

# Reward Reasoning Model

Jiaxin Guo<sup>\*1,2</sup> Zewen Chi<sup>\*1</sup> Li Dong<sup>\*1</sup>  
Qingxiu Dong<sup>1,3</sup> Xun Wu<sup>1</sup> Shaohan Huang<sup>1</sup> Furu Wei<sup>1◇</sup>  
<sup>1</sup> Microsoft Research <sup>2</sup> Tsinghua University <sup>3</sup> Peking University  
<https://aka.ms/GeneralAI>

## Abstract

Reward models play a critical role in guiding large language models toward outputs that align with human expectations. However, an open challenge remains in effectively utilizing test-time compute to enhance reward model performance. In this work, we introduce Reward Reasoning Models (RRMs), which are specifically designed to execute a deliberate reasoning process before generating final rewards. Through chain-of-thought reasoning, RRMs leverage additional test-time compute for complex queries where appropriate rewards are not immediately apparent. To develop RRMs, we implement a reinforcement learning framework that fosters self-evolved reward reasoning capabilities without requiring explicit reasoning traces as training data. Experimental results demonstrate that RRMs achieve superior performance on reward modeling benchmarks across diverse domains. Notably, we show that RRMs can adaptively exploit test-time compute to further improve reward accuracy. The pretrained models are available at <https://huggingface.co/Reward-Reasoning>.

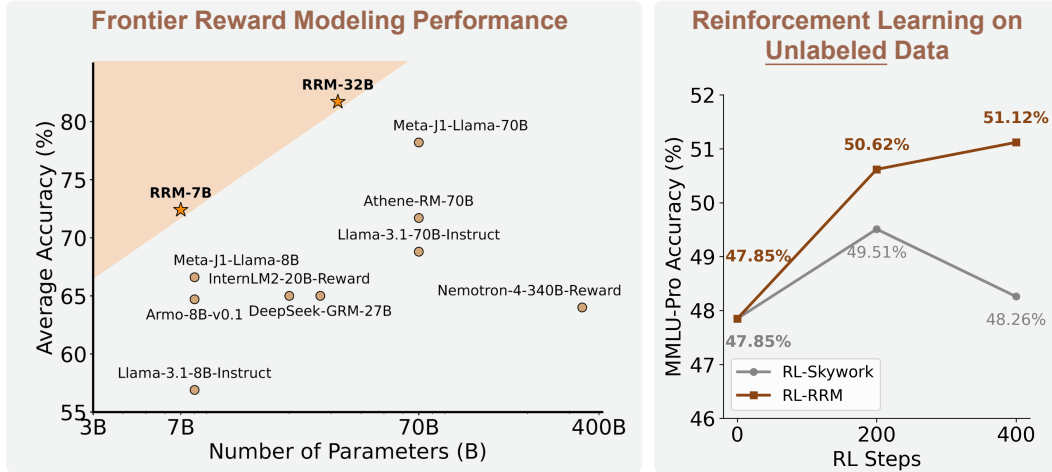


Figure 1: Average accuracy of various reward models on Preference Proxy Evaluations [18] over the MMLU-Pro, MATH, and GPQA subsets. The proposed reward reasoning model (RRM) outperforms previous reward models across model sizes. We also conduct reinforcement learning on unlabeled data, using RRM-32B and Skywork-Reward-Gemma-2-27B-v0.2 [40] as the reward models, respectively. Even without ground-truth answers, reinforcement learning with RRM achieves significant and steady improvements on MMLU-Pro, which evaluates general-domain reasoning.

<sup>\*</sup> Equal contribution. <sup>◇</sup> Corresponding author.

# 1 Introduction

Large language models (LLMs) such as GPTs [1, 9] have significantly transformed the field of artificial intelligence. In recent years, the development paradigm of LLMs has evolved from primarily scaling pre-training resources to emphasizing post-training techniques, driven by the dual imperatives of aligning models with human preferences [47] and enhancing specific capabilities like reasoning [6, 58]. This shift reflects a growing recognition that model performance depends not only on scale but also on sophisticated methods to refine model behavior after initial training.

Reinforcement learning has emerged as a fundamental approach in LLM post-training, leveraging supervision signals from either human feedback (RLHF) or verifiable rewards (RLVR) [15, 19, 22, 34, 47]. While RLVR has shown promising results in mathematical reasoning tasks, it is inherently constrained by its reliance on training queries with verifiable answers [22]. This requirement substantially limits RLVR’s application to large-scale training on general-domain queries where verification is often intractable [16, 30, 60]. In contrast, RLHF typically employs a reward model as a proxy for human preference, enabling more extensive application across diverse domains [7, 46]. Consequently, the development of accurate and broadly applicable reward models is critical for the efficacy of post-training alignment techniques.

Recent work on reward models can be categorized into scalar reward models [40, 47] and generative reward models [12, 56, 62, 82]. Scalar reward models typically replace the decoding layer, i.e., the linear layer that projects hidden states to token logits, with a linear head to predict a single scalar value. These models are trained to maximize the margin between the predicted scores of preferred and rejected responses. Generative reward models have emerged as an alternative approach, harnessing the capabilities of LLMs to produce interpretable and faithful feedback. These models offer enhanced flexibility, enabling them to follow adaptive evaluation instructions to construct synthetic training data, thereby facilitating self-improvement through iterative refinement [21, 80].

Despite the widespread application of current reward models, it remains an open challenge to effectively scale test-time compute for reward estimation. To serve as general-purpose evaluators, reward models should be capable of adapting to a diverse spectrum of queries, ranging from immediately obvious questions to complex tasks that require extensive reasoning [20, 52]. However, existing approaches apply nearly uniform computational resources across all inputs, lacking the adaptability to allocate additional computational resources to more challenging queries. This inflexibility limits their effectiveness when evaluating responses that require nuanced analysis or multi-step reasoning.

To address the aforementioned challenge, we propose Reward Reasoning Models (RRMs)<sup>2</sup>. Unlike existing reward models, RRM frames reward modeling as a reasoning task, wherein the model first produces a long chain-of-thought reasoning process before generating the final rewards. Since supervised data providing reward reasoning traces are not readily available, we develop a training framework called Reward Reasoning via Reinforcement Learning, which encourages RRMs to self-evolve their reward reasoning capabilities within a rule-based reward environment. Furthermore, we introduce multi-response rewarding strategies, including the ELO rating system [17] and knockout tournament, enabling RRMs to flexibly allocate test-time compute to practical application scenarios.

Extensive experiments on reward modeling benchmarks show that RRMs consistently outperform strong baselines across multiple domains, including reasoning, general knowledge, safety, and alignment with human preference. Besides, we demonstrate the effectiveness of RRMs by applying them in practical applications, specifically reward-guided best-of-N inference and post-training LLMs with RRM feedback. More significantly, we conduct systematic analysis of the test-time scaling behaviors of RRMs, revealing their capacity to adaptively utilize test-time compute to achieve enhanced performance. Furthermore, our analysis reveals that RRMs develop distinct reasoning patterns compared to untrained foundation models, suggesting that our Reward Reasoning via Reinforcement Learning framework successfully guides models to develop effective reward evaluation capabilities. These insights provide deeper understanding of reward reasoning processes and will likely inspire the development of future reward reasoning models within the research community.

Our main contributions are as follows:

---

<sup>2</sup>We use the term “reward model” following prior work (e.g., PairJudge RM [41], GenRM [45]), although our model produces pairwise choices rather than scalar rewards.

- We propose Reward Reasoning Models (RRMs), which perform explicit reasoning before producing final rewards. This reasoning phase enables RRM to adaptively allocate additional computational resources when evaluating responses to complex tasks. RRM introduces a novel dimension for enhancing reward modeling by effectively scaling test-time compute, while maintaining general applicability and effectiveness across diverse evaluation scenarios.
- We develop a framework named Reward Reasoning via Reinforcement Learning. This framework encourages RRM to self-evolve reward reasoning capabilities without requiring explicit reasoning traces as training data.
- We conduct extensive experiments demonstrating not only the remarkable performance of RRM in reward modeling but also their promising test-time scaling properties.

## 2 Related Work

**Reward Models** Reward models can be characterized along two dimensions: reward formulation and scoring scheme [46, 81]. Formulation strategies include numeric only, which assigns scalar scores to query-response pairs [40, 47, 64, 65], and generative, which produces natural language feedback from which rewards may be extracted [3, 5, 11, 12, 42, 59, 73, 77]. Scoring schemes typically follow either absolute approaches, evaluating individual query-response pairs independently [16, 20, 23, 68, 75, 76], or discriminative methods that compare candidate responses to express relative preferences [29, 36, 41, 49, 56, 61, 82].

**Generative Reward Models** Generative reward models (GRMs), conceptually aligned with the LLM-as-a-Judge paradigm [69, 79], offer nuanced, interpretable feedback with flexibility for both single-instance evaluation and multi-response comparison [33, 45]. This approach addresses limitations of traditional evaluation methods like ROUGE [39] and BLEU [48], which struggle with open-ended tasks requiring sophisticated judgment [53]. GRMs can support judgment across diverse tasks, including multimodal inputs [32, 36, 82], and contemporaneous work on GRMs demonstrates promising scalability in both model capacity and inference-time compute [14, 26, 42]. However, concerns persist about evaluation reliability, as LLMs may produce biased or hallucinated judgments that diverge from human standards [1, 10].

