused reinforcement learning. These benchmarks were built with high-quality, carefully curated data and extensive human annotation. Our results reveal that current verifiers, especially those with smaller model sizes, still face considerable challenges in accurately assessing reasoning completions. Through detailed analysis, we provide insights into the strengths and weaknesses of existing systems and highlight opportunities for improvement. The proposed benchmarks fill a critical gap in the evaluation landscape, offering a principled foundation for understanding verifier accuracy and guiding the development of more effective reasoning models trained via reinforcement learning.

Limitations

Limited Data Domain In this paper, we utilize datasets exclusively from general reasoning, logical reasoning, and mathematical reasoning, which do not cover the full spectrum of reasoning types, such as commonsense reasoning. Consequently, our test sets may not adequately evaluate the quality of reward systems in out-of-domain scenarios.

Bias from Human Annotation The construction of VerifyBench and VerifyBench-Hard involved extensive human annotation. Although all annotators were trained and a double-checking strategy was employed, it remains challenging to entirely eliminate annotation bias inherent in manual labeling processes.

Reward Hacking Could Not Be Identified While our experiments demonstrate that rule-based reward systems perform worse than model-based approaches on both VerifyBench and VerifyBench-Hard, a critical issue remains unaddressed: reward hacking. Future research should focus on detecting and evaluating reward hacking phenomena.

Proof Problems Excluded During annotation, our guidelines explicitly excluded proof-based questions. We believe such problems require more specialized verification methods, such as formal languages like Lean4. Consequently, proof questions are not included in this study, and their verification remains an open research challenge.

Binary Scoring System The benchmark constructed in this paper employs a binary scoring system, where each completion is labeled as either correct or incorrect. However, real-world scenarios often involve more nuanced cases, such as partially correct reasoning processes or correct solutions to subproblems. Introducing a more fine-grained evaluation scheme could better capture these complexities.

Ethical Considerations

All human annotators involved in constructing the benchmarks were assigned reasonable workloads and fairly compensated for their contributions.

Our annotation process involves minimal subjective preference. Human annotators performed the verification tasks following our detailed instructions. The content of the annotations does not involve ethical issues and poses no ethical risks.

References

01 AI, Alex Young, Bei Chen, Chao Li, Chengen Huang, et al. 2025. Yi: Open foundation models by 01.ai. *arXiv preprint*.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, et al. 2020. Abductive commonsense reasoning. *arXiv preprint*.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences. *arXiv preprint*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, et al. 2025b. Deepseek-v3 technical report. *arXiv preprint*.

Meng Fang, Xiangpeng Wan, Fei Lu, Fei Xing, and Kai Zou. 2024. Mathodyssey: Benchmarking mathematical problem-solving skills in large language models using odyssey math data. *arXiv preprint*.

Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios N. Angelopoulos, et al. 2024. How to evaluate reward models for rlhf. *arXiv preprint*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, et al. 2025. Gemma 3 technical report. *arXiv preprint*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. 2024. The llama 3 herd of models. *arXiv preprint*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, et al. 2024. Folio: Natural language reasoning with first-order logic. *arXiv preprint*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint*.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, et al. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, et al. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint*.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, et al. 2025. Big-bench extra hard. *arXiv preprint*.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. 2023. Boardgameqa: A dataset for natural language reasoning with contradictory information. *arXiv preprint*.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, et al. 2025. Process reward models that think. *arXiv preprint*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, et al. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, et al. 2023. Efficient memory management for large language model serving with pagedattention. *arXiv preprint*.

Hynek Kydlíček. 2025. Math-verify: Math verification library.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, L. J. Miranda, Bill Yuchen Lin, et al. 2024. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint*.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, et al. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, et al. 2023. Let's verify step by step. *arXiv preprint*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint*.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. 2024. Rm-bench: Benchmarking reward models of language models with subtlety and style. *arXiv preprint*.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, et al. 2025. Inference-time scaling for generalist reward modeling. *arXiv preprint*.

Meta AI. 2025. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2021. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint*.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, et al. 2022. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint*.

