CompassVerifier: A Unified and Robust Verifier for Large Language Models

Anonymous ACL submission

Abstract

Answer verification is crucial not only for evaluating large language models (LLMs) by matching their unstructured outputs against standard answers, but also serves as the reward model to guide LLM optimization. Most evaluation frameworks rely on regularized matching or employ general LLMs for answer verification, which demands extensive, repetitive customization for regex rules or evaluation prompts. Two fundamental limitations persist in current methodologies: 1) the absence of comprehensive benchmarks that systematically evaluate verification capabilities across different LLMs; and 2) the nascent stage of verifier development, where existing approaches lack both the robustness to handle complex edge cases and the generalizability across different domains. In this work, we develop **CompassVerifier**, an accurate and robust lightweight verifier model for evaluation and outcome reward. It demonstrates multi-domain competency spanning math, knowledge, and diverse reasoning tasks, with the capability to process various answer types, including multi-subproblems, formulas, and sequence answers, while effectively identifying abnormal/invalid responses. We introduce **VerifierBench** benchmark comprising model outputs collected from multiple data sources, augmented through manual analysis of meta error patterns to enhance CompassVerifier. We anticipate that CompassVerifier and VerifierBench will facilitate answer verification, evaluation protocols, and reinforcement learning research.

1 Introduction

Answer verification plays a critical role in the evaluation and training of large language models (LLMs), particularly for objective questions with verifiable answers (Achiam et al., 2023; Yang et al., 2024; Liu et al., 2024). At the evaluation level, it enables precise measurement of performance differences across models (Chang et al., 2024); at the training level, it serves as a quality check for self-improvement (Hosseini et al., 2024; Song et al., 2025). With the rapid development of reasoning models and reinforcement learning (RL), answer verification has further become a key component in constructing rule-based rewards, providing reliable feedback signals to directly guide model optimization and iteration (Guo et al., 2025; OpenAI, 2024c; Luong et al., 2024; Zhong et al., 2025).

Existing answer verification methods can be broadly categorized into two types. The first type relies on regularized string matching, such as extracting content following "`The answer is`" to compare with reference answers, or using tools like math-verify (huggingface, 2024) to check formula equivalence in mathematical tasks. The second type employs general LLMs for consistency judgment, where a specific prompt is designed to instruct the model to evaluate the alignment between *candidate* and *reference answers*. However, both approaches suffer from significant limitations: the former requires repetitive customization of matching rules for different tasks and is prone to verification failures due to extraction errors; the latter demands frequent prompt adjustments to accommodate diverse tasks, domains, and answer types, while also facing the risk of misjudgment caused by model hallucination. Meanwhile, there is still no challenging benchmark available to evaluate and distinguish the verification capabilities of different models, nor to guide the development and iteration of verifiers.

In this paper, we establish a systematic framework for evaluating and training answer verification systems. We first introduce **VerifierBench**, a challenging benchmark dataset for answer verification that aggregates numerous samples where rule-based methods frequently err or LLMs tend to produce incorrect judgments or hallucinations. We integrated over one million data samples through the OpenCompass (OC-Contributors, 2023) evaluation framework, encompassing responses from more than 50 models across 15 carefully selected datasets. Following large-scale data collection, each sample underwent a multi-stage filtering pipeline culminating in rigorous domain expert review and calibration. VerifierBench facilitates precise measurement of verification capabilities across diverse models, addressing complex scenarios where both rule-based matching and general models often fail, and offering manually analyzed summaries of prevalent error patterns.

We further present **CompassVerifier**, a series of lightweight yet robust and accurate verification models. The training data originates from three key sources: (1) The original training set from VerifierBench, which undergoes multi-model validation with simple, easily verifiable samples removed; (2) Formula-enhanced data, where we leverage the powerful DeepSeek-V3 model to generate numerous equivalent complex formulas with corresponding reasoning processes to improve formulaic answer evaluation; (3) Hallucination-specific data, where we systematically analyze failure patterns from human validation cases and synthesize targeted training samples to address common hallucination errors.

Our contributions are threefold:

- We propose **VerifierBench**, a novel and challenging benchmark meticulously designed for fine-grained evaluation of verification abilities.
- We develop **CompassVerifier**, a series of robust and efficient verification models enhanced through our three proposed techniques, achieving state-of-the-art performance across diverse domains and tasks.
- Through a systematic analysis of prevalent failure modes in LLM-based verification, including characteristic hallucination phenomena and error propagation, we derive actionable insights aimed at advancing the design and robustness of future verification systems.

2 Related Work

2.1 Answer Verification

Unlike traditional discriminative models with well-defined classification labels, the unstructured outputs of generative LLMs pose unique verification challenges (Cobbe et al., 2021). Current approaches to verifying LLM-generated answers can be broadly categorized into *outcome verification* and *process verification* (Kawabata and Sugawara, 2024; Zhang et al., 2025).

Outcome verification focuses on assessing the correctness of final answers, typically through string-based pattern matching (OC-Contributors, 2023; Gao et al., 2024; OpenAI, 2023). Common practice instructs LLMs to output answers in predefined formats for character-level comparison with ground truth. For formulaic answers, specialized tools like Math-Verify (huggingface, 2024) have been developed to handle equivalence checking. However, due to the inherent unpredictability of LLM outputs, such methods often suffer from matching failures or inaccuracies. Many studies thus employ general LLMs as verifiers via tailored prompts. While effective, both methods demand task-specific customization through either regex patterns or verified prompts, creating labor-intensive workflows. *Process verification*, requiring detection of reasoning errors in intermediate steps, has seen recent advances in both LLM-based verifiers and evaluation benchmarks (Lu et al., 2024; o1 Team, 2024; Lightman et al., 2023; Zheng et al., 2024; Zhou et al., 2024). However, process verifiers remain less frequently adopted in evaluations due to instability and high resource costs, and have not demonstrated substantially superior performance compared to outcome verifiers in RL.

We focus on scalable and robust outcome verification by developing a unified verifier that serves dual purposes: 1) as an evaluation model for benchmarking model performance, and 2) as a real-time reward model for RL training. By addressing the limitations of existing methods, such as ad-hoc prompt engineering and brittleness to output variations, CompassVerifier prioritizes efficiency, generalizability, and reliability across diverse tasks.

2.2 LLM-as-a-Judge

The comprehensive capabilities of LLMs enable them to serve as cost-effective alternatives to human experts in evaluation tasks, a concept known as "LLM-as-a-Judge" (Gu et al., 2024; Li et al., 2024a), which can be categorized into two approaches: *subjective judgment* and *objective judgment*.

Subjective judgment typically operates in scenarios without ground-truth answers, where LLMs score individual responses (Pointwise) (Zhu et al., 2025) or express preferences between paired responses (Pairwise) (Wang et al., 2024a). This requires the LLM to evaluate various aspects of responses, including usefulness, harmlessness, and

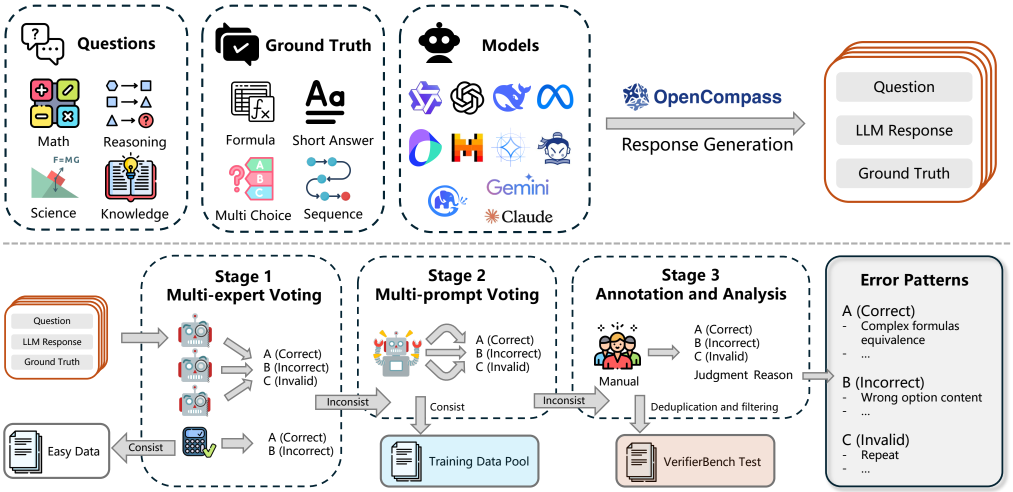


Figure 1: Overview of VerifierBench pipeline. Using OpenCompass (OC-Contributors, 2023), we collected more than 1 million LLM responses, applying multi-stage, multi-model verification with tool-assisted cleaning and filtering to create VerifierBench's test/base training sets and catalog common verification error patterns.

creativity, and even identify reasoning stepwise errors in the responses (Cao et al., 2024; Li et al., 2024c, 2023). Recent studies also employ RL and inference-time scaling like generative critiques, long-CoT, and multi-sampling voting for judgment, albeit with high computational costs (Liu et al., 2025; Shi and Jin, 2025). *objective judgment* is a more straightforward approach, requiring only the evaluation of response correctness against ground-truth. Beyond simple string matching, the prevalent method employs large-scale LLMs with carefully designed evaluation prompts for judgment. Recently, to enable smaller models to achieve comparable verification capabilities to large LLMs, Chen et al. (2025) proposes xVerify and its accompanying benchmark, which trains smaller verifier models by distilling GPT-4o's capabilities. Other concurrent studies have also focused on distilling verification capabilities from large models to smaller ones to achieve better cost-effectiveness (Ma et al., 2025; Su et al., 2025).