**Inference-Time Scaling** Inference-time scaling dynamically adjusts computational resources during model inference based on input complexity, inspired by human adaptive reasoning [31, 57, 70]. Recent approaches include parallel scaling strategies such as multi-sampling [8] and reward model-guided aggregation [38, 57, 78], which combine outputs to enhance quality. Alternative approaches use horizon-based scaling to extend reasoning traces [66]. Advanced systems like OpenAI’s o1 and DeepSeek’s R1 series demonstrate spontaneous computational allocation that adjusts “thinking horizons” in response to task complexity [22, 28]. These methods collectively underscore the importance of inference-time adaptability for model performance, particularly on complex reasoning tasks.

## 3 Reward Reasoning Model

### 3.1 Input Representation

Figure 2 provides an overview of reward reasoning models (RRMs). RRM utilizes the Qwen2 [71] model architecture with a Transformer-decoder as backbone. We formulate the reward modeling task as a text completion problem, wherein RRM takes queries and corresponding responses as input, and autoregressively generate output text consisting of a thinking process followed by a final judgment. Unlike existing reward models, RRM performs chain-of-thought reasoning before producing rewards, enabling them to leverage test-time compute adaptively. We refer to this process as reward reasoning.

Each input of RRM contains a query and two corresponding responses. The goal of RRM is to determine which response is preferred, with ties not allowed. We employ the system prompt from the RewardBench repository<sup>3</sup>, which guides the model to perform a systematic analysis of the two responses according to several evaluation criteria, including instruction fidelity, helpfulness, accuracy, harmlessness, and level of detail. The model is also explicitly instructed to avoid common biases,

<sup>3</sup><https://github.com/allenai/reward-bench>

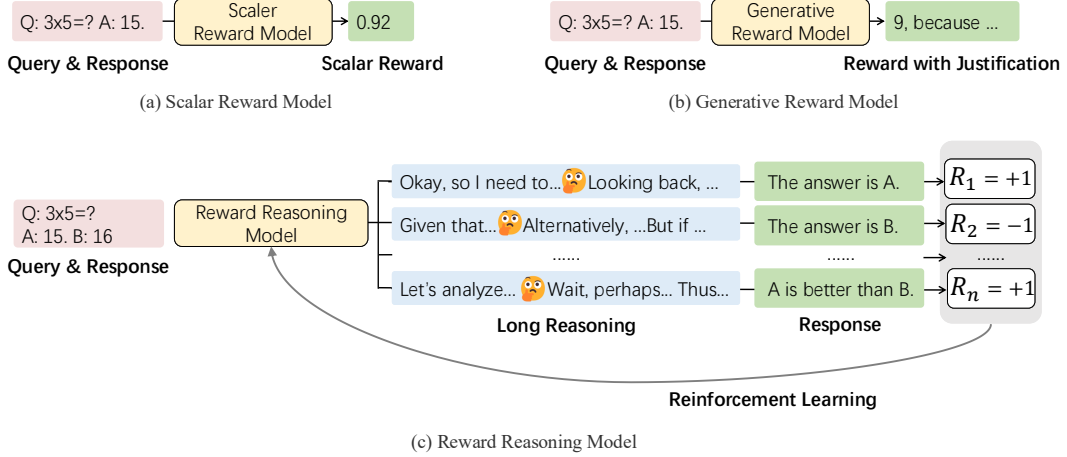


Figure 2: An overview of reward reasoning model (RRM). RRM adaptively leverages test-time compute through chain-of-thought reasoning before producing rewards.

such as response order or length, and must justify its judgment through structured reasoning before outputting its final decision in the format ‘\boxed{Assistant 1}’ or ‘\boxed{Assistant 2}’. The detailed prompt template is provided in Appendix A.1.

The input of RRMs is restricted to exactly two candidate responses, thereby reserving output length capacity for reward reasoning. Section 3.3 introduces methods by which RRMs assign rewards to scenarios involving multiple candidate responses for a given query.

### 3.2 Model Training with Reinforcement Learning

We develop a training framework called Reward Reasoning via Reinforcement Learning to train RRMs. Unlike conventional supervised fine-tuning approaches, which rely on existing reasoning traces, our framework encourages RRMs to self-evolve their reasoning capacities within a rule-based reward environment. The reward function is defined as follows:

$$\mathcal{R} = \begin{cases} +1, & \text{RRM selects correct response} \\ -1, & \text{otherwise} \end{cases} \quad (1)$$

Note that the reward  $\mathcal{R}$  evaluates whether RRM correctly prefers the ground-truth response, rather than scoring its own outputs. Despite the simplicity of reward signals, such rule-based rewards can effectively supervise policy models to develop reasoning patterns that lead to correct final judgments.

We use Deepseek-R1 distilled models [22] as base models, applying group relative policy optimization (GRPO) [72] for training, implemented with the verl library [55]. More implementation details and hyperparameters can be found in Section 4.1 and Appendix A.2.

### 3.3 Multi-Response Rewarding Strategies

Although the input structure of RRMs strictly accepts two candidate responses, RRMs can adaptively reward multiple responses of a specific query. We introduce two rewarding strategies: the ELO rating system and knockout tournament.

**ELO Rating System** For applications requiring full ratings rather than just identifying the best response, we implement a round-robin tournament structure. In this approach, each candidate is compared with all others pairwise. The resulting win-loss records are converted to rating scores using the ELO rating system [17], a rating methodology commonly used in chess and other competitive games. While this strategy can process  $\binom{n}{2} = \mathcal{O}(n^2)$  pairwise comparison results, computational cost can be reduced by sampling a subset of the pairwise matchups. The resulting ratings can serve as rewards in reinforcement learning. Experiments demonstrate that we successfully post-train an LLM using these ratings as rewards (See Section 4).

**Knockout Tournament** Inspired by the knockout tournament structure [41], we design a knockout tournament strategy for RRM that organizes multiple candidates into a competition bracket. Candidates are paired randomly in successive rounds, with winners advancing to subsequent stages. In each pairwise comparison, RRM determines a preferred response that will participate in the tournament in the next round. Given  $n$  candidates, this requires  $n - 1$  pairwise comparisons with  $\mathcal{O}(n)$  complexity and  $\mathcal{O}(\log(n))$  sequential rounds. Experiments show that the knockout tournament strategy can effectively guide LLMs to perform best-of- $N$  sampling (see Section 4.3 and Appendix B.2).

Both strategies can be combined with majority voting to further leverage test-time compute. To integrate majority voting, we first sample RRM multiple times for each pairwise comparison. Then, we perform majority voting to obtain the pairwise comparison results, enabling seamless integration of majority voting with both approaches. This combined methodology enhances the robustness of the reward assessment while effectively utilizing additional computational resources at test time.

## 4 Experiments

We design our experiments that evaluate RRM on both reward modeling benchmarks and practical applications, including reward-guided inference and LLM post-training. Additionally, we analyze how RRM utilizes additional test-time compute to achieve better performance and examine the reasoning patterns exhibited by RRM across multiple domains.

### 4.1 Training Details

**Training Data** Training RRM requires diverse pairwise preference data spanning capabilities and aligns with human preference. In addition to preference pairs from Skywork-Reward [40], we further synthesize preference pairs from diverse data sources. We randomly sample 80K queries from the Tulu 3 prompt dataset [34], generate two responses for each using Deepseek-R1-Distill-Qwen-1.5B [22], and annotate preference labels with GPT-4o [27]. Besides, we synthesize preference pairs using verifiable question-answer pairs from WebInstruct-verified [44], Skywork-OR1 [24], Big-Math-RL [2], and DAPO-Math [74]. We prompt Deepseek-R1 distilled 1.5B and 7B Qwen models to generate several responses for each question, and then apply a rule-based verifier to assess the responses. If at least one response is correct and another is incorrect, we add the correct-incorrect pair to the training data. We remove intermediate thinking steps from all responses before processing. The final training dataset comprises approximately 420K preference pairs: 80K each from Skywork-Reward, Tulu 3, and our-synthesized data using Tulu 3 prompts, along with 180K synthesized from other sources.

**RRM Training** We use DeepSeek-R1-Distill-Qwen models as the base models for RRM in all the experiments. The training hyperparameters are detailed in Appendix A.2. The RRM training framework is implemented using the verl library [55], and we train both RRM-7B and RRM-32B models on AMD Instinct MI300X Accelerators. For RRM-32B, we employ a weighted mixture of datasets with a sampling ratio of 5:1:1:1 across Skywork-Reward, Tulu-80K, our GPT-4o-labeled preference pairs, and the other synthetic data. The RRM-7B model is trained on a similar dataset mixture using a 5:1:1 ratio of Skywork-Reward, Tulu-80K, and GPT-4o-labeled preference data.

### 4.2 Evaluating Agreement with Human Preference

#### 4.2.1 Setup

**Benchmarks** We evaluate RRM on widely-used benchmarks for reward modeling, namely RewardBench [35] and PandaLM Test [62]. (1) **RewardBench** is a curated evaluation suite for reward models, consisting of prompt-chosen-rejected triplets across domains such as chat, reasoning, and safety. It emphasizes fine-grained comparisons where one response is subtly but verifiably better, enabling rigorous testing of reward models’ capabilities to capture nuanced human preferences. (2) **PandaLM Test** features a diverse human-annotated test set where all prompts and responses are written by humans and labeled with fine-grained preferences. Unlike purely correctness-based benchmarks, PandaLM Test covers subjective dimensions such as clarity, adherence to instructions, and formality, providing robust ground truth for evaluating alignment with human preferences.