OpenAI. 2024. Introducing openai o1. https://openai.com/index/introducing-openai-o1-preview/.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint*.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, et al. 2024. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. *arXiv preprint*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint*.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. 2024. Multi-logieval: Towards evaluating multi-step logical reasoning ability of large language models. *arXiv preprint*.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. 2025. Qwen2.5 technical report. *arXiv preprint*.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown. https://qwenlm.github.io/blog/qwq-32b-preview/.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, et al. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint*.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *arXiv preprint*.

ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, et al. 2025. Seed1.5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint*.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, et al. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint*.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. Clutrr: A diagnostic benchmark for inductive reasoning from text. *arXiv preprint*.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, et al. 2022. Learning to summarize from human feedback. *arXiv preprint*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint*.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, et al. 2023. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, et al. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, et al. 2023. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint*.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, et al. 2025. Tablebench: A comprehensive and complex benchmark for table question answering. *arXiv preprint*.

Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, et al. 2025. A minimalist approach to llm reasoning: From rejection sampling to reinforce. *arXiv preprint*.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2025a. Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *IEEE Transactions on Knowledge and Data Engineering*, 37(4):1620–1634.

Yuhui Xu, Hanze Dong, Lei Wang, Caiming Xiong, and Junnan Li. 2025b. Reward models identify consistency, not causality. *arXiv preprint*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. 2025. Qwen3 technical report. *arXiv preprint*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, et al. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint*.

Nathan Young, Qiming Bao, Joshua Bensemann, and Michael Witbrock. 2022. Abductionrules: Training transformers to explain unexpected inputs. *arXiv preprint*.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. *arXiv preprint*.

Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, et al. 2025. Naturalreasoning: Reasoning in the wild with 2.8m challenging questions. *arXiv preprint*.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *arXiv preprint*.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint*.

Zhang, Yifan. 2024. Stackmathqa: A curated collection of 2 million mathematical questions and answers sourced from stack exchange. https://huggingface.co/datasets/math-ai/StackMathQA.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, et al. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint*.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, et al. 2021. Ar-lsat: Investigating analytical reasoning of text. *arXiv preprint*.

Enyu Zhou, Guodong Zheng, Binghai Wang, Zhiheng Xi, Shihan Dou, et al. 2025. Rmb: Comprehensively benchmarking reward models in llm alignment. *arXiv preprint*.

A Related Works

A.1 Reward Systems for Reinforcement Learning

Early reward models (RMs) (Christiano et al., 2023; Stiennon et al., 2022; Ouyang et al., 2022), trained to predict human preference rankings, typically treat the entire response as the evaluation unit. However, such outcome-level RMs lack insight into intermediate reasoning steps, making step-level error correction infeasible (Xu et al., 2025b). To address this limitation, process-level RMs (Lightman et al., 2023; Setlur et al., 2024) have been introduced to assign scores at each reasoning step, thereby providing stepwise feedback. Despite their effectiveness, process-level RMs require extensive manual step-level annotations, resulting in exponential increases in data collection costs and training complexity (Khalifa et al., 2025).

Building on these advances, DeepSeek-R1 (DeepSeek-AI et al., 2025a) employs rule-based reward functions that leverage predefined, maintainable rules for pattern matching and logical validation, offering simplicity and efficiency. However, as task diversity expands, the manual creation of such rules faces significant challenges related to scalability and coverage, ultimately limiting its applicability in open-ended generation scenarios.

More recently, DeepSeek-GRM (Liu et al., 2025) and ThinkPRM (Khalifa et al., 2025) have explored integrating reasoning capabilities into RMs by developing generative reward models (GRMs). GRMs reformulate the scoring task as a token-generation problem: before outputting a numerical score, the model first generates a chain-of-thought (CoT) (Wei et al., 2023) that explicates its evaluation criteria and rationale. This approach not only bridges the interpretability gap between black-box discriminative models and brittle rule-based systems but also substantially enhances test-time scaling capabilities.