We claim that objective judgment with ground-truth has yet to reach maturity, lacking both challenging benchmarks to discriminate model abilities and robust unified models. To address these gaps, we are committed to developing VerifierBench to rigorously test different models' verification capabilities and CompassVerifier to provide the research community with an accurate evaluation tool.

3 VerifierBench

The primary challenge in verifier development lies in the lack of comprehensive benchmarks and rigorous evaluation methodologies. Large-scale commercial models are often preferred for answer-matching tasks due to the prevailing assumption of scaling laws. However, critical questions remain unanswered: 1) To what extent do answer matching and objective judgment tasks adhere to scaling laws? 2) How should we balance model performance against computational costs in verification?

To answer these questions, in this work, we present VerifierBench, a systematic framework for evaluating diverse models' judgment and verification capabilities. VerifierBench addresses this gap through: 1) Large-scale data collection for answer matching (3.1); 2) Multi-round validation involving multiple LLMs and human annotators (3.2); 3) Case analysis of typical error patterns to identify failure modes (3.3).

3.1 Data Collection

Answer verification, while not requiring sophisticated reasoning capabilities, demands authentic and diverse outputs from LLMs. To comprehensively gather such data, we employed the OpenCompass framework (OC-Contributors, 2023) to conduct large-scale evaluations across multiple models and datasets. Our systematic approach yielded more than 1,325,293 samples covering three key domains: knowledge, mathematics, and general reasoning. The collected data features:

- **Answer Type Diversity**: Multiple response formats including multiple-choice questions, formula-based answers, short texts, multi-subproblem items, and long-sequence responses.

- **Prompt Variability**: Input prompts covering few-shot, zero-shot, and dataset-specific formatting requirements.

- **Response Characteristics**: Model outputs ranging from short and long chain-of-thought (CoT) answers to direct responses and anomalous outputs (e.g., repetitions, truncations, refusals).

- **Diverse Model Coverage**: Comprehensive representation across commercial LLMs, open-source LLMs, and emerging large reasoning models (LRMs), spanning diverse model scales.

Formally, our collected data consists of triplets:

$$\mathcal{D} = \{(q_i, a_i^*, r_i^m)\}_{i=1}^N \tag{1}$$

where $q_i \in \mathcal{Q}$ represents the i-th question, $a_i^* \in \mathcal{A}$ denotes the corresponding golden answer, $r_i^m \in \mathcal{R}$ is the response generated by model $m \in \mathcal{M}$. The primary objective of VerifierBench construction is to augment these triplets with verification labels, resulting in verified quadruples:

$$\mathcal{D}_{\text{VerifierBench}} = \{(q_i, a_i^*, r_i^m, v_i)\}_{i=1}^N \tag{2}$$

where $v_i \in \{\text{Correct}, \text{Incorrect}, \text{Invalid}\}$ is the verification label indicating the correctness of r_i^m with respect to a_i^*. Notably, during data collection and curation, we identified numerous responses exhibiting abnormal or exceptional behaviors. These include abruptly truncated outputs, excessive repetition, and cases where models refused to answer due to ethical considerations or other constraints. We therefore categorize such instances as *invalid* responses to enable a more fine-grained evaluation.

3.2 Data Construction Pipeline

Our multi-stage verification pipeline, integrating LLMs, human annotators, and rule-based tools, efficiently identifies high-value training and testing samples from a large collected dataset.

Multi-Expert Voting. Initially, samples undergo direct verification (CoT reasoning) by Qwen2.5-Instruct models (7B, 14B, 32B). Samples with consensus are deemed trivial cases reliably handled by weaker models and are removed, offering minimal value. For mathematical domains (Math, GSM8K, and AIME datasets), we also incorporated Math-Verify (huggingface, 2024) as an additional expert verifier.

Multi-prompt Voting. Disputed samples advance to a second verification stage, where DeepSeek-V3 is employed with multiple prompts to generate diverse CoT reasoning paths. Consensus samples from this stage, representing moderately challenging instances, constitute our training pool. Our experiments revealed significant challenges in developing a universal verification prompt applicable across all datasets, evidenced by substantial residual disagreements after the second verification round. To address this, we implemented an additional verification phase for selected datasets, featuring domain-optimized prompts. For instance, the Chinese SimpleQA dataset required specially crafted Chinese-language prompts to achieve reliable verification outcomes.

Human Annotation and Analysis. The remaining disputed samples are human-annotated, with high-value cases primarily allocated to the test set. For the VerifierBench test set, we systematically excluded proof-based questions, open-ended problems, and numerical answers with ambiguous acceptability thresholds. These non-binary judgment cases, requiring specialized verification tools or domain expertise, are deferred to future work, ensuring VerifierBench focuses on clearly verifiable samples.

Identification of Flawed Samples. Human annotation also identified a distinct category: "flawed samples." Errors in these samples stem not from model deficiencies in problem-solving but from issues inherent to the questions (e.g., ambiguity, incorrect standard answers) or external factors (e.g., improper output truncation, generation of meaningless repetitive text, model refusal to answer). Such flawed samples, if not distinguished, can skew model capability assessment and hinder effective model iteration. These issues are often overlooked in traditional evaluation paradigms. Consequently, we explicitly label these samples as "Invalid" and integrate them into the VerifierBench test set. This approach enables a more granular, multi-dimensional, and realistic perspective for model performance verification.

3.3 Statistics and Analysis

We present the statistical characteristics of the Verifier test set across three dimensions: label categories (Table 4), problem domains (Table 5), and answer types (Table 6). After filtering and balancing, the dataset composition shows an approxi-

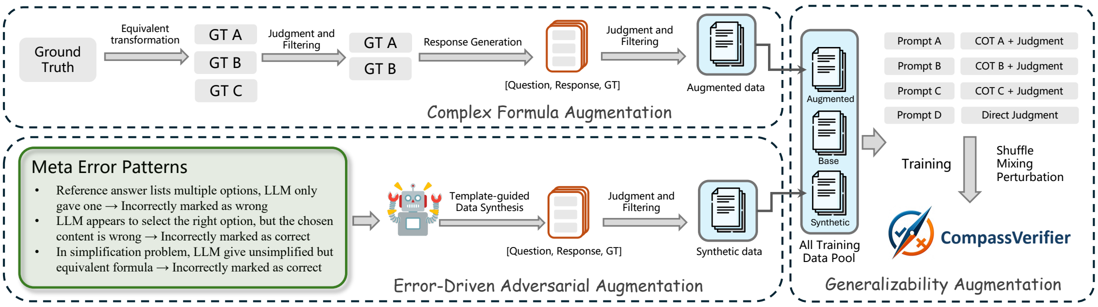


Figure 2: Overview of CompassVerifier training pipeline.

mate 4:6 ratio between Category A and B samples, with Category C representing about 5% of the total. Regarding problem domains, general reasoning, and mathematical reasoning constitute the majority, aligning with the current needs of reinforcement learning on large reasoning models. Classified by DeepSeek-V3, the answer types comprise seven categories: multiple-choice, numerical values, short answers, formulas, multi-subproblem, sequences, and binary answers. The detailed dataset sources are provided in Table 3.

4 CompassVerifier

CompassVerifier is designed to deliver efficient, high-performance, and robust answer verification. The system leverages filtered (question, reference answer, model response) triples from VerifierBench with golden judgments as training supervision. We also propose three key techniques to drive its performance: *Complex Formula Augmentation* enhances formula variants verification, *Error-Driven Adversarial Augmentation* fortifies against failures, and *Generalizability Augmentation* ensures cross-domain and cross-prompt applicability. Figure 2 shows the whole pipeline of training CompassVerifier. Details of the composition of the training Data in Appendix A.7.

4.1 Error-Driven Adversarial Augmentation

To address potential annotation inaccuracies in our filtered data (see Section 3.2), we employ a three-phase adversarial augmentation strategy.

Human-in-the-Loop Analysis. Domain experts manually verify 5,000 annotated samples, identify and document failure rationales such as LLM misunderstandings of task constraints, misinterpretation of critical information in questions, and divergent penalty thresholds among judge models.

Pattern Clustering. We apply density-based clustering to these rationales, revealing over 20 high-impact error categories, particularly vulnerabilities in perspective-taking and format adherence. Analysis and details are shown in Appendix A.4.

Meta-Judge Template Generation. For each error cluster, we develop structured templates that encode: *1) Question Characteristics* (domain-specific requirements, content/format constraints) and *2) Response Error Patterns* (failure types, localization, severity).

This aligns model judgments with human values and improves robustness against: (1) overstrict format-based rejection, (2) underpenalization of conceptual errors in fluent responses, and (3) context-sensitive scoring variations.

4.2 Complex Formula Augmentation

Verifying answers in domains such as the natural sciences is challenging due to the prevalence of complex expressions. These expressions often exhibit diverse notational conventions (e.g., symbolic, algebraic, floating-point, integer). Consequently, automated verifiers lacking robust mathematical equivalence understanding may erroneously reject semantically correct responses that differ superficially from reference solutions.

To address this issue, we introduce a *Complex Formula Augmentation* strategy that systematically generates multiple, notation-variant answers for each problem instance. Our procedure is as follows:

Reference Normalization. For each original question–answer pair in our dataset, we first convert the reference answer into a canonical representation, normalizing numeric precision and symbolic structure.

Variant Generation. We leverage the DeepSeek-v3 (Ma et al., 2025) to produce between one and three alternative formulations of the canonical answer. These variants include: *1) Symbolic rearrangements* (e.g., rationalizing denominators, applying algebraic identities). *2) Precision-preserving floating-point expansions*. *3) Equivalent integer or fraction representations*. We enforce strict constraints to avoid precision loss and ensure each variant remains mathematically equivalent to the original answer within the problem context.