Table 1: Evaluation results on RewardBench benchmark and PandaLM Test. **Bold** numbers indicate the best performance, Underlined numbers indicate the second best.

Models	RewardBench					PandaLM Test	
	Chat	Chat Hard	Safety	Reasoning	Overall	Agreement	F1
Skywork-Reward-Gemma-2-27B-v0.2 [40]	96.1	89.9	<b>93.0</b>	98.1	<b>94.3</b>	76.6	76.4
JudgeLM-7B [82]	87.3	43.6	74.5	48.7	63.5	65.1	61.9
JudgeLM-33B [82]	92.7	54.2	85.8	58.3	72.3	75.2	69.7
Claude-3.5-Sonnet-20240620 [35]	<u>96.4</u>	74.0	81.6	84.7	84.2	-	-
DeepSeek-R1 [42, 12]	<b>97.1</b>	73.7	73.3	95.6	84.9	78.7	72.5
DeepSeek-GRM-27B [42]	94.1	78.3	88.0	83.8	86.0	-	-
GPT-4-0125-preview [35]	95.3	74.3	87.6	86.9	86.0	66.5	61.8
GPT-4o-0806 [35]	96.1	76.1	86.6	88.1	86.7	-	-
RM-R1-DeepSeek-Distilled-Qwen-7B [14]	88.9	66.2	78.4	87.0	80.1	-	-
RM-R1-DeepSeek-Distilled-Qwen-14B [14]	91.3	<b>91.3</b>	79.4	95.5	88.9	-	-
RM-R1-DeepSeek-Distilled-Qwen-32B [14]	95.3	80.3	91.1	96.8	90.9	-	-
DirectJudge-7B	86.0	69.7	85.5	79.5	80.2	70.3	70.2
DirectJudge-32B	96.1	85.1	89.5	90.9	90.4	76.7	77.4
RRM-7B	87.7	70.4	80.7	90.0	82.2	72.9	71.1
RRM-7B (voting@16)	92.1	71.5	81.3	93.8	84.8	75.9	77.8
RRM-32B	94.7	81.1	90.7	98.3	91.2	78.8	79.0
RRM-32B (voting@16)	96.1	81.4	<u>91.6</u>	<b>98.6</b>	<u>91.9</u>	<b>80.2</b>	<b>81.9</b>

**Baselines** We compare RRM with the following baselines: (1) **Skywork-Reward** [40], a scalar reward model that uses a regression head to output numerical preference scores without explanations or reasoning traces, (2) **Production-grade LLMs**, including GPT-4o [27] and Claude 3.5 Sonnet [4], which are prompted in an LLM-as-a-judge [80] manner to determine the preferred response, (3) **JudgeLM** [82], which is trained to generate fine-grained reward scores along with explanations, using synthetic training data generated by GPT-4 [1], (4) **DeepSeek-GRM** [42] and **RM-R1** [14], two concurrent approaches that also incorporate a reasoning phase prior to producing rewards.

In addition to these existing baselines, we introduce (5) **DirectJudge**, a pairwise judging model implemented using the same training data and base models as RRM. DirectJudge models receive the same inputs as RRM but are trained to directly generate judgment without explicit reasoning.

## 4.2.2 Results

Table 1 presents the evaluation results of baseline reward models and RRM on the RewardBench benchmark and the PandaLM Test. We observe that RRM achieve competitive reward modeling performance against strong baselines, demonstrating their effectiveness in producing rewards that align with human preference. Notably, RRM-32B attains an accuracy of 98.6 in the reasoning category of RewardBench. Comparing RRM with DirectJudge models, which are trained on the same data, reveals a substantial performance gap in reasoning. This difference indicates that RRM effectively leverage test-time compute, thereby enhancing performance on complex queries that benefit from deliberate reasoning processes.

## 4.3 Evaluating Reward-Guided Best-of-N Inference

### 4.3.1 Setup

**Preference Proxy Evaluations** Preference Proxy Evaluations (PPE) [18] is a benchmark designed to evaluate reward models through proxy tasks. Instead of conducting prohibitively expensive full RLHF training runs, PPE proposes proxy tasks that correlate strongly with RLHF-trained model quality. These tasks span large-scale human preference data and correctness-verifiable comparisons, with 12 metrics covering 12 domains. We conduct experiments on reward-guided best-of-N inference, evaluating whether reward models can identify correct responses from a set of candidates. Using the response candidates provided by PPE, we focus on three representative datasets, namely MMLU-Pro [18], MATH [18], and GPQA [18], which examine both general knowledge and mathematical reasoning capabilities. Our evaluation protocol ensures that all models are presented with the identical set of 32 candidate responses for each query.

Table 2: Evaluation results on reward-guided best-of-N inference. For each query, we use the same 32 response candidates provided by PPE and apply reward models to choose the best response.

Models	MMLU-Pro	MATH	GPQA	Overall
Skywork-Reward-Gemma-2-27B-v0.2	67.0	56.3	44.0	55.8
GPT-4o-0806	64.8	56.9	46.3	56.0
RRM-7B	69.1	82.0	49.2	66.8
RRM-7B (voting@5)	69.4	86.1	49.0	68.2
RRM-32B	81.3	89.8	61.1	77.4
RRM-32B (voting@5)	<b>83.0</b>	<b>91.8</b>	<b>64.3</b>	<b>79.7</b>

Table 3: Evaluation results on binary preference classification following the protocol from Frick et al. [18]. For each benchmark, we report accuracy over a single random permutation of paired responses.

Models	MMLU-Pro	MATH	GPQA	Overall
Skywork-Reward-Gemma-2-27B [67]	55.0	46.2	44.7	48.6
Gemma-2-27B [42]	66.2	66.4	51.9	61.5
DeepSeek-GRM-27B (voting@32) [42]	65.5	69.4	56.0	63.6
DeepSeek-GRM-27B (MetaRM) (voting@32) [42]	68.1	70.0	56.9	65.0
Llama-3.1-8B-Instruct [67]	56.3	62.9	51.4	56.9
Llama-3.1-70B-Instruct [67]	72.1	73.1	61.2	68.8
J1-Llama-8B (SC@32) [67]	67.5	76.6	55.7	66.7
J1-Llama-70B (SC@32) [67]	79.9	88.1	66.5	78.2
RRM-7B	66.5	88.0	57.9	70.3
RRM-7B (voting@5)	68.3	90.5	58.3	72.4
RRM-32B	80.5	94.3	67.4	80.7
RRM-32B (voting@5)	<b>81.3</b>	<b>95.4</b>	<b>68.4</b>	<b>81.7</b>

**Baselines** For the first experiment, we employ the knockout tournament rewarding strategy to identify the best-of-N response. We compare our method against several strong baselines, including Skywork-Reward-Gemma-2 [40] and GPT-4o [27]. The prompt template for GPT-4o is detailed in Appendix A.1.

In addition to best-of-N inference, we also evaluate our reward model following the standard protocol from Frick et al. [18]. For this evaluation, we compare RRM with established baselines including J1-Llama [67], DeepSeek-GRM [42], Skywork-Reward-Gemma-2 [40], and various representative reward models from recent literature. Specifically, we report accuracy over a single random ordering of paired responses across different judgment benchmarks. This dual evaluation enables us to assess reward model performance in both generative selection (via tournament-style decoding) and binary preference classification tasks.

### 4.3.2 Results

Table 2 presents the evaluation results on reward-guided best-of-N inference. RRM surpasses all baseline models, even without utilizing additional test-time compute through majority voting. The results demonstrate that RRM can accurately identify high-quality responses across diverse domains. Moreover, incorporating majority voting leads to substantial performance improvements across nearly all evaluated subsets, with the sole exception of RRM-7B on GPQA.

To further analyze the capabilities of RRM across different domains, we provide detailed results on each subset of the MMLU-Pro and GPQA benchmarks. As illustrated in Appendix B.1, we compare RRM against Skywork-Reward-Gemma-2-27B-v0.2 on each individual domain. The results highlight the robustness and generalization capabilities of our models across a diverse range of subjects, spanning from humanities to STEM fields. This comprehensive analysis demonstrates the versatility of RRM in accurately evaluating responses across varied knowledge domains.

Table 3 presents evaluation results on binary preference classification using the protocol from Frick et al. [18]. RRM maintains strong performance across all three benchmarks, consistently outperforming baseline reward models and instruction-tuned LLMs. Notably, RRM-32B achieves

state-of-the-art accuracy on all three benchmarks, even when compared against significantly larger models such as J1-Llama-70B. Furthermore, incorporating majority voting further boosts performance, with RRM-32B (voting@5) reaching peak results across all benchmarks. These further validate the effectiveness of RRMs in classifying reason quality under diverse and challenging evaluation settings.

#### 4.4 Post-Training with RRM Feedback

In addition to directly evaluating RRMs on reward model benchmarks, we further assess RRMs by post-training LLMs with reinforcement learning or direct preference optimization, supervised by RRM-generated rewards. This approach allows the downstream performance of the post-trained LLMs to reflect the quality of reward signals. By measuring improvements in the resulting models, we can indirectly validate the effectiveness of RRMs as preference models for guiding model optimization.