A.2 Evaluation of Reward Systems

There are two primary approaches to evaluating reward systems. The first approach employs standardized benchmarks that objectively assess reward system effectiveness by designing diverse tasks and datasets (Frick et al., 2024). The second approach examines the performance of reward systems when integrated directly into downstream optimization loops, such as Best-of-N selection (Nakano et al., 2022) or rejection sampling fine-tuning (Zelikman et al., 2024; Xiong et al., 2025), to measure their impact on generation quality and alignment.

Reward system benchmarks can be further categorized into outcome-level (Liu et al., 2024; Lambert et al., 2024) and process-level (Lightman et al., 2023; Zheng et al., 2024) suites. In constructing these benchmarks, researchers generate multiple responses to the same prompt by varying model architectures or hyperparameters. During the manual annotation phase, outcome-level benchmarks require annotators to compare or assign multi-point scores to complete responses, emphasizing overall preference. In contrast, process-level benchmarks provide fine-grained gold verdicts by requiring step-by-step correctness labels for each reasoning step.

Beyond benchmark-based evaluation, practical applications of reward systems serve as another common assessment method. In the Best-of-N (BoN) paradigm, WebGPT (Nakano et al., 2022) introduced using a reward model to score N candidate answers and select the top-ranked response. Subsequent work has framed reward models as downstream rankers—for example, Self-Consistency in chain-of-thought models (Wang et al., 2023), where the reward model identifies the most coherent solution among candidates. Unlike Best-of-N, rejection sampling fine-tuning (RFT) (Zelikman et al., 2024; Xiong et al., 2025) samples multiple trajectories from the current policy, scores them using a reward model, and retains only the highest-scoring examples as silver supervision for further fine-tuning. This approach has proven particularly effective at bootstrapping reasoning capabilities without requiring full preference-learning pipelines.

B Data Source

Table 7 provides a comprehensive overview of all datasets used in constructing VerifyBench, detailing their respective licenses and the number of samples drawn from each. All data usage strictly complies with the terms and conditions stipulated by the original sources.

C Prompts

C.1 Prompt for Answer Type Classification

We present the prompt we used to generate answer types in Figure 4.

C.2 Prompt for LLM-as-a-judge

We present the prompt we used in LLM-as-a-judge evaluation with a reference answer in Figure 5.

C.3 Prompt for LLM-as-a-judge without Reference

We present the prompt we used in LLM-as-a-judge evaluation with a reference answer in Figure 6.

D Experimental Details

Training. For the rejection sampling fine-tuning experiments, we used Llama-3.1-8B (Grattafiori et al., 2024) as the base model for SFT. The learning rate was set to a constant value of 1e-5. Training was conducted using the Megatron-LM framework, with a global batch size of 256 and a context length of 4096. To accelerate training, we packed the training samples and trained for one epoch in total. All training experiments were conducted on 32 Ascend H910B-64G GPUs.

Evalution. For evaluation, we used the *vLLM*(Kwon et al., 2023) framework for inference. To reduce evaluation variance, we set the temperature to 0.7 and sampled each test example 16 times, then computed the average accuracy. All inference were conducted on 8 NVIDIA A100-80G.

E LLM Usage

We list all the LLMs we used to generate completions for curated question.

Series	Model
OpenAI	gpt-4o-2024-11-20 gpt-4o-mini
anthropic	claude-3.7-sonnet
deepseek-math	deepseek-math-7b-instruct (Shao et al., 2024) deepseek-math-7b-rl (Shao et al., 2024)
DeepSeek	DeepSeek-V3 (DeepSeek-AI et al., 2025b) DeepSeek-R1 (DeepSeek-AI et al., 2025a) DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI et al., 2025a) DeepSeek-R1-Distill-Qwen-32B (DeepSeek-AI et al., 2025a)
gemma-3	gemma-3-1b-it (Gemma Team et al., 2025) gemma-3-4b-it (Gemma Team et al., 2025) gemma-3-12b-it (Gemma Team et al., 2025)
Llama-3	Llama-3.3-70B-Instruct (Grattafiori et al., 2024) Llama-3-8B-Instruct (Grattafiori et al., 2024)
Qwen2.5	Qwen2.5-7B-Instruct (Qwen et al., 2025) Qwen2.5-72B-Instruct (Qwen et al., 2025)
Qwen2.5-Math	Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) Qwen2.5-Math-7B-Instruct (Yang et al., 2024) Qwen2.5-Math-72B-Instruct (Yang et al., 2024)
QwQ	QwQ-32B (Qwen Team, 2024)
Yi-1.5	Yi-1.5-9B-Chat-16K (AI et al., 2025) Yi-1.5-34B-Chat (AI et al., 2025)

Table 6: LLMs used in this paper.