Quality Control. All generated variants are automatically checked for equivalence using a symbolic algebra engine, and a subset is manually reviewed by subject-matter experts to confirm correctness and naturalness of presentation.

By exposing the verifier to diverse but equivalent formulae, we markedly improve its ability to recognize correct answers regardless of notational differences, thereby reducing false negative rates in formula-intensive tasks.

4.3 Generalizability Augmentation

Existing verifier models often rely on task-specific prompts, limiting their generalizability across different problems and subtle answer variations (e.g., numerical precision in TheoremQA (Chen et al., 2023)). To address this, we propose a *Generalizability Augmentation* strategy to enhance adaptability by systematically expanding prompt diversity in training data. We collect diverse prompts from public datasets (e.g., TheoremQA, GPQA (Rein et al., 2024), GAOKAOBench (Zhang et al., 2023)) and real-world scenarios, covering over 20 task types. For each prompt type, we design multiple variants, varying questioning styles, context lengths, linguistic registers, and instruction granularity. Our augmentation employs two key techniques:

Prompt Rewriting and Perturbation. We use LLMs (e.g., DeepSeek-v3) to automatically generate paraphrases, structural modifications, and detail-enriched prompt variants. We also introduce noise perturbations to improve robustness.

Cross-Domain Transfer Augmentation. We transfer high-quality prompts and verification tasks to new domains or task types, using domain adaptation techniques. This includes interdisciplinary transfer (e.g., math to physics) and cross-format transfer (e.g., natural language to code). Furthermore, during training, we introduce prompt random sampling, dynamic mixing, and a prompt-invariance mechanism to prevent overfitting and encourage consistent judgments across different prompt formulations, thereby enhancing generalization.

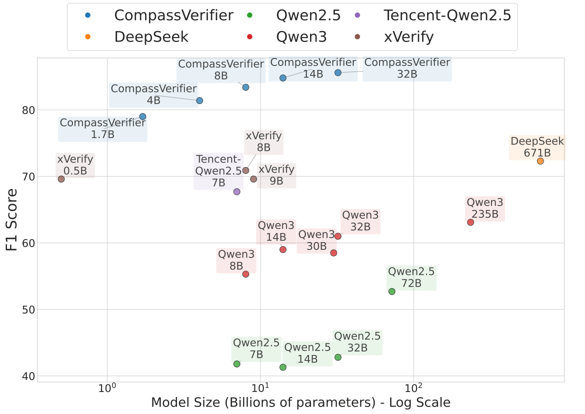


Figure 3: Model performances with size on VerifierBench. We show the F1 score in main results.

5 Experiments

Baselines and Setup. We conduct comprehensive evaluations on VerifierBench across various model scales of CompassVerifier, ranging from 1.7B to 32B parameters. Baseline models include: (1) general LLMs such as Qwen2.5 (Yang et al., 2024), Qwen3 (Yang et al., 2024), DeepSeek-V3 (Guo et al., 2025), and GPT-4o (OpenAI, 2024a); and (2) two recently proposed specialized verifier models: xVerify (Chen et al., 2025) and Tencent-Qwen2.5-7B-Instruct-RLVR (Su et al., 2025). We ask the model directly generate the final judgment of the given response and report F1 and Accuracy as metrics. More evaluation and training details are shown in Appendix A.3.

5.1 Main Results

From the Perspective of the Domain. We show the main results of VerifierBench in Table 1. Our CompassVerifier establishes new state-of-the-art performance across all VerifierBench categories, achieving 82.6–93.0% accuracy and 77.7–92.6% F_1-score in the 32B configuration. Three findings emerge: (1) As shown in Figure 3, verification capability exhibits progressive improvement with increasing scale, demonstrating accuracy gains from **84.6%** to **89.9%** and F_1-score improvements from **78.1%** to **86.2%** as parameters scale from 1.7B to 32B. (2) Verification-specific architectures yield substantial gains: CompassVerifier-14B surpasses the similarly-sized original Qwen3-14B by an absolute F_1-score improvement of 26.7%. (3) Science verification sees the largest improvements (92.4% accuracy), suggesting enhanced reasoning **in com-**

Table 1: Main results on the VerifierBench benchmark. For fair comparison, we treat the "**Invalid**" instances in VerifierBench as incorrect labels, presenting results in a binary classification framework. We report Accuracy and F1 scores (%) across four categories and their average.

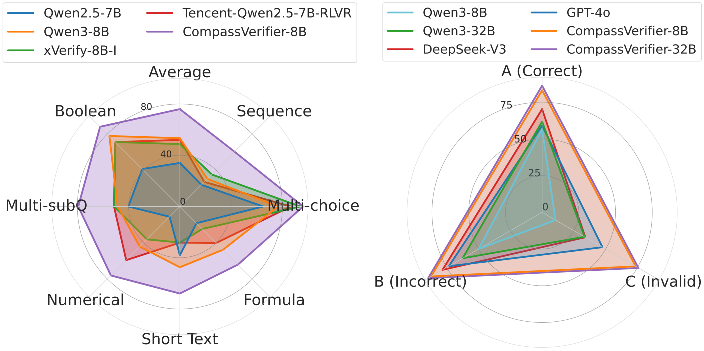
Model	Math		General Reasoning		Knowledge		Science		Average	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
General LLMs										
Qwen2.5-7B-Instruct	53.0	30.0	58.9	51.1	55.8	50.7	64.0	36.6	57.9	42.1
Qwen2.5-14B-Instruct	51.6	37.4	57.3	44.9	50.9	37.8	70.0	47.9	57.4	42.0
Qwen2.5-32B-Instruct	53.1	31.6	64.6	42.2	60.0	46.4	77.4	48.8	63.8	42.2
Qwen2.5-72B-Instruct	57.0	37.5	61.4	49.0	70.0	68.5	77.9	60.5	66.6	53.9
Qwen3-8B	53.0	51.6	61.6	61.8	63.8	69.4	57.9	42.9	59.1	56.4
Qwen3-14B	65.1	44.1	76.8	66.7	69.8	66.7	81.6	56.8	73.3	58.6
Qwen3-30B-A3B	59.7	62.4	63.4	63.2	61.5	64.4	59.5	48.7	61.0	59.7
Qwen3-32B	64.4	54.6	74.9	70.3	68.7	69.5	74.7	52.8	70.7	61.8
Qwen3-235B-A22B	64.2	53.9	78.5	73.7	67.4	73.1	74.0	50.0	71.0	62.7
GPT-4.1-2025-04-14	66.6	42.0	85.4	79.5	84.0	82.9	88.4	75.0	81.1	69.8
GPT-4o-2024-08-06	63.9	34.9	78.7	68.2	79.8	78.3	83.2	54.9	76.4	59.1
DeepSeek-V3-0324	69.4	54.7	81.5	76.6	80.6	81.2	84.7	68.5	79.1	70.3
Verifier Models										
xVerify-0.5B-I	61.7	42.6	84.0	78.5	87.1	86.2	86.3	72.6	79.8	70.0
xVerify-8B-I	64.3	42.6	84.3	78.9	86.1	85.1	88.7	74.9	80.8	70.4
xVerify-9B-C	64.3	48.0	82.8	77.0	82.7	81.7	86.3	69.8	79.0	69.1
Tencent-Qwen2.5-7B-RLVR	71.2	55.3	80.9	73.8	78.0	76.8	84.0	62.6	78.5	67.1
CompassVerifier-1.7B	73.1	60.6	89.5	86.4	86.8	86.9	89.0	78.6	84.6	78.1
CompassVerifier-4B	79.3	75.4	87.5	85.1	89.7	89.8	85.8	76.1	85.6	81.6
CompassVerifier-8B	80.7	77.3	88.8	86.8	91.2	91.3	86.3	76.9	86.8	83.1
CompassVerifier-14B	**82.9**	**80.3**	**92.0**	**90.0**	91.0	90.9	88.7	80.2	88.6	85.3
CompassVerifier-32B	82.6	77.7	91.5	89.3	**93.0**	**92.6**	**92.4**	**85.3**	**89.9**	**86.2**

prehending and validating scientific information. Despite progress, mathematical verification remains challenging (82.9% accuracy vs. 92.4% for science), highlighting persistent gaps in step-wise logical validation. Our smallest 1.7B variant outperforms GPT-4o by an absolute F_1-score improvement of 29.0%, demonstrating parameter efficiency. Consistent performance across domains further underscores the model's robustness. For instance, our CompassVerifier-32B model achieves high F_1-scores across all evaluated categories, such consistency indicates a well-generalized verification capability, effectively handling diverse types of information and reasoning processes.

From the Perspective of the Answer Type. Figure 4a demonstrates the performance comparison of similarly-sized models across different answer/question types. Notably, CompassVerifier-8B achieves consistent improvements across all categories. As evident from the results, multiple-choice questions emerge as the easiest category, with most models attaining strong performance, a finding attributable to their prevalence in evaluation benchmarks. However, baseline models show marked deficiencies in handling formula-based answers, multi-subquestions, and sequential answers, particularly struggling with sequential answers where none exceed 40 F_1-score. This likely stems from the inherent complexity of sequential answers, which often require element-by-element matching of multiple components, significantly increasing verification difficulty. These challenging cases represent precisely the focus of CompassVerifier and constitute critical directions for future research. Complete results are presented in Table 7.