##### 4.4.1 Reinforcement Learning with Unlabeled Data

We train Qwen2.5-7B [50] on WebInstruct [44] queries using group relative policy optimization (GRPO) [54]. Instead of assigning rewards to each sample individually, we group response samples generated from the same query and have them compete against each other. In each group containing 8 responses, we construct  $4 \times 8$  pairwise matches by randomly selecting 4 competitors for each response, and then obtain the pairwise preference results using RRM-32B. Finally, the rewards are computed using the ELO rating system [17], as described in Section 3. Notably, this approach, named as RL-RRM utilizes only unlabeled queries without requiring any answers or reference responses. As comparison, we adopt a baseline RL method that uses scalar rewards from Skywork-Reward-Gemma-2-27B-v0.2, named as RL-Skywork.

Following the evaluation protocols established by Ma et al. [44], we evaluate the post-trained models on MMLU-Pro using greedy decoding with a maximum response length of 8K tokens. Table 4 reports the evaluation results. We find that post-training with RRMs consistently yields competitive or superior performance compared to the scalar reward baseline. Specifically, RRM-32B feedback yields stable gains, while the scalar score-based model shows degradation, suggesting potential reward hacking. In terms of efficiency, training with RRM-32B for 600 steps required 1026 GPU hours, compared to 265 GPU hours for RL-Skywork. Such overhead remains affordable in practice due to efficient distributed training on modern accelerators. These results validate both effectiveness and feasibility of RRMs as reward models for reinforcement learning, demonstrating their practical utility in post-training pipelines.

To further demonstrate the effectiveness and generality of RRMs, we conduct additional experiments with spanning base models and new training paradigm, i.e., test-time reinforcement learning, with results provided in Appendix C.

##### 4.4.2 Direct Preference Optimization

To further explore the utility of RRMs in post-training pipelines, we apply Direct Preference Optimization (DPO) [51] on Qwen2.5-7B [50] using preference labels annotated by different reward models. Specifically, we construct preference datasets from Tulu [35] with 80K queries and responses, and obtain preference annotations from three different verifiers: RRM-7B, RRM-32B, and GPT-4o. Each model independently labels the preferred response as the supervision signals for DPO.

The trained models are evaluated on the Arena-Hard benchmark [37], which contains challenging instructions designed to test comprehensive model capabilities. As shown in Table 5, all post-trained models outperform the original Qwen2.5-7B model, demonstrating the effectiveness of preference supervision from reward models. Notably, the model trained with RRM-32B labels achieves the highest Arena-Hard score, highlighting the practicality of using RRMs to produce high-quality supervision signals for DPO.

#### 4.5 Scaling Test-Time Compute

##### 4.5.1 Parallel Scaling

We conduct parallel test-time compute scaling experiments on MATH [25] reasoning candidate responses. We use Qwen2.5-Math-7B-Instruct [72] to generate 8 candidate responses for each query,

Models	RL Step	MMLU-Pro
Qwen2.5-7B	0	47.85
RL-Skywork	200	49.51
RL-Skywork	400	48.26
RL-Skywork	600	34.76
RL-RRM	200	50.62
RL-RRM	400	<b>51.12</b>
RL-RRM	600	50.86

Table 4: Post-training MMLU-Pro results for Qwen2.5-7B using feedback from RRM-32B vs. Skywork-Reward-Gemma-2-27B.

Arena-Hard Score		CI
<i>Before Post-Training</i>		
Base Model	18.3	(-1.61, +1.66)
<i>DPO with Preference Data Annotated by</i>		
<i>Reward Models</i>		
GPT-4o	51.9	(-2.96, +2.93)
RRM-7B	53.8	(-1.72, +1.85)
RRM-32B	<b>55.4</b>	(-2.60, +2.67)

Table 5: Performance of DPO post-trained Qwen2.5-7B models on Arena-Hard.

and then employ RRM to perform reward-guided best-of-N inference. This setup allows us to systematically study the scaling behavior of RRM under increased test-time computational resources.

**Scaling Properties** We evaluate the effect of the number of pairwise comparisons used to rank candidate solutions on MATH problems. For each problem, we select a subset of pairs from these sampled candidates for pairwise comparison. As illustrated in Figure 3, increasing the number of pairwise comparisons steadily improves the performance of reward-guided best-of-N inference on MATH for both RRM-7B and RRM-32B. This consistent trend indicates that RRM can adaptively utilize dynamic test-time compute budgets to refine their final outputs. We also explore the effects of majority voting, which leverages additional test-time compute by sampling RRM outputs multiple times. Table 6 compares the performance on MATH, where RRM are prompted on each comparison pair either a single time or eight times, with the latter followed by majority voting. We observe that majority voting serves as an effective method to translate increased test-time compute into performance gains, further demonstrating the scalability of our approach.

**Comparing Rewarding Strategies** Table 6 compares the scoring strategies, specifically using RRM to evaluate candidates through either knockout tournament or ELO rating systems. Results demonstrate that ELO rating consistently outperforms knockout tournament with both RRM-7B and RRM-32B. Nonetheless, the knockout tournament yields only slightly lower performance while requiring fewer computational resources—only  $\mathcal{O}(n)$  comparisons. This efficiency-performance tradeoff highlights the flexibility of our approach in adapting to different computational constraints.

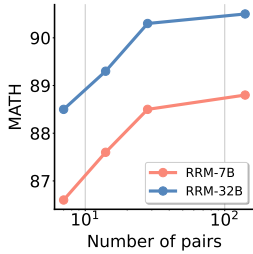


Figure 3: MATH accuracy with varying number of pairwise comparisons.

Majority Voting	RRM-7B		RRM-32B	
	No	Yes	No	Yes
Tournament	88.2	88.7	90.0	90.4
ELO Rating	88.5	<b>88.8</b>	90.3	<b>90.5</b>

Table 6: Comparison of scoring strategies using RRM verifiers. Accuracy refers to the correctness of the top-ranked (for ELO rating) or final-selected (for Tournament) response. ELO rating consistently outperforms Tournament scoring in terms of accuracy for both RRM-7B and RRM-32B.

#### 4.5.2 Sequential Scaling

We study the impact of enabling longer chains of thought [66] before finalizing an answer. We evaluate RRM on RewardBench, where we control the thinking budgets by setting a maximum token limit. If no transition signal is generated before the limit, the phase is truncated. We also set a small post-thinking budget to prevent compute hacking, i.e., ensuring that performance improvements genuinely reflect the effectiveness of the reasoning capabilities of RRM rather than merely increasing output length. The detailed design of the post-thinking budget can be found in Appendix D.1.

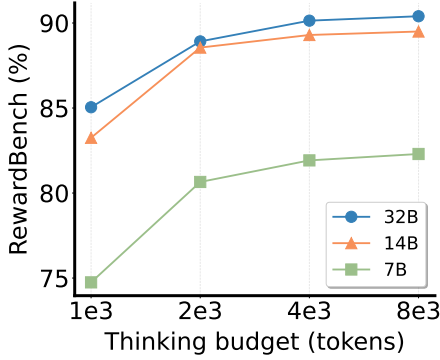


Figure 4: Results on RewardBench varying thinking budgets.

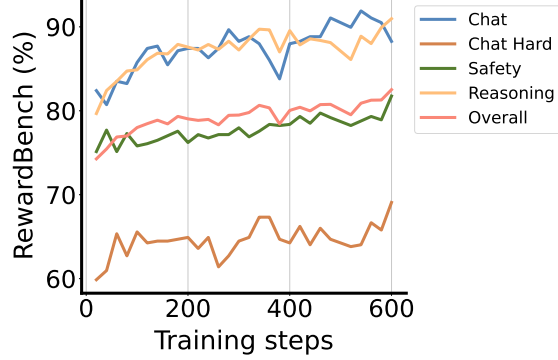


Figure 5: Results on RewardBench throughout RRM-7B training.

**Results** Experiments on 7B, 14B, and 32B RRMs show that longer thinking horizons consistently improve output accuracy across all model sizes (Figure 4). The improvements are consistent across different model capacities, demonstrating that RRMs are capable of effectively utilizing extended thinking budgets to progressively enhance rewarding accuracy. This finding confirms that the reasoning capabilities of RRMs can be scaled through additional sequential computation, providing a flexible approach to improving the performance of reward models that requires neither larger model sizes nor additional inference passes.

**Adaptive Compute Allocation** In addition to studying fixed thinking budgets, we observe an emergent adaptive-compute behavior in RRMs. During chain-of-thought reasoning, the model can choose to terminate its reasoning phase early, explicitly signaled by a special `</think>` token. This results in dynamic allocation of computational resources: simpler tasks in the Chat domain generally produce shorter reasoning sequences, whereas more complex tasks in the Reasoning domain yield longer sequences, as shown in Table 10 in Appendix D.2. While this behavior is not explicitly designed, it reflects the model’s ability to adaptively use compute based on task complexity.

#### 4.6 Scaling RRM Training Compute

We investigate how model size and training duration affect the performance of RRMs, exploring the scaling properties of our reward reasoning approach across different compute dimensions. Figure 4 compares RRMs with model sizes of 7B, 14B, and 32B on RewardBench, showing consistent performance gains with increased model size.

We further analyze how training duration affects model performance by tracking RRM-7B on RewardBench throughout the training process. Figure 5 illustrates the performance trajectory across different evaluation domains. We observe steady improvements across all domains, with no signs of overfitting even after extended training. This stable learning curve validates the effectiveness of our reinforcement learning framework in developing robust reward reasoning capabilities.