F Examples of VerifyBench

We provide some examples of VerifyBench with four different answer types in Figure 7, 8, 9 and 10.

Domain	Source	License	# of Questions
general_reasoning	BBH (Suzgun et al., 2022)	MIT	4520
	BBEH (Kazemi et al., 2025)	Apache 2.0	6511
	MMLU_pro (Wang et al., 2024)	Apache 2.0	2000
	natural_reasoning (Yuan et al., 2025)	CC-BY-NC 4.0	10000
logic_reasoning	AbductionRules (Young et al., 2022)	MIT	1000
	anlg (Bhagavatula et al., 2020)	/	1000
	anli (Nie et al., 2020)	CC-BY-NC 4.0	1000
	ARLSAT (Zhong et al., 2021)	MIT	230
	bAbI15 (Weston et al., 2015)	/	1000
	bAbI16 (Weston et al., 2015)	/	1000
	BoardgameQA (Kazemi et al., 2023)	CC-BY-4.0	1000
	clutrr (Sinha et al., 2019)	CC-BY-NC 4.0	1000
	FOLIO (Han et al., 2024)	CC-BY-SA-4.0	134
	hellaswag (Zellers et al., 2019)	MIT	1000
	logicbenchBQA (Parmar et al., 2024)	MIT	1000
	logicbenchMCQA (Parmar et al., 2024)	MIT	1000
	LogiQA (Liu et al., 2020)	/	1000
	MultiLogiEval (Patel et al., 2024)	MIT	1000
	NeuLRabductive (Xu et al., 2025a)	/	1000
	NeuLRdeductive (Xu et al., 2025a)	/	1000
	NeuLRinductive (Xu et al., 2025a)	/	1000
	ProntoQA (Saparov and He, 2023)	Apache 2.0	500
	ProofWriter (Tafjord et al., 2021)	/	1000
	ReClor (Yu et al., 2020)	/	500
	tablebench (Wu et al., 2025)	Apache 2.0	886
math_reasoning	AIME24	MIT	30
	AIME25	MIT	30
	asdiv-a (Miao et al., 2021)	CC-BY-NC 4.0	1218
	Math Odyssey (Fang et al., 2024)	MIT	389
	GPQA_diamond (Rein et al., 2023)	MIT	198
	gsm8k (Cobbe et al., 2021)	MIT	1319
	math401 (Yuan et al., 2023)	/	392
	mathematics (Saxton et al., 2019)	Apache 2.0	3360
	MATH(Hendrycks et al., 2021)	MIT	5000
	OlympiadBench-EN (He et al., 2024)	MIT	675
	SVAMP (Patel et al., 2021)	MIT	1000
	NuminaMath-CoT (Li et al., 2024)	Apache 2.0	20000
	orca-math-word-problems (Mitra et al., 2024)	MIT	10000
	ArtOfProblemSolving	self-curated	7997
	stackmathqa (Zhang, Yifan, 2024)	CC-BY-4.0	10000
	DeepMath-103K-RL (He et al., 2025)	MIT	20000

Table 7: The datasets we used and the number of samples drawn from each, including the license information of these datasets.