5.2 Analysis of CompassVerifier

Beyond Binary Verification: Identifying Invalid Responses. Figure 4b presents the three-class classification performance of six top-performing models. Notably, even advanced general LLMs like GPT-4o and DeepSeek-V3 without task-specific training exhibit significant performance bias, demonstrating substantially better results on categories A and B compared to C. Our manual analysis reveals that general models show particular insensitivity to duplicated patterns or truncated responses. To address this, we implemented a duplicate string detection script during data filtering (Section 3.2). Crucially, we argue that Category



(a) Seven answer types. (b) Ternary classification labels.

Figure 4: Results (F1) on VerifierBench across 7 types and 3 labels.

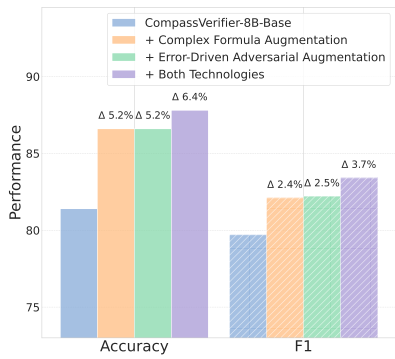


Figure 5: Ablation study on CompassVerifier-8B with different technologies.

Table 2: Experimental results of CompassVerifier as a reward model. We report the avg@32 performance on AIME24, AIME25, and MATH500.

Model	AIME24	AIME25	MATH500
Qwen3-4B-Base	2.71	1.77	34.11
GRPO *w/* Math-Verify	7.19	5.62	63.70
GRPO *w/* CompassVerifier-1.7B	**7.81**	**6.25**	**65.20**

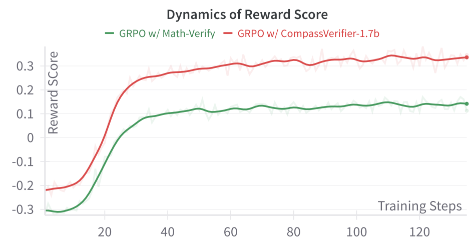


Figure 6: Dynamics of rewards during GRPO training.

C requires distinct treatment as they are particularly susceptible to reward hacking in reinforcement learning scenarios. Full results of the ternary classification performance are shown in Table 8.

Impact of Data Augmentation Components. Figure 5 details the impact of our data augmentation strategies on CompassVerifier-8B. The baseline model (CompassVerifier-8B-Base) achieves 81.4% accuracy and 79.7% F1. Introducing *Complex Formula Augmentation* alone improves accuracy to 86.6% (+5.2) and F1 to 82.1% (+2.4). This demonstrates the strategy's effectiveness in enhancing the model's capability to handle diverse formulaic expressions. Similarly, *Error-Driven Adversarial Augmentation* alone boosts accuracy to 86.6% (+5.2) and F1 to 82.2% (+2.5), underscoring its utility in fortifying the model against previously identified failure modes. Combining both strategies yields the best performance, with accuracy reaching 87.8% (+6.4) and F1 at 83.4% (+3.7), demonstrating their complementary and synergistic contributions to overall verification capabilities. Details are shown in Table 9.

5.3 CompassVerifier as Reward Model

To validate the efficacy of CompassVerifier as a reward model in reinforcement learning (RL), we examine its influence on enhancing the reasoning performance of models trained using RL. Specifically, we utilize GRPO (Shao et al., 2024) to train base LLMs with rule-based verifier Math-Verify (huggingface, 2024) and CompassVerifier-1.7B and rigorously evaluate the reasoning capabilities of the trained models. More experimental settings are provided in Appendix A.8.

Comparative results are detailed in Table 2. Experimental data indicate that models trained with CompassVerifier outperform both the base model and those trained with Math-Verify, underscoring the superior potential of CompassVerifier as a reward model. Additionally, as demonstrated in Figure 6, CompassVerifier exhibits a higher verification capacity, enabling it to deliver more valid signals (i.e., rewards) to the model during training, which results in superior performance.

6 Conclusion

To address the critical gap in large-scale answer verification evaluation, we present **VerifierBench**, featuring a meticulously designed pipeline for large-scale data collection, filtering, and annotation. We also introduce **CompassVerifier**, a novel verification model specifically engineered to handle multi-domain scenarios, diverse answer types, varied prompt formats, and irregular responses. CompassVerifier achieves superior accuracy and robustness compared to larger general LLMs and baseline verifier models. We anticipate that VerifierBench and CompassVerifier would significantly advance research in answer verification for evaluation frameworks and reward modeling for RL.

Limitations

While VerifierBench provides a comprehensive benchmark and CompassVerifier demonstrates strong capabilities in both evaluation and reward modeling for reinforcement learning, our work still has several limitations:

Limited Out-of-Distribution (OOD) Evaluation: Although VerifierBench facilitates thorough testing of verifier models like CompassVerifier, and its utility is shown in practical reinforcement learning scenarios, our OOD evaluation is constrained by the current scarcity of diverse, publicly available datasets specifically designed for verifier assessment. While we believe VerifierBench is a valuable contribution towards addressing this gap, further research is needed to establish broader OOD generalization capabilities for verifier models across a wider array of unseen domains and task formulations. We encourage the community to contribute to the development of more extensive OOD benchmarks for verifiers.

Emphasis on Outcome-Based rather than Process-Based Verification: CompassVerifier is primarily trained to assess the correctness of the final answer generated by an LLM, with less emphasis on evaluating the intermediate reasoning steps or the entire generation process. This design choice was influenced by the inherent complexity of LLM responses and considerations for verifier model scale and training efficiency. Consequently, our current model may not fully distinguish between correct answers derived from sound reasoning versus those resulting from flawed or incomplete derivations. Future work could explore methods for incorporating process-based supervision signals, potentially enhancing the verifier's ability to assess the faithfulness and interpretability of the reasoning process, which is crucial for complex, multi-step tasks.

Ethical Considerations

For our benchmark and models, we relied on reference materials and closed-source models that are accessible to the public, thereby avoiding any potential harm to individuals or groups. The data produced by the LLMs underwent a meticulous human selection and processing phase to ensure the protection of privacy and confidentiality. We did not use any personally identifiable information, and all data were anonymized prior to analysis. Additionally, we employed ChatGPT and Grammarly to refine our manuscript's language.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI-MO. 2024. Aime 2024.

Anthropic. 2024. Claude 3.5 sonnet.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, and 1 others. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*.

Maosong Cao, Alexander Lam, Haodong Duan, Hongwei Liu, Songyang Zhang, and Kai Chen. 2024. Compassjudger-1: All-in-one judge model helps model evaluation and evolution. *arXiv preprint arXiv:2410.16256*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

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

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. TheoremQA: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7901, Singapore. Association for Computational Linguistics.

Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. 2024. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*.

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

XTuner Contributors. 2023. Xtuner: A toolkit for efficiently fine-tuning llm. `https://github.com/InternLM/xtuner`.

Quy-Anh Dang and Chris Ngo. 2025. Reinforcement learning for reasoning in small llms: What works and what doesn't. *Preprint*, arXiv:2503.16219.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation.

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

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024a. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Yancheng He, Shilong Li, Jiaheng Liu, Yingshui Tan, Weixun Wang, Hui Huang, Xingyuan Bu, Hangyu Guo, Chengwei Hu, Boren Zheng, and 1 others. 2024b. Chinese simpleqa: A chinese factuality evaluation for large language models. *arXiv preprint arXiv:2411.07140*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.

huggingface. 2024. Math-verify: A robust mathematical expression evaluation system designed for assessing large language model outputs in mathematical tasks. `https://github.com/huggingface/Math-Verify`.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Akira Kawabata and Saku Sugawara. 2024. Rationale-aware answer verification by pairwise self-evaluation. *arXiv preprint arXiv:2410.04838*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *SIGOPS*.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, and 1 others. 2024a. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024b. Numinamath tir. Hugging Face repository. Dataset documentation available at `https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf`.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, hai zhao, and Pengfei Liu. 2024c. Generative judge for evaluating alignment. In *The Twelfth International Conference on Learning Representations*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

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

Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yunlong Feng, and Zhijiang Guo. 2024. Autopsv: Automated process-supervised verifier. *Advances in Neural Information Processing Systems*, 37:79935–79962.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 3.

Kaijing Ma, Xinrun Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, and 1 others. 2024. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. *arXiv preprint arXiv:2410.06526*.

Xueguang Ma, Qian Liu, Dongfu Jiang, Zejun Ma, and Wenhu Chen. 2025. General-reasoner: Advancing llm reasoning across all domains. https://github.com/TIGER-AI-Lab/General-Reasoner.

Skywork o1 Team. 2024. Skywork-o1 open series. https://huggingface.co/Skywork.

OC-Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

OpenAI. 2023. Openai evals: Evals is a framework for evaluating llms and llm systems, and an open-source registry of benchmarks. https://github.com/openai/evals.

OpenAI. 2024a. Gpt-4o.

OpenAI. 2024b. Gpt-4o mini.

OpenAI. 2024c. O1-preview.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

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

ByteDance Seed. 2025. Doubao-1.5-pro.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient RLHF framework. In *EuroSys*, pages 1279–1297. ACM.

Wenlei Shi and Xing Jin. 2025. Heimdall: test-time scaling on the generative verification. *arXiv preprint arXiv:2504.10337*.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham M. Kakade, Dean Foster, and Udaya Ghai. 2025. Mind the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*.

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv:2312.11805*.

Yidong Wang, Zhuohao Yu, Wenjin Yao, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2024a. PandaLM: An automatic evaluation benchmark for LLM instruction tuning optimization. In *The Twelfth International Conference on Learning Representations*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark. *arXiv preprint arXiv:2305.12474*.