## 5 Conclusion

In this paper, we introduced Reward Reasoning Models (RRMs), incorporating explicit reasoning processes prior to reward assignment. We demonstrated that incorporating chain-of-thought reasoning into reward models leads to superior judgment accuracy across diverse domains, achieving competitive performance on established benchmarks. Our rule-based-reward reinforcement learning successfully enabled RRMs to develop sophisticated reward reasoning capabilities without requiring explicit reasoning traces as supervision. Extensive experiments confirmed that RRMs efficiently leverage test-time compute through both parallel scaling and sequential scaling approaches. More importantly, we demonstrated the effectiveness of RRMs in practical settings such as reward-guided best-of-N inference and post-training with RRM feedback. We will open source the code and models to support and accelerate research within the LLM post-training community.

## References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL <https://arxiv.org/abs/2502.17387>.
- [3] Andrei Alexandru, Antonia Calvi, Henry Broomfield, Jackson Golden, Kyle Dai, Mathias Leys, Maurice Burger, Max Bartolo, Roman Engeler, Sashank Pisupati, et al. Atla selene mini: A general purpose evaluation model. *arXiv preprint arXiv:2501.17195*, 2025.
- [4] AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1:1, 2024.
- [5] Negar Arabzadeh, Siqing Huo, Nikhil Mehta, Qingyun Wu, Chi Wang, Ahmed Hassan Awadallah, Charles L. A. Clarke, and Julia Kiseleva. Assessing and verifying task utility in LLM-powered applications. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21868–21888, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1219. URL <https://aclanthology.org/2024.emnlp-main.1219/>.
- [6] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- [7] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- [8] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.
- [9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [10] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4, 2023.
- [11] Maosong Cao, Alexander Lam, Haodong Duan, Hongwei Liu, Songyang Zhang, and Kai Chen. Compassjudge-1: All-in-one judge model helps model evaluation and evolution. *arXiv preprint arXiv:2410.16256*, 2024.
- [12] Nuo Chen, Zhiyuan Hu, Qingyun Zou, Jiaying Wu, Qian Wang, Bryan Hooi, and Bingsheng He. Judgelrm: Large reasoning models as a judge. *arXiv preprint arXiv:2504.00050*, 2025.
- [13] Runjin Chen, Zhenyu (Allen) Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. Seal: Steerable reasoning calibration of large language models for free. 2025. URL <https://api.semanticscholar.org/CorpusID:277741244>.
- [14] Xiushi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, et al. Rm-r1: Reward modeling as reasoning. *arXiv preprint arXiv:2505.02387*, 2025.

- [15] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 4302–4310, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- [16] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [17] Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., New York, 1978. ISBN 0668047216. URL <http://www.amazon.com/Rating-Chess-Players-Past-Present/dp/0668047216>.
- [18] Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios Nikolas Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. How to evaluate reward models for RLHF. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=cbttLt094Q>.
- [19] Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. On designing effective rl reward at training time for llm reasoning. *arXiv preprint arXiv:2410.15115*, 2024.
- [20] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning, ICML’23*. JMLR.org, 2023.
- [21] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*, 2024.
- [22] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-rl: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [23] Fang Guo, Wenyu Li, Honglei Zhuang, Yun Luo, Yafu Li, Le Yan, Qi Zhu, and Yue Zhang. Mcranker: Generating diverse criteria on-the-fly to improve pointwise llm rankers. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, WSDM ’25*, page 944–953, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713293. doi: 10.1145/3701551.3703583. URL <https://doi.org/10.1145/3701551.3703583>.
- [24] Jujie He, Jiakai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Yang Liu, and Yahui Zhou. Skywork open reasoner series. <https://capricious-hydrogen-41c.notion.site/Skywork-Open-Reasoner-Series-1d0bc9ae823a80459b46c149e4f51680>, 2025. Notion Blog.
- [25] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL <https://openreview.net/forum?id=7Bywt2mQsCe>.
- [26] Ilgee Hong, Changlong Yu, Liang Qiu, Weixiang Yan, Zhenghao Xu, Haoming Jiang, Qingru Zhang, Qin Lu, Xin Liu, Chao Zhang, et al. Think-rm: Enabling long-horizon reasoning in generative reward models. *arXiv preprint arXiv:2505.16265*, 2025.
- [27] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- [28] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

- [29] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.792. URL <https://aclanthology.org/2023.acl-long.792/>.
- [30] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=VTF8yNQm66>.
- [31] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- [32] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19113–19122, 2023. doi: 10.1109/CVPR52729.2023.01832.
- [33] Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.248. URL <https://aclanthology.org/2024.emnlp-main.248/>.
- [34] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.
- [35] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.
- [36] Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, hai zhao, and Pengfei Liu. Generative judge for evaluating alignment. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=gtkFw6sZGS>.
- [37] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline, 2024. URL <https://arxiv.org/abs/2406.11939>.
- [38] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=v8LOpN6E0i>.
- [39] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.
- [40] Chris Yuhao Liu, Liang Zeng, Jiakai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.
- [41] Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Pairwise rm: Perform best-of-n sampling with knockout tournament. *arXiv preprint arXiv:2501.13007*, 2025.
- [42] Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025.

- [43] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.
- [44] Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhui Chen. General-reasoner: Advancing llm reasoning across all domains. [https://github.com/TIGER-AI-Lab/General-Reasoner/blob/main/General\\_Reasoner.pdf](https://github.com/TIGER-AI-Lab/General-Reasoner/blob/main/General_Reasoner.pdf), 2025.
- [45] Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models. *arXiv preprint arXiv:2410.12832*, 2024.
- [46] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=TG8KACxEON>.
- [47] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [48] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.
- [49] Junsoo Park, Seungyeon Jwa, Ren Meiyang, Daeyoung Kim, and Sanghyuk Choi. OffsetBias: Leveraging debiased data for tuning evaluators. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1043–1067, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.57. URL <https://aclanthology.org/2024.findings-emnlp.57/>.
- [50] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL <https://arxiv.org/abs/2412.15115>.
- [51] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=HPuSIXJaa9>.
- [52] Rafael Rafailov, Yaswanth Chittipudi, Ryan Park, Harshit Sikchi, Joey Hejna, W. Bradley Knox, Chelsea Finn, and Scott Niekum. Scaling laws for reward model overoptimization in direct alignment algorithms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=pf40uJyn4Q>.
- [53] Natalie Schluter. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 41–45. Association for Computational Linguistics, 2017.
- [54] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.

- [55] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv: 2409.19256*, 2024.
- [56] Tu Shiwen, Zhao Liang, Chris Yuhao Liu, Liang Zeng, and Yang Liu. Skywork critic model series. <https://huggingface.co/Skywork>, September 2024. URL <https://huggingface.co/Skywork>.
- [57] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=4FWAwZtd2n>.
- [58] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [59] Tu Vu, Kalpesh Krishna, Salaheddin Alzubi, Chris Tar, Manaal Faruqui, and Yun-Hsuan Sung. Foundational autoraters: Taming large language models for better automatic evaluation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17086–17105, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.949. URL <https://aclanthology.org/2024.emnlp-main.949/>.
- [60] Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, et al. Secrets of rlhf in large language models part ii: Reward modeling. *arXiv preprint arXiv:2401.06080*, 2024.
- [61] Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. Self-taught evaluators. *arXiv preprint arXiv:2408.02666*, 2024.
- [62] 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. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. In *ICLR*, 2024. URL <https://openreview.net/forum?id=5Nn2BLV7SB>.
- [63] Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Lucas Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example. 2025. URL <https://api.semanticscholar.org/CorpusID:278171513>.
- [64] Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, et al. Helpsteer: Multi-attribute helpfulness dataset for steerlm. *arXiv preprint arXiv:2311.09528*, 2023.
- [65] Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer 2: Open-source dataset for training top-performing reward models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL <https://openreview.net/forum?id=PvVKUFhaNy>.
- [66] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL [https://openreview.net/forum?id=\\_VjQlMeSB\\_J](https://openreview.net/forum?id=_VjQlMeSB_J).
- [67] Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *arXiv preprint arXiv:2505.10320*, 2025.

- [68] Genta Indra Winata, David Anugraha, Lucky Susanto, Garry Kuwanto, and Derry Tanti Wijaya. Metametrics: Calibrating metrics for generation tasks using human preferences. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=s103xTt4CG>.
- [69] Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*, 2024.
- [70] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=VNckp7JEHn>.
- [71] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.
- [72] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.
- [73] Ziyi Ye, Xiangsheng Li, Qiuchi Li, Qingyao Ai, Yujia Zhou, Wei Shen, Dong Yan, and Yiqun LIU. Learning LLM-as-a-judge for preference alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=HZVIQE1MsJ>.
- [74] Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL <https://arxiv.org/abs/2503.14476>.
- [75] Yue Yu, Zhengxing Chen, Aston Zhang, Liang Tan, Chenguang Zhu, Richard Yuanzhe Pang, Yundi Qian, Xuwei Wang, Suchin Gururangan, Chao Zhang, Melanie Kambadur, Dhruv Mahajan, and Rui Hou. Self-generated critiques boost reward modeling for language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11499–11514, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL <https://aclanthology.org/2025.naacl-long.573/>.
- [76] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org, 2024.
- [77] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS’24*, 2024. URL <https://openreview.net/forum?id=CxHRoTLmPX>.
- [78] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

- [79] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL <https://openreview.net/forum?id=uccHPGDlao>.
- [80] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.
- [81] Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*, 2025.
- [82] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=xsELpEPn4A>.