Prompt for Answer Type Classification

You are a professional LLM evaluator and now you are tasked with identifying the type of answer that corresponds to a problem, here is what is required of you:
1. I will give you a problem and the corresponding solution, you need to analyze the problem carefully and understand what objective you need to get for that problem;

2. Please choose the most appropriate type for the target (i.e. final answer) to the problem, based on the objective of the give problem, the types available to you include:
A Numeric values
A1 Integers(whole numbers), e.g. 2, 351, etc.
A2 Numerical values containing constants, e.g. 2\pi, etc.
A3 Fractions/Float numbers/Proportions, e.g. \frac{1}{2}, 3.123, 1:2, etc.
A4 Numerical values containing radicals, e.g. \sqrt{3}
A5 Complex numbers, e.g. 3+4i
A6 Angles, e.g. 120^\circ
A7 Non-decimal numbers (not base 10 numeric values), e.g. 1234_8
A8 Multiple values with no order between them, e.g. 1,-2, 1 and 2, etc.
B Mathematical expressions
B1 Algebraic formulas, e.g. ax+b, etc.
B2 Equations or systems of equations, e.g. ax+b=0, etc.
B3 intervals, including concatenation and intersection of intervals, e.g. (1,2] (1,2) \cup (3,4), etc.
B4 A set, a collection of elements, e.g. {1,2,3}, (1,2,3), etc.B5 a matrix or vector, e.g. \begin{pmatrix} -7 \\ 16 \\ 5 \end{pmatrix}, (1, 2) etc.
C Multiple-choice questions
C1 Single-choice questions, choose one correct option from multiple choices
C2 Multiple choice, select multiple correct choices from multiple options
C3 Finite state selection, choose one correct state from a finite number of states, e.g. True/False, yes/no, valid/invalid, etc.
D String
D1 Specific natural language representation, more concerned with whether a particular word or expression is mentioned correctly or not
D2 Semantically consistent natural language representations, where the answer is difficult to be expressed in a single unified word, and there is a preference for comparing the semantic consistency of answers
E Other cases not categorized by the above categories

3. Please analyze carefully and output the serial numbers of the corresponding answer categories, you need to output your reasons for making such a classification. Your output should be in the format of a legal json in {"explanation": "xxx", "type": "xx"}, where 'explanation' is your reasoning, 'type' is the serial number of the final corresponding answer category, and you can output more than one serial number, separated by ‘,’.

4. You only need to care about the type of the final answer.
Question:{question}
Answer:{answer}

Figure 4: Prompt for answer type classification.

Prompt for llm-as-a-judge

Given the following math problem and the reference answer. Judge the correctness of the answers given later, with some ability to generalize and match the form and format of the answer results. The following specific requirements are followed when judging:

1. Judge only whether the final result of the reference answer and the answer to be judged agree; do not consider whether there are any errors in the process. Don't verify the correctness of the answer by yourself, please only refer to the reference answer for the correctness of the answer.
2. The reference answer and the answer to be judged only need to be essentially the same, ignoring irrelevant details such as units, symbols, whether or not to approximate, and the form of expression in the answer. The two answers are considered to be consistent if they are equivalently transformable.
3. All your analysis answer must be in English.
4. Please analyze the judged answer and try to compare it with the reference answer.
At the end of all analysis, give the result of the judgment on an extra line at the end of the answer in the form "Final Judgment: Yes/No".

Problem: {question}
Reference Answer: {answer}
Solution to be evaluated: {completion}

Figure 5: Prompt for LLM-as-a-judge evaluation.

Prompt for llm-as-a-judge without a Reference Answer

Given the following math problem, please judge the correctness of the answers given later. The following specific requirements are followed when judging:

1. All your analysis answer must be in English.
2. Please analyze the math problem and the answer and try to tell whether the given completion is a correct answer. At the end of all analysis, give the result of the judgment on an extra line at the end of the answer in the form ”Final Judgment: Yes/No”.

Figure 6: Prompt for LLM-as-a-judge evaluation without reference answers.

Question:

How many positive and negative integers is 12 a multiple of?

Reference Answer:

The number 12 is a multiple of $-12, -6, -4, -3, -2, -1, 1, 2, 3, 4, 6,$ and $12,$ for a total of $\boxed{12}$ integers.