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

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. 2025. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*.

Zihao Zhou, Shudong Liu, Maizhen Ning, Wei Liu, Jindong Wang, Derek F Wong, Xiaowei Huang, Qiufeng Wang, and Kaizhu Huang. 2024. Is your model really a good math reasoner? evaluating mathematical reasoning with checklist. *arXiv preprint arXiv:2407.08733*.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2025. JudgeLM: Fine-tuned large language models are scalable judges. In *The Thirteenth International Conference on Learning Representations*.

A Appendix

A.1 Details of VerifierBench Statistics

Table 3: Dataset Source Distribution

Source	Count	Percentage (%)
BBH	640	22.56
GaokaoBench	202	7.12
Math	184	6.49
MMLU Pro	172	6.06
GPQA Diamond	51	1.80
GSM8K	15	0.53
AIME2024	3	0.11
SimpleQA	97	3.42
Numina Train	109	3.84
HLE	357	12.58
KorBench	395	13.92
OlympiadBench	351	12.37
ARC Prize Public Evaluation	176	6.20
TheoremQA	85	3.00

Table 4: Category Distribution

Category	Count	Percentage (%)
A	1095	38.84
B	1541	54.66
C	183	6.49

Table 5: Domain Distribution

Domain	Count	Percentage (%)
General Reasoning	1152	40.87
Mathematical Reasoning	900	31.93
Knowledge	387	13.73
Scientific Reasoning	380	13.48

Table 6: Answer Type Distribution

Answer Type	Count	Percentage (%)
Multiple Choice	892	31.64
Short Text	354	12.56
Numerical	434	15.40
Formula	344	12.20
Multi-subproblem	281	9.97
Sequence	468	16.60
Boolean Answer	46	1.63

A.2 Details of VerifierBench Construction

Data Collection. Our experimental evaluation encompasses a comprehensive collection of 53 LLMs, including representative examples such as Qwen-2.5 (Yang et al., 2024), LLaMA3 (Grattafiori et al., 2024), DeepSeek-V3 (Liu et al., 2024), DeepSeek-R1 (Guo et al., 2025), GPT-4o (OpenAI, 2024a), GPT-4o-mini (OpenAI, 2024b), Gemini (Team et al., 2023), claude3-5 (Anthropic, 2024), Doubao-1.5-Pro (Seed, 2025), InternLM (Cai et al., 2024) and Mixtral (Jiang et al., 2024). All specific models are listed in Table 11. These models are evaluated across sixteen diverse benchmarks: GSM8K (Hosseini et al., 2024), Math (Hendrycks et al., 2021), AIME2024 (AI-MO, 2024), BBH (Suzgun et al., 2022), GaokaoBench (Zhang et al., 2023), HLE (Phan et al., 2025), KorBench (Ma et al., 2024), GPQA (Rein et al., 2024), SimpleQA (Wei et al., 2024), ChineseSimpleQA (He et al., 2024b), MMLU-Pro (Wang et al., 2024b), ARC (Chollet et al., 2024), OlympiadBench (He et al., 2024a), TheoremQA (Chen et al., 2023), NuminaMath (Li et al., 2024b), and Drop (Dua et al., 2019). Through the OpenCompass (OC-Contributors, 2023) framework, we collected more than 1.32 million response models, creating the most comprehensive response datasets to date.

VerifierBench Construction Details. For samples with inconsistent verification results across multiple models and prompts, we identified numerous cases that were either redundant or unworthy of human annotation. We employed a string-matching script to detect and remove duplicate responses, which predominantly belonged to category C (invalid responses). Additionally, we utilized DeepSeek-V3 to identify problematic cases, including: (1) questions with obvious open-ended nature, (2) incomplete reference answers, and (3) proof-based problems - all of which cannot be objectively evaluated solely based on reference answers and may introduce ambiguity in test set evaluation. After deduplication, approximately 5,000 samples underwent human annotation, where annotators further flagged the aforementioned problematic types. Annotation results revealed that most of the inconsistent samples were ultimately labeled as category B (incorrect responses), suggesting a potential tendency of LLM judges toward false positives. To maintain better label balance, we further applied similarity-based filtering to remove redundant samples within the category B subset. This rigorous filtering process yielded a final high-quality dataset of 2,819 samples.

A.3 Details of CompassVerifier Experiments

Evaluation Setup. We use OpenCompass (OC-Contributors, 2023) and employ both F1 score and Accuracy as evaluation metrics, with particular emphasis on the F1 score, as it provides a more comprehensive assessment considering the precision, recall, and balance of the class distribution simultaneously. For all open-source models, we use vllm (Kwon et al., 2023) for the acceleration of inference. For all models, we employ temperature=1.0 for data synthesis and temperature=0.0 for evaluation/verification, with both max_gen_len and max_model_len set to their maximum values. We use the official prompt for Xverify and Tencent-Qwen2.5-7B-Instruct-RLVR, and a general non-cot prompt for CompassVerifier and general LLMs can be found in the first prompt in Appendix A.6.

Training Setup. We use XTuner (Contributors, 2023) for training our CompassVerifier model on Qwen3 (Yang et al., 2024) series models, largely adhering to the original hyperparameters. Fine-tuning is conducted using a learning rate of 2×10^{-5} with a max sequence length 32768. A multiplicative learning rate decay is applied after each epoch, with a gamma value of 0.85. The batch sizes are set to 32. All models are trained for one epoch on the training set and fully fine-tuned on 8×A100 80GB GPUs.

Table 7: Detailed results on VerifierBench across different question types. We report Accuracy (Acc.) and F1 scores (%) for various problem categories and their average. Bold numbers indicate the best performance in each column.

Model	Boolean		Multi-sub		Numerical		Short Text		Formula		Multiple Choice		Sequence		Average	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Qwen2.5-7B-Instruct	63.0	41.4	45.9	40.2	49.5	11.3	65.0	38.0	53.5	18.4	62.0	65.0	59.2	23.9	56.9	34.0
Qwen2.5-14B-Instruct	63.0	66.7	54.5	45.0	57.4	39.3	59.9	42.3	53.8	26.9	49.0	45.9	68.8	34.8	58.0	43.0
Qwen2.5-32B-Instruct	58.7	53.7	65.8	37.7	56.7	33.9	61.3	27.7	59.3	19.5	55.8	52.5	80.6	19.5	62.6	34.9
Qwen2.5-72B-Instruct	73.9	71.4	65.8	46.7	62.0	36.8	57.9	47.7	57.0	27.5	61.9	62.4	74.8	40.4	64.8	47.6
Qwen3-8B	73.9	77.8	50.2	48.5	52.5	44.3	52.3	47.4	54.7	47.7	70.4	76.8	53.0	30.4	58.1	53.3
Qwen3-14B	69.6	66.7	69.8	52.0	64.8	39.0	76.6	56.1	66.6	27.7	72.4	73.8	84.6	39.0	72.0	50.6
Qwen3-30B-A3B	71.7	69.8	45.9	44.9	66.1	66.4	53.7	47.4	48.8	51.4	74.9	79.8	55.1	28.1	59.5	55.4
Qwen3-32B	80.4	80.9	63.4	55.9	64.8	51.4	68.6	57.1	64.2	44.3	74.3	77.8	78.4	46.0	70.6	59.1
Qwen3-235B-A22B	67.4	57.1	60.9	52.6	63.8	48.9	67.8	56.1	62.5	43.5	79.0	82.6	83.3	50.4	69.2	55.9
GPT-4.1-2025-04-14	80.4	80.0	68.3	44.7	64.1	31.6	83.1	64.7	68.6	22.9	89.4	91.0	88.3	43.3	77.4	54.0
GPT-4o-2024-08-06	65.2	63.6	63.7	37.0	63.6	29.5	79.7	54.4	67.2	11.0	80.0	81.9	86.8	35.4	72.3	44.7
DeepSeek-V3-0324	63.0	56.4	61.2	52.0	68.2	48.9	81.6	66.3	69.5	39.3	85.4	87.6	85.5	54.1	73.5	57.8
xVerify-0.5B-I	67.4	59.5	66.9	25.6	63.6	37.8	64.7	36.6	60.8	22.0	95.7	96.6	85.5	35.0	72.1	44.7
xVerify-8B-I	71.7	71.1	73.0	51.3	65.2	36.3	65.3	28.1	66.6	24.8	92.6	94.0	88.3	35.3	74.7	48.7
xVerify-9B-C	67.4	70.6	76.9	50.4	65.2	40.8	58.8	34.8	63.4	30.0	92.3	93.6	85.9	29.8	72.8	50.0
Tencent-Qwen2.5-7B-Instruct-RLVR	71.7	71.1	69.0	51.4	74.9	59.2	71.2	28.2	69.8	40.2	84.2	86.5	85.0	27.1	75.1	52.0
CompassVerifier-1.7B	82.6	82.6	77.2	62.8	73.7	60.1	77.4	56.8	67.7	36.6	96.3	97.1	90.2	54.0	80.7	64.3
CompassVerifier-4B	78.3	80.0	77.9	68.1	76.7	69.1	80.5	68.8	75.6	67.9	95.2	96.3	88.0	59.4	81.7	72.8
CompassVerifier-8B	87.0	88.0	87.5	79.3	81.1	76.1	79.4	67.9	72.7	64.1	95.2	96.2	88.0	60.4	84.4	76.0
CompassVerifier-14B	**91.3**	**92.0**	**90.8**	**84.5**	**81.3**	**76.7**	84.2	**73.8**	74.7	67.4	96.6	97.4	**91.2**	**66.7**	87.2	**79.8**
CompassVerifier-32B	89.1	90.2	87.9	79.8	80.0	72.4	**85.6**	73.6	**80.2**	**69.6**	**97.3**	**97.9**	**91.2**	61.7	**87.3**	77.9

Table 8: Three-label classification performance on VerifierBench. Beyond binary correctness (correct/incorrect), this evaluation requires models to identify invalid responses. We report Accuracy and macro-F_1 scores (in %) across four distinct categories and their overall average.