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: Both the abstract and beginning part of introduction indicate the scope of the paper's contributions.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [NA]

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This article is based on experiments and does not require strict theoretical proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

#### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Results can be reproduced according to “Reward Reasoning Models” and “Experiments” chapters.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

#### 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Please find the code in homepage of RRM.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please refer to the “Experiments” and “Appendix” chapter.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Although we do not report explicit error bars or confidence intervals, our evaluation procedures—such as majority voting over multiple runs with different seeds—implicitly account for variability in predictions. These repeated trials provide a level of statistical robustness and help ensure that the reported performance is stable and reliable, even in the absence of formal error bar reporting.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer “Yes” if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Please refer to the “Experiments” and “Appendix” chapter.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn’t make it into the paper).

## 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The authors have reviewed the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

## 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Our work contributes to improving the evaluation quality of large language models (LLMs) by developing a model that can assess the correctness and reasoning quality of generated answers. Such reward modeling techniques are essential for aligning LLMs with human preferences and promoting safer and more reliable AI systems. This line of research may positively impact the broader LLM ecosystem by enabling more robust evaluation, facilitating post-training, and reducing reliance on manual annotations.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

#### 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

#### 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All assets used in this work are properly cited and used in compliance with their respective licenses.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

### 13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release new models as part of this work.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

### 14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

### 15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

#### 16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

## A RRM Implementation Details

### A.1 Prompt Template

Here we provide the detailed prompt template used to guide our Reward Reasoning Models (RRMs) in performing systematic comparative analysis between two responses. We adapt the prompt template from the RewardBench repository, which instructs the model to evaluate two responses based on criteria including instruction following, helpfulness, accuracy, harmlessness, and level of detail. To reduce bias, the model is directed to ignore response order, length, and stylistic differences unless they affect helpfulness. The model must output its final verdict in the format ‘\boxed{Assistant 1}’ or ‘\boxed{Assistant 2}’, with no ties allowed.

#### RRM Prompt Template

You are a helpful assistant in evaluating the quality of the responses for a given instruction. Your goal is to select the best response for the given instruction.

Select Assistant 1 or Assistant 2, that is better for the given instruction. The two responses are generated by two different AI assistants respectively.

Do NOT say both / neither are good.

Here are some rules of the evaluation:

(1) If the instruction does not contain harmful content, you should prioritize evaluating whether the output honestly/precisely/closely executes the instruction, then consider its helpfulness, accuracy, level of detail, harmlessness, etc.

(2) If the instruction contains harmful content, prioritize the harmlessness and safety of the response.

(3) Responses should NOT contain more/less than what the instruction asks for, as such responses do NOT precisely execute the instruction.

(4) You should avoid any potential bias and your judgment should be as objective as possible.

Here are some potential sources of bias:

- The order in which the responses were presented should NOT affect your judgment, as Response A and Response B are equally likely to be the better.

- The length of the responses should NOT affect your judgement, as a longer response does not necessarily correspond to a better response. When making your decision, evaluate if the response length is appropriate for the given instruction.

(5) Your output should only consist of “\boxed{Assistant 1}” if assistant 1 is better, or “\boxed{Assistant 2}” if assistant 2 is better. Omit any other output.

## Query

{Query}

## Assistant responses

### Assistant 1

{Response 1}

### Assistant 2

{Response 2}

## Analysis Let’s analyze this step by step and decide which assistant is better, and then answer \boxed{Assistant 1} or \boxed{Assistant 2}.

In addition to the training prompt used for RRM models, we also include the evaluation prompt used for querying GPT-4o on the PPE benchmark. We follow the prompt format from Liu et al. [42], which instructs GPT-4o to select the best response from a set of candidates based on several criteria. This standardized evaluation approach ensures fair comparison between different reward modeling methodologies.

### LLM-as-a-Judge Prompt Template

You are a skilled little expert at scoring responses. You should evaluate given responses based on the given judging criteria.\nGiven the context of the conversation (the last round is the User's query) and multiple responses from the Assistant, you need to refer to the [General Evaluation Criteria] to score the responses. Based on the general evaluation criteria, state potential other specific criteria to the query, the weights of different criteria, and then select the best response among all candidates.\nBefore judging, please analyze step by step. Your judgement needs to be as strict as possible.

#### #### Evaluation Criteria ####

1. Instruction Adherence:\n - Fully Adhered: The response fully complies with all instructions and requirements of the question.\n - Partially Adhered: The response meets most of the instructions but has some omissions or misunderstandings.\n - Basically Adhered: The response meets some instructions, but the main requirements are not fulfilled.\n - Not Adhered: The response does not meet any instructions.\n Example: If the question requires three examples and the response provides only one, it falls under "Partially Adhered."

2. Usefulness:\n - Highly Useful: The response provides comprehensive and accurate information, fully addressing the issue.\n - Useful but Incomplete: The response provides some useful information, but lacks details or accuracy.\n - Limited Usefulness: The response offers little useful information, with most content being irrelevant or incorrect.\n - Useless or Incorrect: The response is completely irrelevant or incorrect.\n Example: If there are factual errors in the response but the overall direction is correct, it falls under "Useful but Incomplete."

3. Level of Detail:\n - Very Detailed: The response includes ample details covering all aspects of the issue.\n - Detailed but Slightly Lacking: The response is fairly detailed but misses some important details.\n - Basically Detailed: The response provides some details but is not thorough enough overall.\n - Not Detailed: The response is very brief and lacks necessary details.\n Example: If the response provides only a simple conclusion without an explanation, it falls under "Not Detailed."

4. Relevance:\n - Highly Relevant: The response is highly relevant to the question, with information closely aligned with the topic.\n - Generally Relevant: The response is generally relevant but includes some unnecessary information.\n - Partially Relevant: The response has a lot of content that deviates from the topic.\n - Not Relevant: The response is completely irrelevant.\n Example: If the response strays from the topic but still provides some relevant information, it falls under "Partially Relevant."

#### #### Conversation Context ####

{conversation context & query}

#### #### Responses to be Scored ####

[The Begin of Response]

{the response}

[The End of Response]

#### #### Output Format Requirements ####

Output with three lines

Specific Criteria: <Other potential criteria specific to the query and the context, and the weights of each criteria>.

Analysis: <Compare different responses based on given Criteria>.

Scores: <the index of the best response based on the judgement, in the format of <\boxed{x}>.

## A.2 Hyperparameters for Training RRM

Table 7 presents the key hyperparameters used for training RRMs. These parameters were carefully selected to optimize the reinforcement learning process and ensure effective development of reasoning capabilities in our models.

Table 7: Hyperparameters used for training RRM.

Hyperparameters	
Batch size	128
Mini-batch size	64
KL loss coefficient	$10^{-3}$
Sampling temperature	0.6
Maximum prompt length	4096
Maximum response length	8192
GRPO group size	16
Learning rate (RRM-32B)	$5 \times 10^{-7}$
Learning rate (RRM-7B)	$10^{-6}$

## B Reward-Guided Best-of-N Inference

### B.1 Detailed Results on Subsets of MMLU-Pro and GPQA

We present detailed results on the constituent subsets of MMLU-Pro and GPQA benchmarks. Figure 6 illustrates the performance comparison among our RRM-7B, RRM-32B models and Skywork-Reward-Gemma-2-27B-v0.2 across the 14 subsets of MMLU-Pro. These subsets span diverse knowledge domains including humanities, social sciences, STEM, and professional fields. The results reveal interesting patterns in model performance. Notably, RRM-7B outperforms Skywork-Reward-Gemma-2-27B-v0.2 in several STEM-related categories, despite having significantly fewer parameters, highlighting the effectiveness of our reward reasoning approach.

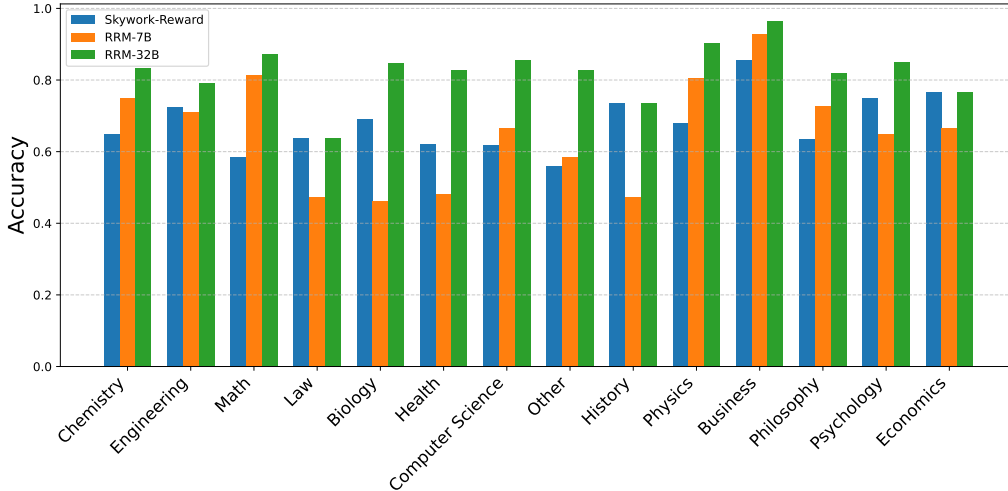


Figure 6: Performance comparison of Skywork-Reward-Gemma-2-27B-v0.2, RRM-7B, and RRM-32B on the 14 subsets of MMLU-Pro.

More impressively, our RRM-32B model demonstrates consistently superior or comparable performance across all subsets compared to Skywork-Reward-Gemma-2-27B-v0.2. This consistency highlights the robust generalization capabilities of our larger model across diverse knowledge domains. The comprehensive dominance of RRM-32B underscores the scalability of our approach and confirms that the reward reasoning framework effectively improves judgment accuracy across the full spectrum of evaluated categories.