Completion:

We want to find the number of positive and negative integers that divide 12.
The positive integers that divide 12 are the positive divisors of 12. The divisors of 12 are the integers that divide 12 without leaving a remainder.
We can find the divisors of 12 by listing pairs of integers that multiply to 12.
$1 \times 12 = 12$
$2 \times 6 = 12$
$3 \times 4 = 12$
So the positive divisors of 12 are $1, 2, 3, 4, 6, 12$. There are 6 positive divisors of 12.

The negative integers that divide 12 are the negative divisors of 12. These are the negative integers that divide 12 without leaving a remainder.
The negative divisors of 12 are $-1, -2, -3, -4, -6, -12$. There are 6 negative divisors of 12.

The total number of integers that divide 12 is 6 (positive divisors) $+ 6$ (negative divisors) $= 12$.
Alternatively, we can find the prime factorization of 12.
$12 = 2^2 \cdot 3^1$.
The number of positive divisors is $(2+1)(1+1) = 3 \cdot 2 = 6$.
The number of negative divisors is also 6.
Therefore, the total number of integers that divide 12 is $6 + 6 = 12$.

The positive integers that divide 12 are $1, 2, 3, 4, 6, 12$. There are 6.
The negative integers that divide 12 are $-1, -2, -3, -4, -6, -12$. There are 6.

The number of positive integers that divide 12 is 6.
The number of negative integers that divide 12 is 6.

Final Answer: The final answer is $\boxed{6}$

Golden Label: **False**

Figure 7: An data example from VerifyBench with answer type: **Numeric Values**.

Question:

Consider two lines parameterized by:
\[
\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} + t \begin{pmatrix} 2 \\ -3 \end{pmatrix}
\]
and
\[
\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix} + u \begin{pmatrix} 1 \\ -1 \end{pmatrix}
\]
Determine the coordinates of their intersection point.

Reference Answer:

For the first line, \[\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} + t \begin{pmatrix} 2 \\ -3 \end{pmatrix} = \begin{pmatrix} 1 + 2t \\ 2 - 3t \end{pmatrix} \] For the second line, \[\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix} + u \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 4 + u \\ 5 - u \end{pmatrix} \] Setting equal the parametric equations for \(x\) and \(y\): \[1 + 2t = 4 + u \quad \text{(1)} \] \[2 - 3t = 5 - u \quad \text{(2)} \] From equation (1), solve for \(u\): \[u = 1 + 2t - 4 = - 3 + 2t \quad \text{(3)} \] Substitute \(u\) from equation (3) into equation (2): \[2 - 3t = 5 - (-3 + 2t) \] \[2 - 3t = 8 - 2t \] \[-t = 6 \] \[t = -6 \] Substitute \(t = -6\) back into \(u\)'s equation: \[u = -3 + 2(-6) = -3 - 12 = -15 \] Plug \(t = -6\) into the parameterization for the first line: \[\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + 2(-6) \\ 2 - 3(-6) \end{pmatrix} = \begin{pmatrix} 1 - 12 \\ 2 + 18 \end{pmatrix} = \begin{pmatrix} -11 \\ 20 \end{pmatrix} \] So, the intersection point is \(\boxed{\begin{pmatrix} -11 \\ 20 \end{pmatrix}} \).

Completion:

Okay, so I have to find the intersection point of two lines given in parametric form. The first line is parameterized as:

\[
\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} + t \begin{pmatrix} 2 \\ -3 \end{pmatrix}
\]

<omitted>
\[
-x = 11 \implies x = -11
\]
\[
y = -(-11) + 9 = 20
\]

Thus, the coordinates of the intersection point are \(\boxed{(-11, 20)}\).

Golden Label: **True**

Figure 8: An data example from VerifyBench with answer type: **Expression**.

Question:

Given that point P lies on the circle $C: x^{2}-6x+y^{2}-6y+14=0$, and the line $AB: x+y-2=0$, then which of the following statements is true? A: Line AB intersects circle C. B: Line AB is disjoint from circle C. C: The distance from point P to line AB is greater than $\frac{1}{2}$. D: The distance from point P to line AB is less than 5.