Model	Math		General Reasoning		Knowledge		Science		Average	
	Acc.	macro-F1	Acc.	macro-F1	Acc.	macro-F1	Acc.	macro-F1	Acc.	macro-F1
Qwen2.5-7B-Instruct	39.6	29.2	49.2	37.8	45.2	34.6	50.3	34.2	46.1	34.0
Qwen2.5-14B-Instruct	44.2	37.7	50.9	40.1	42.9	37.6	57.1	44.1	48.8	39.9
Qwen2.5-32B-Instruct	46.0	35.7	59.8	47.8	55.6	45.7	70.8	52.5	58.0	45.4
Qwen2.5-72B-Instruct	51.1	43.0	57.3	48.6	67.4	52.2	72.9	58.8	62.2	50.7
Qwen3-8B	48.2	35.8	54.0	42.3	56.1	41.1	47.9	36.5	51.5	38.9
Qwen3-14B	61.3	57.3	72.3	63.5	65.4	54.7	74.7	61.9	68.4	59.4
Qwen3-30B	53.3	45.6	49.6	42.1	54.8	50.2	45.0	39.0	50.7	44.2
Qwen3-32B	57.2	54.2	61.6	54.4	60.2	51.7	58.7	50.0	59.4	52.6
Qwen3-235B-A22B	58.8	42.8	73.8	55.0	65.4	48.6	67.6	52.4	66.4	49.7
GPT-4.1-2025-04-14	61.7	59.6	78.1	73.6	78.3	69.7	79.5	68.4	74.4	67.8
GPT-4o-2024-08-06	57.9	53.9	68.3	62.9	73.4	66.0	71.1	57.1	67.7	60.0
DeepSeek-V3-0324	63.2	49.1	77.4	66.2	76.5	60.3	80.5	67.8	74.4	60.9
CompassVerifier-1.7B	70.3	69.7	87.4	85.7	85.0	80.9	86.6	81.1	82.3	79.3
CompassVerifier-4B	77.6	78.2	85.6	82.2	88.1	82.9	83.4	80.1	83.7	80.9
CompassVerifier-8B	79.2	79.5	87.2	82.3	90.2	84.9	84.2	78.7	85.2	81.3
CompassVerifier-14B	**81.1**	**81.0**	88.9	85.1	89.7	**86.2**	86.3	82.6	86.5	83.7
CompassVerifier-32B	80.6	80.1	**89.7**	**86.3**	**91.5**	85.3	**89.5**	**83.6**	**87.8**	**83.8**

Table 9: Ablation study on CompassVerifier-8B with different augmentation strategies on VerifierBench main results. *Complex Formula Augmentation* enhances formula variants verification, *Error-Driven Adversarial Augmentation* fortifies against failure cases.

Setting	**Accuracy (%)**	**Δ Acc (%)**	**F1 (%)**	**Δ F1 (%)**
CompassVerifier-8B-Base	81.4	-	79.7	-
+ Complex Formula Augmentation	86.6	+5.2	82.1	+2.4
+ Error-Driven Adversarial Augmentation	86.6	+5.2	82.2	+2.5
+ Both Augmentations	87.8	+6.4	83.4	+3.7

A.4 Details of Meta Error Patterns

Error Analysis and Patterns. VerifierBench is designed not merely as a benchmark dataset for model evaluation, but as a comprehensive framework incorporating extensive *human analysis* and *case studies*. During annotation, we required annotators to provide detailed judgment rationales in addition to final labels. Through systematic collection and analysis of these rationales, we identified and categorized over 30 meta error patterns, which represent fundamental causes of mistakes and hallucinations in LLM-based answer verification. For example, while mathematically equivalent formulas are conventionally accepted as correct answers by LLMs or tools, they should be rejected for expression simplification problems. Similarly, for questions admitting multiple valid answers listed in the reference answer, a model response matching any one option should be considered correct, rather than complete coverage. We have found these meta patterns invaluable for both diagnostic analysis and targeted model improvement, and have incorporated them into our training framework.

We display the meta error patterns in three categories: A (Correct), B (Incorrect), and C (Invalid) as shown in the following figures.

Meta Pattern: A (Correct)

- The units in the LLM Response differ from those in the final answer, resulting in different numerical expressions, but they are consistent upon conversion, should be judged as Correct.
- The reference answer is an extremely complex formula, and the LLM Response appears very different in form but simplifies to an equivalent expression, with no explicit requirement for simplification in the question, should be judged as Correct.
- The question requires calculating a numerical decrease, and the LLM Response has the opposite sign of the reference answer because either uses negative signs to represent decrease, but they are equivalent, should be judged as Correct.
- The reference answer provides multiple candidate answers without requiring all possibilities. The LLM Response provides one of them, should be judged as Correct.
- The question doesn't explicitly specify answer format (numerical or formula). The LLM Response and reference answer differ in form but are equivalent when calculated, should be judged as Correct.
- The question requires specific formatting (order, capitalization, etc.). While the LLM Response appears different from the reference answer in formatting, upon inspection it fully complies, should be judged as Correct.
- When calculating values with units, the reference answer and LLM Response may differ in unit representation or numerical values, but are equivalent after unit conversion, should be judged as Correct.
- For multiple-choice or true/false questions, the LLM Response ultimately gives the correct answer despite showing significant uncertainty, should be judged as Correct.
- The question requires expressions meeting simple conditions (sum, product, logical relations, etc.), and the reference answer may include multiple valid forms. The LLM Response differs in form but meets all requirements, should be judged as Correct.
- The LLM initially provides an incorrect answer but corrects it after reflection, should be judged as Correct.
- The reference answer consists of multiple sub-questions. The LLM answers all sub-questions correctly during reasoning, even if not presented together at the end, should be judged as Correct.

Meta Pattern: B (Inorrect)

- For multiple-choice questions, the LLM Response selects the correct option but follows with unrelated option content, should be judged as Incorrect.
- The question requires formula simplification. The LLM answer isn't fully simplified to minimal form, even if equivalent to the reference answer, should be judged as Incorrect.

- The reference answer is a formula with specified output format. The LLM answer doesn't comply with this format, even if equivalent, should be judged as Incorrect.
- The question requires an expression where the sum equals a certain value with each number used once. The LLM Response repeats numbers while satisfying the sum, should be judged as Incorrect.
- The reference answer is an un-simplified logical formula after substitution. The LLM Response is incorrect due to simplification causing format errors, should be judged as Incorrect.
- The LLM Response only provides solution code without final results, should be judged as Incorrect.
- The LLM Response (formula/numerical) and reference answer aren't equivalent when calculated, should be judged as Incorrect.
- When describing numerical intervals, the reference answer and LLM Response differ in endpoint inclusion (open/closed), should be judged as Incorrect.
- For sequence decryption requiring exact matching, the LLM Response doesn't match the reference answer, should be judged as Incorrect.
- The reference answer is a long sequence requiring exact correspondence. The LLM Response has minor differences with some errors, should be judged as Incorrect.
- The question explicitly requires multiple candidate answers (provided in reference), but the LLM Response gives only one, should be judged as Incorrect.
- The LLM initially provides a correct answer but changes to incorrect or "unanswerable" after reflection, should be judged as Incorrect.
- For symbolic sequences, the LLM Response contains garbled characters, should be judged as Incorrect.
- The reference answer is numerical, and the LLM Response provides more decimal places but rounds differently, should be judged as Incorrect.
- The reference answer is an extremely large number, and the LLM Response provides a high-order power expression that doesn't match after calculation, should be judged as Incorrect.
- After detailed reasoning, the LLM Response fails to provide a clear answer or states the question is unanswerable, should be judged as Incorrect.
- For multi-part questions, the number of final answers in the LLM Response doesn't match the reference answer, should be judged as Incorrect.

Meta Pattern: C (Invalid)

- The question contains multiple sub-questions, but the number of reference answers doesn't match, indicating quality issues, should be judged as Invalid.
- The reference answer has serious omissions, truncation, or formatting issues, should be judged as Invalid.
- The question itself has serious omissions, truncation, or formatting issues, should be judged as Invalid.
- The LLM doesn't answer normally, stating it needs more information or internet access, should be judged as Invalid.
- The LLM Response is clearly truncated and incomplete, should be judged as Invalid.
- The LLM Response is mostly garbled text with no valuable information extractable, should be judged as Invalid.
- The LLM Response contains extensive meaningless repetition, making correct answers unidentifiable, should be judged as Invalid.