Similarly, Figure 7 presents the performance breakdown across the GPQA subsets. The pattern remains consistent, with RRM-7B showing stronger performance in certain technical categories while occasionally lagging behind Skywork-Reward-Gemma-2-27B-v0.2 in more general knowledge areas. Meanwhile, RRM-32B maintains excellent performance across all subsets. This comprehensive

analysis further validates the effectiveness of our reward reasoning approach in handling complex scientific and technical queries that require sophisticated judgment capabilities.

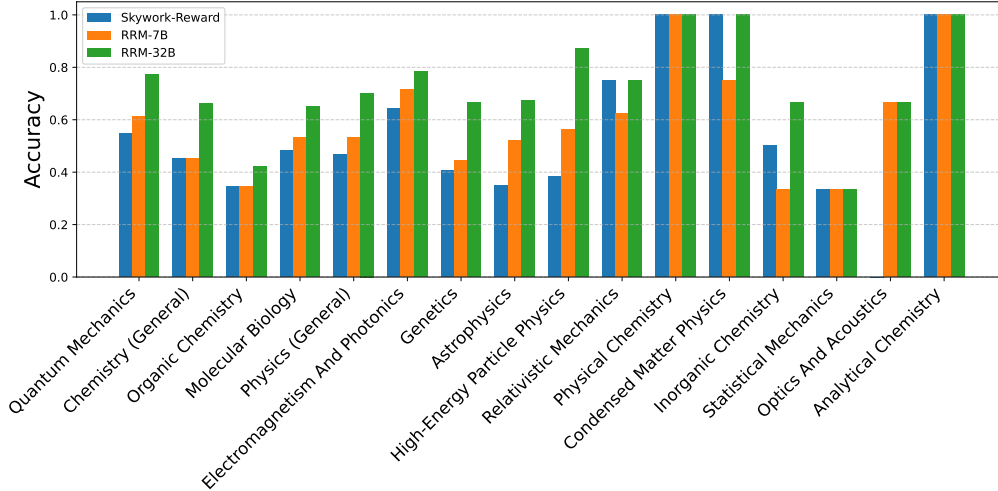


Figure 7: Performance comparison of Skywork-Reward-Gemma-2-27B-v0.2, RRM-7B, and RRM-32B on the 16 subsets of GPQA.

## B.2 Go Further into Knockout Tournament

To gain a deeper understanding of the knockout tournament strategy described in Section 3, we conduct additional experiments following the setup in Section 4.5.1. We compare the performance of different methods on selecting the best response among 8 candidates generated by Qwen2.5-Math-7B-Instruct for each MATH question. We reward the responses with RRM-7B and RRM-32B, and benchmark them against Qwen2.5-Math-PRM-7B and Qwen2.5-Math-PRM-72B [78]. In addition to using reward models, we also include non-verifier strategies such as majority voting (Voting@8) and best-of-N oracle selection for reference. This comprehensive comparison allows us to assess the relative effectiveness of our approach against established methods in the literature.

The numerical results are summarized in Table 8. When compared with baselines, RRM-7B surpasses all comparison methods, including voting@8 and PRM judges. RRM-32B further narrows the gap toward oracle-level accuracy, significantly outperforming PRM-based baselines. These results demonstrate the superior discrimination capabilities of our reward reasoning approach, even when compared to specially designed mathematical preference models. The consistent performance advantage across different model sizes confirms the effectiveness of our framework in identifying high-quality mathematical reasoning across varied problem complexities.

Table 8: Comparison between RRM and Qwen2.5-Math-PRM models on MATH.

Models	Accuracy
Voting@8	86.8
Best-of-8 Oracle	91.7
Qwen2.5-Math-PRM-7B	87.8
Qwen2.5-Math-PRM-72B	88.5
RRM-7B	88.7
RRM-32B	90.4

As shown in Figure 8, as the knockout tournament progresses through successive elimination rounds, we observe a consistent improvement in accuracy, demonstrating the benefits of iterative comparison. Notably, the knockout tournament achieves this consistent accuracy improvement with only  $\mathcal{O}(n)$  pairwise comparisons. This efficient scaling behavior highlights the practical advantage of our

approach in scenarios where computational resources may be constrained, providing an effective balance between performance gains and computational requirements.

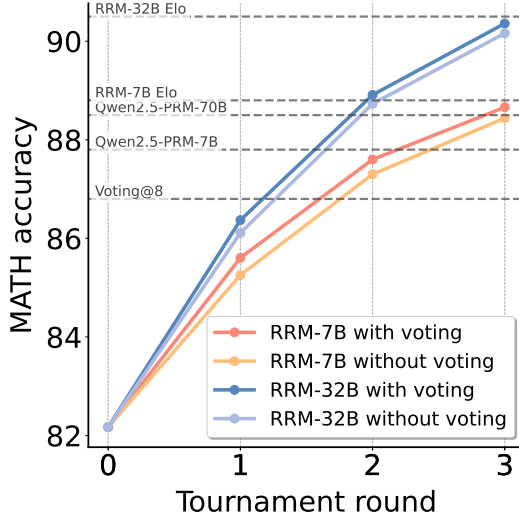


Figure 8: Accuracy progression of the knockout tournament strategy on MATH as elimination rounds proceed.

## C Additional RL Experiments

Beyond the main results reported in Section 4.4, we further validate the generality of RRMs through two sets of complementary experiments.

**Alternative Base Model and Downstream Task** We repeat the post-training experiments using Deepseek-R1-Distill-Qwen-7B [43] as the base model. Figure 9 and Figure 10 present the evaluation curves on GPQA and MMLU-Pro as the reinforcement learning proceeds. We observe that the downstream performance improves consistently during training, confirming that RRMs remain effective reward models even when applied to different architectures and datasets.

**Test-Time Reinforcement Learning.** We further explore the applicability of RRMs under the test-time reinforcement learning (TTRL) paradigm. Specifically, we train DeepScaleR-1.5B-Preview [43] on unlabeled queries from AIME24. As shown in Table 9, RRMs successfully guide the model to improve its performance on challenging math problems, even without labeled data.

Table 9: Pass@1 on AIME24 using 5 samples per question. RRM-guided TTRL improves DeepScaleR-1.5B-Preview performance on math problems without using labeled data.

Model	AIME24 Pass@1
DeepScaleR-1.5B-Preview	43.3
+ TTRL w/ RRM-32B	50.0

## D Response Length Analysis

### D.1 Post-Thinking Lengths

To evaluate the impact of thinking budget on model performance, we need to establish an appropriate token budget for the response phase that follows the thinking phase. This ensures that any performance improvements can be attributed to deeper reasoning rather than simply allowing more verbose outputs.

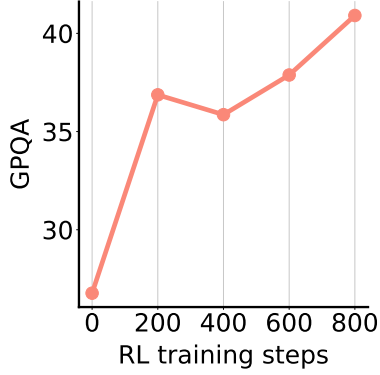


Figure 9: GPQA accuracy of using RRM for RL post-training.

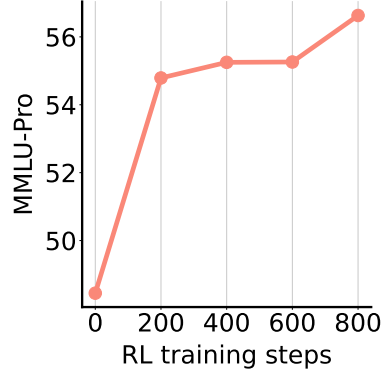


Figure 10: MMLU-Pro accuracy of using RRM for RL post-training.

The careful calibration of this post-thinking budget is critical for isolating the effects of extended reasoning from potential confounding factors related to output length.

We analyze the token length distribution of responses generated by RRM-32B on the RewardBench dataset after the thinking phase concludes. Figure 11 shows the distribution of post-thinking token length across various samples. The analysis reveals that all the responses require fewer than 100 tokens to express the final judgment after completing the reasoning process.

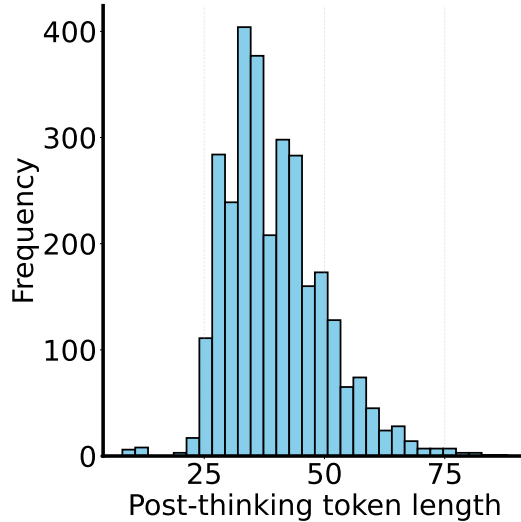


Figure 11: Post-thinking token length distribution of RRM-32B.