Reference Answer:

To analyze the relationship between circle C and line AB, we first need to rewrite the equation of circle C in its standard form and calculate the distance from the center of C to line AB. 1. **Circle Equation Transformation**: Given equation of circle C is $x^{2}-6x+y^{2}-6y+14=0$. To rewrite it in standard form, we complete the square for x and y terms: - For x: $x^{2}-6x = (x-3)^{2} - 9$ - For y: $y^{2}-6y = (y-3)^{2} - 9$ Substituting these into the circle's equation, we get: $(x-3)^{2} + (y-3)^{2} - 9 - 9 + 14 = 0$ Simplifying, we find: $(x-3)^{2} + (y-3)^{2} = 4$ This shows that circle C has a center at $C(3,3)$ and a radius $r=2$. 2. **Distance from Center to Line**: The distance d from the center of the circle $(3,3)$ to line AB (given by $x+y-2=0$) can be calculated using the formula for the distance from a point to a line: $d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$ For line AB, $A=1$, $B=1$, $C=-2$, and substituting $(x_0, y_0) = (3,3)$, we get: $d = \frac{|3 + 3 - 2|}{\sqrt{1^2 + 1^2}} = \frac{4}{\sqrt{2}} = 2\sqrt{2}$ Since $2\sqrt{2} > r = 2$, line AB does not intersect circle C, making statement A false and B true. 3. **Analysis of Statements C and D**: - The maximum distance from a point on the circle to the line is $2\sqrt{2} + r = 2\sqrt{2} + 2$. - The minimum distance is $2\sqrt{2} - r = 2\sqrt{2} - 2$. Since the minimum distance from point P to line AB is $2\sqrt{2} - 2$, which is greater than $\frac{1}{2}$, statement C is true. And since the maximum distance is $2\sqrt{2} + 2$, which is less than 5, statement D is also true. Therefore, the correct choices are $\boxed{BCD}$.

Completion:

To analyze the problem, let's break it down step by step. --- ### Step 1: Rewrite the equation of the circle \(C \) The given equation of the circle is: \[x^2 - 6x + y^2 - 6y + 14 = 0. \] Complete the square for both \(x \) and \(y \): \[x^2 - 6x = (x - 3)^2 - 9, \quad y^2 - 6y = (y - 3)^2 - 9. \] Substitute these into the equation: \[(x - 3)^2 - 9 + (y - 3)^2 - 9 + 14 = 0. \] Simplify: \[(x - 3)^2 + (y - 3)^2 = 4. \] This is the equation of a circle with center \((3, 3) \) and radius \(2 \). --- ### Step 2: Analyze the line \(AB \) The equation of the line \(AB \) is: \[x + y - 2 = 0. \] Rewriting it: \[y = -x + 2. \] --- ### Step 3: Determine the relationship between the circle and the line To check whether the line \(AB \) intersects the circle \(C \), calculate the distance from the center of the circle \((3, 3) \) to the line \(AB \). The formula for the distance from a point \((x_1, y_1) \) to a line \(ax + by + c = 0 \) is: \[\text{Distance} = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}}. \] Here, the line \(AB \) is \(x + y - 2 = 0 \), so \(a = 1 \), \(b = 1 \), \(c = -2 \), and the point is \((3, 3) \). Substituting: \[\text{Distance} = \frac{|1(3) + 1(3) - 2|}{\sqrt{1^2 + 1^2}} = \frac{|3 + 3 - 2|}{\sqrt{2}} = \frac{|4|}{\sqrt{2}} = \frac{4}{\sqrt{2}} = 2\sqrt{2}. \] The radius of the circle is \(2 \). Since the distance from the center of the circle to the line (\(2\sqrt{2} \)) is greater than the radius (\(2 \)), the line \(AB \) does **not** intersect the circle. Therefore, **statement B** ("Line \(AB \) is disjoint from circle \(C \)") is true. --- ### Step 4: Analyze the distance from point \(P \) to line \(AB \) Point \(P \) lies on the circle \(C \), so its distance to the center \((3, 3) \) is exactly \(2 \) (the radius of the circle). The maximum possible distance from \(P \) to the line \(AB \) is the distance from the center \((3, 3) \) to the line \(AB \) plus the radius, which is: \[2\sqrt{2} + 2. \] The minimum possible distance from \(P \) to the line \(AB \) is the distance from the center \((3, 3) \) to the line \(AB \) minus the radius, which is: \[2\sqrt{2} - 2. \] Numerically: \[2\sqrt{2} \approx 2.828, \quad 2\sqrt{2} - 2 \approx 0.828, \quad 2\sqrt{2} + 2 \approx 4.828. \] Thus, the distance from \(P \) to the line \(AB \) is always greater than \(\frac{1}{2} \) and less than \(5 \). Therefore, both **statement C** and **statement D** are true. --- ### Final Answer: The true statements are: - **B**: Line \(AB \) is disjoint from circle \(C \). - **C**: The distance from point \(P \) to line \(AB \) is greater than \(\frac{1}{2} \). - **D**: The distance from point \(P \) to line \(AB \) is less than \(5 \).