A.5 Meta-Judge Template Generation Fields

Table 10: Meta-Judge Template Generation Fields (Academic Disciplines and Subfields)

Category	Discipline	Subfields
Natural Sciences	Mathematics	Differential calculus, Integral calculus, Probability statistics, Operations research, Mathematical logic, Financial mathematics, Topology, Algebraic geometry
	Physics	Theoretical physics, Quantum mechanics, Condensed matter physics, Astrophysics, Nuclear physics, Optics, Acoustics
	Chemistry	Analytical chemistry, Organic chemistry, Inorganic chemistry, Physical chemistry, Materials chemistry, Environmental chemistry, Chemical biology
	Biology	Molecular biology, Genetics, Ecology, Cell biology, Biochemistry, Microbiology
	Earth Sciences	Geology, Geophysics, Atmospheric sciences, Oceanography, Environmental science, Paleontology
	Statistics	Data science, Biostatistics, Economic statistics, Machine learning algorithms, Bayesian analysis
Engineering	Mechanical Engineering	Mechanical design & manufacturing, Automatic control, Robotics, Vehicle engineering, Thermal & power engineering, MEMS
	Computer Science & Technology	Artificial intelligence, Computer networks, Software engineering, Computer vision, Cybersecurity, Big data analytics
	Electronic Information Engineering	Communication engineering, IC design, Optoelectronic technology, Wireless sensor networks, Smart grid
	Civil Engineering	Structural engineering, Bridge & tunnel design, Geotechnical engineering, Hydraulic engineering, Urban planning
	Materials Science & Engineering	Nanomaterials, Metallic materials, Polymer materials, Composite materials, Material processing
	Chemical Engineering	Chemical process design, Petroleum refining, Biochemical engineering, Catalytic reaction engineering, Separation technology
	Environmental Engineering	Pollution control technology, Environmental monitoring, Ecological restoration, Solid waste treatment, Clean energy development
	Aerospace Engineering	Aircraft design, Propulsion systems, Aerodynamics, Satellite navigation, Aerospace materials
	Biomedical Engineering	Medical imaging technology, Biomaterials, Artificial organs, Biosensors, Rehabilitation engineering
	Energy & Power Engineering	Nuclear technology, Wind energy development, Solar energy utilization, Fuel cells, Thermal system optimization

A.6 PromptList

```
Please as a grading expert, judge whether the final answers given by the candidates
below are consistent with the standard answers, that is, whether the candidates
answered correctly.
Here are some evaluation criteria:
1. Please refer to the given standard answer. You don't need to re-generate the
answer to the question because the standard answer has been given. You only need to
judge whether the candidate's answer is consistent with the standard answer
according to the form of the question. THE STANDARD ANSWER IS ALWAYS CORRECT AND THE
 QUESTION IS PERFECTLY VALID. NEVER QUESTION THEM.
2. ONLY compare the FINAL ANSWER - COMPLETELY IGNORE any potential errors in the
REASONING PROCESSES.
3. Some answers may be expressed in different ways, such as some answers may be a
mathematical expression, some answers may be a textual description, as long as the
meaning expressed is the same. Before making a judgment, please understand the
question and the standard answer first, and then judge whether the candidate's
answer is correct.
4. Some answers may consist of multiple items, such as multiple-choice questions,
multiple-select questions, fill-in-the-blank questions, etc. Regardless of the
question type, the final answer will be considered correct as long as it matches the
 standard answer, regardless of whether the reasoning process is correct. For
multiple-select questions and multi-blank fill-in-the-blank questions, all
corresponding options or blanks must be answered correctly and match the standard
answer exactly to be deemed correct.
5. If the prediction is given with \\boxed{{}}, please ignore the \\boxed{{}} and
only judge whether the candidate's answer is consistent with the standard answer.
6. If the candidate's answer is invalid (e.g., incomplete (cut off mid-response),
lots of unnormal repetitive content, or irrelevant to the question, saying it can't
answer the question because some irresistible factors, like ethical issues, no
enough information, etc.), select option C (INVALID).Please judge whether the
following answers are consistent with the standard answer based on the above
criteria. Grade the predicted answer of this new question as one of:
```

```
A: CORRECT
B: INCORRECT
C: INVALID
Just return the letters ”A”, ”B”, or ”C”, with no text around it.
Here is your task. Simply reply with either CORRECT, INCORRECT, or INVALID. Don't
apologize or correct yourself if there was a mistake; we are just trying to grade
the answer.
<Original Question Begin>:
{question}
<Original Question End>
<Standard Answer Begin>:
{gold_answer}
<Standard Answer End>
<Candidate's Answer Begin>:
{llm_response}
<Candidate's Answer End>
Judging the correctness of the candidate's answer:
```

Prompt 1: Prompt for general LLM evaluation

```
As a grading expert, your task is to determine whether the candidate's final answer
matches the provided standard answer. Follow these evaluation guidelines precisely:

Evaluation Protocol:
1. Reference Standard:
   - The standard answer is definitive and always correct
   - The question is perfectly valid - never question them
   - Do not regenerate answers; only compare with the given standard

2. Comparison Method:
   - Carefully analyze the question's requirements and the standard answer's
   structure
     * Determine whether the question expects exact matching of the entire standard
     answer or allows partial matching of its components.
     * This determination must be made based on the question's phrasing and the
     nature of the standard answer.
   - Compare ONLY the candidate's final answer (ignore all reasoning/explanation
   errors)
   - Disregard any differences in formatting or presentation style
   - For mathematical expressions: calculate step by step whether the two formulas
   are equivalent
   - For multiple-choice questions: compare only the final choice and corresponding
   option content

3. Multi-part Answers:
   - For questions requiring multiple responses (e.g., multi-select):
   - All parts must match the standard answer exactly.
   - Compare each sub-answer step by step. Partial matches are considered incorrect.

4. Validity Check:
   - Reject answers that are:
     * Incomplete (cut off mid-sentence in the final sentence, lacking a complete
     response) - Label as INCOMPLETE
     * Repetitive (repetition of words or phrases in a loop) - Label as REPETITIVE
     * Explicit refusals (e.g., directly return ”I cannot answer/provide/access
     ...") - Label as REFUSAL
   - For invalid answers, specify the type in the judgment (e.g., \boxed{C} -
   INCOMPLETE).

Grading Scale:
\boxed{A} - CORRECT:
   - Answer matches standard exactly (including equivalent expressions)
   - For numerical answers: consider as equivalent if values match when rounded
   appropriately
   - Semantically equivalent responses

\boxed{B} - INCORRECT:
   - Any deviation from standard answer
   - Partial matches for multi-part questions
```

```
\boxed{C} - INCOMPLETE/REPETITIVE/REFUSAL:
   - Fails validity criteria above (must specify: INCOMPLETE/REPETITIVE/REFUSAL)

Execution Steps and Output Formats:

Analysis step by step: [
Thoroughly evaluate the candidate's answer including:
(1) First check if the answer is INCOMPLETE (cut off mid-sentence), REPETITIVE (
looping repetition), or a REFUSAL (explicit denial) - if so, immediately classify as
 \boxed{C} with the corresponding type.
(2) Analyze the question's core requirements and the standard answer's structure,
for example:
- Strict requirements: Identify mandatory constraints (e.g., simplification, answer
order, multi-part completeness)
- Tolerant allowances: Ignore non-critical deviations (e.g., missing option labels
in MCQs, equivalent but unformatted expressions)
- Required answer type, precision level, etc.
(3) Perform a detailed comparison between the candidate's final answer and the
standard answer, for example:
- Content equivalence
- Permitted variations in numerical precision
- Allowed expression formats]
Final Judgment: \boxed{A/B/C} - <CORRECT/INCORRECT/INCOMPLETE/REPETITIVE/REFUSAL>

Here is your task.
<Original Question Begin>
{question}
<Original Question End>

<Standard Answer Begin>
{gold_answer}
<Standard Answer End>

<Candidate's Answer Begin>
{llm_response}
<Candidate's Answer End>

Analysis step by step and Final Judgment:
```

Prompt 2: Prompt A for CoT answer verification

```
As a grading expert, your task is to determine whether the candidate's final answer
matches the provided standard answer. Follow these evaluation guidelines precisely:

Evaluation Protocol:
1. Reference Standard:
   - The standard answer is definitive and always correct
   - The question is perfectly valid. Never question them
   - Do not regenerate answers; only compare with the given standard answer

2. Thoroughly evaluate the candidate's answer follow these steps
   - Carefully analyze the question's content and requirements
     * Strict requirements: Identify mandatory constraints (e.g., simplification,
     answer order, multi-part completeness)
     * Tolerant requirements: Ignore non-critical deviations (e.g., missing option
     labels in MCQs, equivalent but unformatted expressions)
   - Carefully analyze the standard answer's content and structure. Determine
   whether the question expects exact matching of the entire standard answer or
   allows partial matching of its components
   - Validity Check for the candidate's answer. Reject answers that are:
     * Incomplete (cut off mid-sentence in the final sentence, lacking a complete
     response) - Label as INCOMPLETE
     * Repetitive (repetition of words or phrases in a loop) - Label as REPETITIVE
     * Explicit refusals (e.g., directly return "I cannot answer/provide/access
     ...") - Label as REFUSAL
   - Perform a detailed comparison between the candidate's final answer and the
   standard answer
     * Compare ONLY the candidate's final answer (ignore all reasoning/explanation
     errors)
```

```
    * Disregard any differences in formatting or presentation style
    * For mathematical expressions: calculate step by step whether the two formulas
     are equivalent
    * For multiple-choice questions: compare only the final choice and the
    corresponding option content
    * For questions requiring multiple sub-answers (e.g., multi-select): All parts
    must match the standard answer exactly. Compare each sub-answer step by step.
    Partial matches are considered incorrect.

3. Grading Scale:
   \boxed{A} - CORRECT:
      - Answer matches standard exactly (including equivalent expressions)
      - For numerical answers: consider as equivalent if values match when rounded
      appropriately
      - Semantically equivalent responses
   \boxed{B} - INCORRECT:
      - Any deviation from standard answer
      - Partial matches for multi-part questions
   \boxed{C} - INCOMPLETE/REPETITIVE/REFUSAL:
      - Fails validity criteria above (must specify: INCOMPLETE/REPETITIVE/REFUSAL)

Output Formats:
Analysis: [Analysis and evaluate step by step here.]
Final Judgment: \boxed{A/B/C} - <CORRECT/INCORRECT/INCOMPLETE/REPETITIVE/REFUSAL>

Here is your task.
<Original Question Begin>
{question}
<Original Question End>

<Standard Answer Begin>
{gold_answer}
<Standard Answer End>

<Candidate's Answer Begin>
{llm_response}
<Candidate's Answer End>

Analysis:
Final Judgment:
```