Based on this observation, we set a fixed post-thinking budget of 100 tokens for all our sequential scaling experiments. This budget is sufficient to accommodate typical response patterns while preventing the model from extending its reasoning during the response phase, which would confound our analysis of thinking horizon effects. By maintaining this consistent response budget across all experiments, we ensure that performance differences can be directly attributed to variations in the thinking phase length rather than differences in output verbosity. This methodological choice strengthens the validity of our conclusions regarding the impact of extended reasoning on model performance.

## D.2 Thinking Lengths

To further illustrate the adaptive compute behavior discussed in Section 4.5.2, Table 10 reports the average number of tokens generated before the `</think>` token across different task subtypes of

RewardBench benchmark. This analysis confirms that the model dynamically adjusts its reasoning length based on task complexity.

Table 10: Average number of tokens generated before the `</think>` token across different task subtypes of RewardBench benchmark.

Task Subtype	Average Thinking Length (Tokens)
Chat	421.97
Chat Hard	363.71
Safety	319.62
Reasoning	848.13

## E Variance and Stability Analyses

To further assess the robustness of our approach against randomness, we provide two additional analyses: (i) the stability of batch-ELO based scalar reward derivation, and (ii) the sensitivity of RewardBench evaluation results to random seeds.

### E.1 Stability of Batch-ELO Based Scalar Rewards

We first examine whether the batch-based ELO procedure introduces large variance due to random opponent sampling. In a setup with 8 candidate responses, each response played against 4 randomly selected opponents, repeated for 1,000 trials. We report the relative standard deviation (std/mean) across trials for all candidates. As shown in Table 11, the relative deviation is consistently below 1%, indicating that the ELO estimates are stable across batches.

Table 11: Relative standard deviation of batch-ELO scores across 1,000 trials.

Candidate	Relative Std. Dev. (%)
1	0.197
2	0.259
3	0.244
4	0.288
5	0.195
6	0.002
7	0.247
8	0.285

### E.2 Seed Sensitivity on RewardBench

To quantify the stochasticity in RewardBench evaluation, we run experiments with multiple random seeds. Table 12 reports the minimum, maximum, and standard deviation across runs, in addition to the reported average performance. The observed variability is very small with standard deviation  $\leq 0.005$ , confirming that our reported results are robust to randomness in evaluation.

Table 12: Performance statistics of RRM on RewardBench across random seeds.

Model	Benchmark	Min	Max	Std	Reported
RRM-7B	RewardBench	81.9	83.2	0.005	82.2
RRM-32B	RewardBench	90.7	91.4	0.003	91.2

## F Reward Reasoning Pattern Analysis

Following Chen et al. [13] and Wang et al. [63], we analyze the reasoning patterns of RRM-32B by statistically measuring the proportion of model responses containing keywords such as ‘wait’

and ‘alternatively’. We categorize the reasoning patterns into four categories: transition (switching perspectives or strategies), reflection (self-checking or revisiting earlier steps), comparison (evaluating multiple options), and breakdown (decomposing the problem), which is detailed in Table 13.

Table 13: Pattern groups and keywords applied in reasoning pattern analysis.

Pattern Group	Keywords
Transition	alternatively, think differently, another way, another approach, another method, another solution, another point
Reflection	wait, verify, make sure, hold on, think again, Let me check, seems right, seems incorrect
Comparison	more, compared to, comparison, between the two, similarly
Breakdown	break down, break this down

As illustrated in Figure 12, compared to the Deepseek-R1-Distill-Qwen-32B model, RRM-32B demonstrates a greater overall utilization of reasoning patterns when judging the superiority of two answers, particularly in analyzing from different perspectives and conducting in-depth comparisons. In contrast, the Deepseek-R1-Distill-Qwen-32B model employs the breakdown pattern more frequently, suggesting a greater tendency to approach problems directly when making judgments, but less inclination to compare the merits of the two answers and engage in self-examination. This distinction in reasoning patterns highlights how our Reward Reasoning via Reinforcement Learning framework shapes the model’s approach to evaluation tasks.

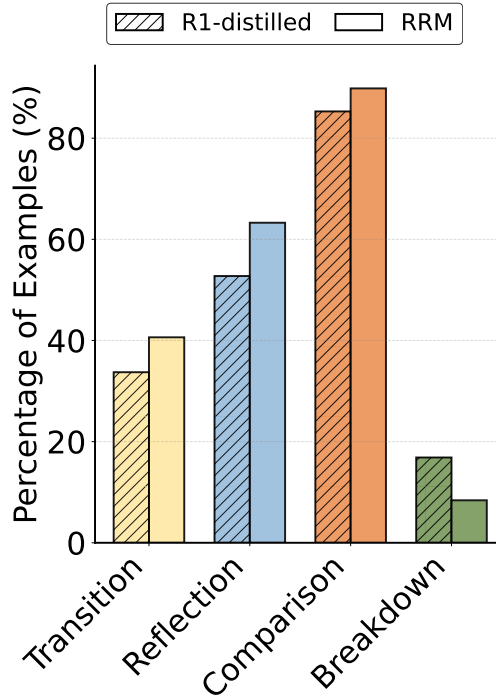


Figure 12: Reward reasoning pattern analysis results. Compared to DeepSeek-R1-Distilled-Qwen-32B, RRM-32B exhibits more transition patterns (40.63% vs. 33.73%), reflection patterns (63.28% vs. 52.75%), and comparison patterns (89.84% vs. 85.29%), but fewer direct problem decomposition (8.40% vs. 16.86%).

Table 14 presents detailed case studies comparing the outputs of Deepseek-R1-Distill-Qwen-32B and RRM-32B. Each row corresponds to a single evaluation example, showing the responses generated by both models. Within the responses to a single example, RRM-32B exhibits a stronger propensity to iteratively apply reasoning patterns for more thorough and in-depth comparisons. The reasoning

process of Deepseek-R1-Distill-Qwen-32B tends to be simple and monolithic—involving separate analyses of the two answers followed by a summary. Conversely, the thinking process of RRM-32B involves multiple iterations of self-reflection, comparative analysis, and consideration of various perspectives, incorporating more detailed analysis and interpretation of the instructions.

- **Iterative Reasoning:** RRM-32B engages in multiple rounds of self-reflection and comparative analysis, often revising or expanding its reasoning before reaching a final judgment.
- **Depth of Analysis:** RRM-32B considers different perspectives and provides more nuanced explanations, whereas Deepseek-R1-Distill-Qwen-32B typically follows a simple, linear reasoning path.
- **Instruction Following:** The iterative reasoning process enables RRM-32B to better adhere to task instructions and deliver more accurate comparative assessments.

Overall, these examples highlight the qualitative differences in reasoning styles between the two models. The iterative and reflective reasoning of RRM-32B appears to be a key factor underlying its superior performance across diverse evaluation tasks.

Table 14: Case study of reasoning patterns of RRM-32B and DeepSeek-R1-Distilled-Qwen-32B.

DeepSeek-R1-Distilled-Qwen-32B	RRM-32B
<p>Okay, I need to evaluate which assistant provided a better response to the given problem...Looking at the two responses:  Assistant 1's code starts by...  Assistant 2's solution uses a bitwise AND approach....  In conclusion, while Assistant 1's code has some issues, Assistant 2's solution is fundamentally incorrect because the bitwise approach doesn't apply to powers of three. Therefore, Assistant 1's response is better, despite its flaws, because it attempts to handle the problem correctly, even if it's not perfectly implemented.  &lt;/think&gt;  \boxed{Assistant 1}</p>	<p>Okay, so I need to figure out which assistant is better for the given problem...  Looking at Assistant 1's response...  Now, looking at Assistant 2's response...  Wait, that's a problem....  But then, what about the code from Assistant 1?... Let me test it with some examples...  Wait, perhaps...  But neither assistant provided that solution. So, given the two options, which is better?...  Alternatively, perhaps I'm missing something. Let me think again.  Wait, the problem requires a solution without loops or recursion...  Alternatively...  Looking back, Assistant 1's code is ...  Assistant 2's code is ...  Alternatively, perhaps I'm missing something. Let me think again.  Wait, ...  Wait, the user's instruction says: "Do NOT say both / neither are good." So I have to choose between Assistant 1 or 2.  Given that, I think Assistant 2's code is better because ...  &lt;/think&gt;  The better response is...  \boxed{Assistant 2}</p>
<p>Alright, let's evaluate the problem step by step...  First, I look at Assistant 1's response...  Now, Assistant 2's response is much simpler...  Therefore, Assistant 1 correctly addressed the exponentiation but made a minor error in the final subtraction...  &lt;/think&gt;  The correct evaluation requires...\boxed{Assistant 1}</p>	<p>Okay, so I need to figure out which assistant is better for the given query. ...  Assistant 1 starts by ...  Assistant 2, on the other hand, ...  Looking at the instruction, the user wrote...  However, the problem is that the instruction might be ambiguous. If ..., then Assistant 2 is correct. But if ..., then Assistant 1 is correct. Since the instruction is...So I think the intended operation is exponentiation.  Therefore, Assistant 1's approach is more accurate for the likely intended problem. However, if the problem was meant to be multiplication, then Assistant 2 is correct. But given the notation, exponentiation is more probable. So Assistant 1's answer is better in that context.  But wait, the user's instruction is...  Hmm, this is a bit confusing. I need to decide based on the instruction. ...  But if ...  Given that, Assistant 2's interpretation is more likely correct...  So, considering the instruction's ambiguity, but leaning towards multiplication, Assistant 2 is better.  &lt;/think&gt;  The instruction is ambiguous, but Assistant 2 correctly interpreted it as multiplication, providing the accurate result.  \boxed{Assistant 2}</p>