Golden Label: **True**

Figure 9: An data example from VerifyBench with answer type: **Multi-choice**.

Question:

Vlad numbered the squares of a chessboard (from 1 to 64) in some order. Gosha did the same with his chessboard, but the numbering turned out to be different. Can it be that the squares on Vlad's board are connected by a knight's move if and only if the squares on Gosha's board with the same numbers are connected by a king's move?

Reference Answer:

To solve this problem, we will perform a proof by contradiction. 1. Assume that it is possible that the cells on Vlad's chessboard connected by a knight's move are precisely those same-numbered cells connected by a king's move on Gosha's chessboard. 2. Let's examine a corner cell on the chessboard, for example, cell \(A1 \) on Vlad's chessboard. A knight starting in cell \(A1 \) has only two possible moves, which are to cells \(B3 \) and \(C2 \). 3. Consider the connectivity due to the king's move. A king positioned at any corner cell like \(A1 \) can move to the three adjacent cells (in Vlad's case) or up to eight adjacent cells in the middle. Therefore, a king's movement from \(A1 \) can go to cells like \(B1 \), \(B2 \), and \(A2 \). 4. Let's map this to Gosha's cell numbering. Assume, - Cell \(A1 \) (Vlad) = Cell \(X \) (Gosha) - Cell \(B3 \) and Cell \(C2 \) (Vlad) = Cells \(Y \) and \(Z \) (Gosha), respectively. Here, cells \(Y \) and \(Z \) are where the knight can move directly from cell \(X \) in Vlad's grid. 5. According to our assumption, in Gosha's grid, the corresponding cell moves (cells \(X \) to \(Y \) and \(Z \)) must be achievable by a king's move. However, for any corner cell (like \(X \)), a king only has three adjacent cells and thus clearly cannot reach two cells (\(Y \), \(Z \)) in one move if those cells are not adjacent. 6. This contradiction shows our initial assumption is incorrect, because it is impossible for cells connected by a knight's move to be the same as cells connected by the king's move. Conclusion: \[\boxed{\text{No, it cannot happen.}} \]

Completion:

No, it is not possible for the squares on Vlad's chessboard to be connected by a knight's move if and only if the squares on Gosha's chessboard with the same numbers are connected by a king's move.

Reasoning:

1. **Knight's Move Properties**:
- A knight's move on a chessboard is an "L"-shaped move: two squares in one direction and one square in a perpendicular direction (or vice versa).
- A knight's move always connects squares of opposite colors (e.g., from a white square to a black square or vice versa).

2. **King's Move Properties**:
- A king's move allows movement to any of the 8 surrounding squares (horizontally, vertically, or diagonally).
- A king's move can connect squares of the same color or opposite colors.

<omitted>

5. **Conclusion**:
- No matter how Vlad and Gosha number their chessboards, the condition that a knight's move on Vlad's board corresponds exactly to a king's move on Gosha's board cannot be satisfied. The fundamental differences in the movement rules of the knight and the king make this impossible.

Golden Label: **True**

Figure 10: An data example from VerifyBench with answer type: **String**.