Prompt 3: Prompt B for CoT answer verification

```
As a grading expert, your task is to determine whether the candidate's final answer
matches the provided standard answer. Follow these evaluation guidelines precisely:

Evaluation Protocol:
1. Reference Standard:
   - The standard answer is definitive and always correct
   - The question is perfectly valid - never question them
   - Do not regenerate answers; only compare with the given standard

2. Comparison Method:
   - Extract ONLY the candidate's final answer (ignore all reasoning/explanation
   errors)
   - If no complete final answer exists (e.g., response is cut off or contains only
   reasoning) - INVALID
   - Compare this directly with the standard answer
   - Disregard any differences in formatting or presentation style
   - For mathematical expressions: compare semantic equivalence, not syntax
   - For \boxed{} format: ignore the \boxed notation when comparing

3. Multi-part Answers:
   - For questions requiring multiple responses (e.g., multi-select):
   - All parts must match the standard answer exactly
   - Partial matches are considered incorrect

4. Validity Check:
   - Reject answers that are:
```

```
    * Incomplete (cut off mid-response or missing final answer)
    * Purely reasoning without final answer
    * Repetitive or uninterpretable
    * Irrelevant to the question
    * Explicit refusals (e.g., "I cannot answer/provide/access ...")

Grading Scale:
\boxed{A} - CORRECT:
   - Answer matches standard exactly (including equivalent expressions)
   - For numerical answers: allow 1% tolerance for floating-point variations
   - Semantically equivalent responses

\boxed{B} - INCORRECT:
   - Any deviation from standard answer
   - Partial matches for multi-part questions

\boxed{C} - INVALID:
   - Fails validity criteria above

Execution Steps and Output Formats:
Analysis:
1. Completeness and Validity Check: [confirm if candidate's answer is complete and
include the final answer]
2. Extracted Final Answer: [state what was identified as final answer]
3. Standard Comparison: [describe how it matches/mismatches]
Final Judgment: [\boxed{A/B/C}]

Here is your task.
<Original Question Begin>
{question}
<Original Question End>

<Standard Answer Begin>
{gold_answer}
<Standard Answer End>

<Candidate's Answer Begin>
{llm_response}
<Candidate's Answer End>

Analysis and Final Judgment:
```

Prompt 4: Prompt C for CoT answer verification

Table 11: List of Models Used in the Experiment with Response Counts

Model Family	Model Name	Response Count
Yi	Yi-Lightning	18496
	Yi-1.5-9B-Chat	17722
GPT	GPT-4o	18495
	GPT-4o-mini	44502
	GPT-4-1-2025-0414	2673
	GPT-4.5-preview-2025-02-27	18381
Doubao	Doubao-Pro-32k-241215	6378
	Doubao-Pro-1.5-32k-250115	18517
	Doubao-Pro-32k-240828	5692
Qwen	Qwen-Max-0919	18434
	Qwen-Max-2025-01-25	29173
	Qwen2.5-Max	18320
	Qwen2.5-7B-Instruct	49003
	Qwen2.5-14B-Instruct	32116
	Qwen2.5-32B-Instruct	37477
	Qwen2.5-72B-Instruct	37568
	QwQ-32B	20623
Gemini	Gemini-2.0-Flash-Exp	17303
	Gemini-1.5-Pro	18429
	Gemini-2-5-Pro-03-25	669
DeepSeek-R1	DeepSeek-Chat-R1	16556
	DeepSeek-R1-distill-Qwen-1.5B	16012
	DeepSeek-R1-distill-Qwen-7B	16364
	DeepSeek-R1-distill-Llama-8B	15731
	DeepSeek-R1-distill-Qwen-14B	16671
	DeepSeek-R1-distill-Qwen-32B	16042
	DeepSeek-R1-distill-Llama-70B	15772
Llama	Llama-3-1-8B-Instruct	44857
	Llama-3-1-70B-Instruct	18018
	Llama-3-2-3B-Instruct	28618
	Llama-3-3-70B-Instruct	28307
Mixtral	Mistral-Small-Instruct-2409	18233
	Mistral-Small-3.1-24B-Instruct	14331
	Ministral-8B-Instruct-2410	17962
	Mixtral-Large-Instruct-2411	18381
Claude	Claude-3-5-Sonnet-20241022	18521
	Claude-3-7-Sonnet-20250219	18474
	Claude-3-7-Sonnet-20250219-Thinking	4723
Gemma	Gemma-2-9B-It	34541
	Gemma-2-27B-It	34704
	Gemma3-27B-It	13120
DeepSeek-Chat	DeepSeek-V2.5	31896
	DeepSeek-Chat-V3	31950
InternLM	InternLM2.5-7B-Chat	43336
	InternLM2.5-20B-Chat	37594
	InternLM3-8B-Instruct	15976
Phi	Phi-4	18360
GLM	GLM-4-9B-Chat	17537
	GLM-4-Plus	18486
MiniMax	MiniMax-Text-01	39570
Moonshot	Moonshot-V1-32k	18067
Hunyuan	Hunyuan-Standard-256K	18082
StepFun	Step-2-16k	18405

A.7 Details of CompassVerifier Model Train Data

For the composition of CompassVerifier train dataset, we use 54420 consist samples from the VerifierBench pipeline as shown in Figure 1 as the base train set, we then use **Error-Driven Adversarial Augmentation** and **Complex Formula Augmentation** to construct extra data comprehensively enhance the capabilities of the CompassVerifier model. The composition of our train data list in Table 12.

Table 12: Composition of CompassVerifier Training Data

Data Source	Number of Samples	Percentage (%)
Base Train Set (VerifierBench)	54,420	56.20
Error-Driven Adversarial Augmentation	24,294	25.09
Complex Formula Augmentation	18,118	18.71
Total	**96,832**	**100.00**

Error-Driven Adversarial Augmentation Using DeepSeek-v3, we generate 34 Meta-Judge Templates covering common and extreme error scenarios then generate 224294 synthetic examples that emphasize decision boundary cases, especially where human judges tolerate minor errors that baseline verifiers over-penalize.

Complex Formula Augmentation Applying this augmentation pipeline, we have synthesized approximately 18118 enhanced examples spanning 14 distinct scientific and engineering disciplines.

A.8 Details of CompassVerifier-as-Reward Experimental Settings

Base LLMs. we utilize Qwen3-4B-Base (Yang et al., 2025) as the base LLM for the GRPO training.

Training Template. We utilize the following training template to prompt the base LLM to generate a response for each question. We only verify the format correctness to ensure the final answer is encapsulated within '\boxed{...final answer...}'.

Training Template of CompassVerifier

A conversation between a User and an Assistant. The User poses a question, and the Assistant provides a solution. The Assistant's response follows these structured steps:

1. **Reasoning Process**: The Assistant comprehensively thinks about the problem through a reasoning process.

2. **Conclusion**: The Assistant reaches a conclusion, which is enclosed within '<conclusion>' and '</conclusion>' tags. The final answer is highlighted within '\boxed{...final answer...}'.

3. **Response Format**: The complete response should be formatted as:

...reasoning process...

<conclusion>

...conclusion...

The answer is \boxed{...final answer...}

</conclusion>

Training Data. We utilize the challenging mathematical reasoning dataset Open-S1 (Dang and Ngo, 2025) as the RL training corpus. To increase the difficulty of our validation, we curate the final training set by specifically excluding problems with integer solutions from the original Open-S1 dataset.

Reward Design. We design a simple reward scheme: -1 for format errors, 0 for answer errors, and 1 for correct responses.

Training Parameters. We utilize the following loss function, with Table 13 detailing the training parameters:

$$\mathcal{L} = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\min\left(\frac{\pi_\theta\left(o_{i,t}|q,o_{i,<t}\right)}{\pi_{\theta_{\text{old}}}\left(o_{i,t}|q,o_{i,<t}\right)}a_{i,t}, \text{clip}\left(\frac{\pi_\theta\left(o_{i,t}|q,o_{i,<t}\right)}{\pi_{\theta_{\text{old}}}\left(o_{i,t}|q,o_{i,<t}\right)}, 1-\epsilon_{\text{min},}, 1-\epsilon_{\text{max}}\right)a_{i,t}\right)\right], \tag{3}$$

where $\mathcal{D}$ denotes the training data, (q, a) represents the question-answer pair, G signifies the group size, and

$$a_{i,t} = r_i - \text{mean}(\{r_i\}_{i=1}^{G}). \tag{4}$$

In this context, $a_{i,t}$ signifies the advantage of response o_i at the t-th position, and r_i denotes the reward of response o_i. Essentially, the KL penalty of the original GRPO loss is omitted, and zero mean normalization is employed to estimate the advantage.

Table 13: Training parameters of CompassVerifier as reward experiments.

Parameters	**Value**
train batch size	256
learning rate	$1e$-6
max prompt length	4096
max response length	12288
G	8
ϵ_{min}	0.2
ϵ_{max}	0.28

Hardware. All experiments are conducted on clusters equipped with 8 NVIDIA A800-SXM4-80GB GPUs and Intel(R) Xeon(R) Platinum 8336C CPUs, implementing with veRL (Sheng et al., 2025).