Small Language Models as Judges for Rubric-Based Reinforcement Learning

Anonymous ACL submission

Abstract

Rubric-based reinforcement learning extends RL beyond tasks with exact answers or rule-based verifiers by scoring responses against instance-specific criteria. However, this makes reward computation expensive: training requires repeated rubric judging, often with proprietary APIs or local generative LLM judges with 7B parameters or more. We study whether smaller language models can serve as efficient and reliable rubric-based judges. To make this question measurable, we construct PointRubric and RaR-Science-Static, two pointwise rubric-based evaluation datasets with instance-specific criteria and itemwise satisfaction labels. We compare three ways of extracting criterion-level judgments from small models: Generative verdicts, Yes/No Logprob margins, and Probe judges. Across both datasets, the Qwen3-1.7B Probe judge achieves the strongest criterion-level agreement among these methods, outperforming Generative and Logprob judges. Used as a GRPO reward model, it trains a policy from 0.232 to 0.643 on RaR-Science rubric score, compared with 0.594 for an 8B Generative judge baseline, while the baseline requires 10.7× more reward-judge time. Task and domain transfer experiments further suggest that Probe judges preserve criterion-level reward structure across settings. [1]

1 Introduction

Recent progress in reinforcement learning with verifiable rewards (RLVR) has shown that outcome rewards from rule-based verifiers can effectively elicit reasoning abilities in large language models (LLMs) on verifiable tasks such as math and coding (Le et al.; Liu et al., 2023; Shao et al., 2024; Wen et al., 2025; Guo et al., 2025). However, many increasingly important agentic applications go beyond this narrow notion of verifiability. In long-form generation tasks such as Deep Research, quality can depend on content coverage, factuality, source use, and task-specific constraints, which are hard to capture with simple automatic checkers (Citron, 2024; Arora et al., 2025; OpenAI, 2025a; Shao et al., 2025). Rubric-based reinforcement learning offers a promising alternative for such settings: rubrics evaluate multiple aspects of generation quality through instance-specific criteria while still producing scalar rewards for optimization (Ouyang et al.; Gunjal et al., 2025).

A key component of rubric-based RL is obtaining the rubric reward, where an LLM judge evaluates responses against explicit rubric criteria and aggregates these judgments into rewards for RL training (Zheng et al.; Kim et al.). Existing rubric-based RL approaches mainly rely on proprietary model APIs or large generative judges to maintain judgment quality (Hurst et al., 2024; Gunjal et al., 2025; Shao et al., 2025). This makes reward computation expensive at scale because each RL step may require scoring long responses against multiple criteria. Proprietary judges also make exact reproduction harder because access depends on a fixed model snapshot. These limitations raise a central question: *Can much smaller language models serve as effective rubric judges for rubric-based RL?*

To answer this question, we need evaluation data that matches the reward computation used in rubric-based RL: explicit criteria, multiple responses under the same rubric, labels for each response–criterion pair, and a weighted scoring rule. Existing resources provide important pieces of this setup, but not the full structure. Preference and reward-model datasets focus on pairwise comparisons or scalar reward modeling (Ouyang et al.; Lambert et al.; Zhu et al.). Fine-grained and rubric-evaluation benchmarks provide skills, criteria, or judge-evaluation targets, but typically lack

[1] Our code and data is currently available at https://anonymous.4open.science/r/SLM-rubric-RL-A738/ and will be released publicly upon publication.

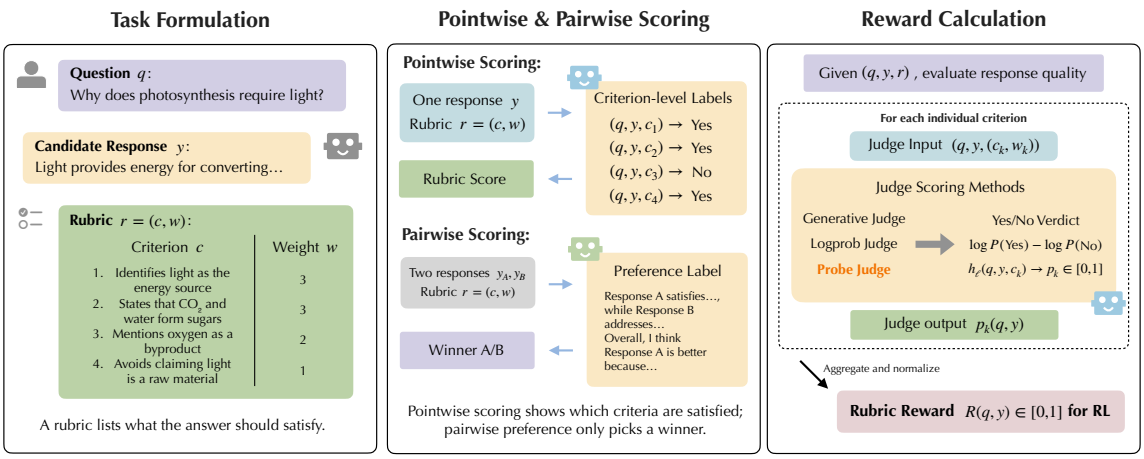

Figure 1: Rubric scoring setup. A rubric specifies criteria and weights for a candidate response. Pointwise scoring yields criterion-level labels, while pairwise scoring yields a relative winner. Criterion scores can then be aggregated into a scalar reward for evaluation or reinforcement learning.

weighted rubric rewards or response–criterion satisfaction labels for fixed candidate responses (Ye et al., 2023; Liu et al., 2025; Zhang et al., 2026; Pan et al., 2026; Wen et al., 2026). Domain-specific rubric datasets provide structured rubrics and scoring rules, but not controlled static response banks with labels for every response–criterion pair (Gunjal et al., 2025; Arora et al., 2025). We therefore derive PointRubric, a controlled pointwise benchmark from OpenRubrics, and construct RaR-Science-Static, a fixed response bank of similar format based on RaR-Science, for evaluating the same judge methods under the rubric-based RL format.

We compare three rubric-judging methods on PointRubric and RaR-Science-Static using Qwen3 models from 0.6B to 8B parameters (Yang et al., 2025): Generative verdicts, Yes/No Logprob margins, and hidden-state Probes. Across both static settings, Probe judges substantially outperform Generative and Logprob judges, suggesting that even 1.7B-scale models encode useful evaluative signals that are more reliably exposed through hidden states than through generated verdicts or verbalizer probabilities (Gisserot-Boukhlef et al., 2026; Li et al., 2026).

We then use the Qwen3-1.7B Probe judge as the reward model for GRPO in a controlled RaR-Science science setup (Shao et al., 2024; Gunjal et al., 2025). This Probe reward improves the policy's rubric score from 0.232 to 0.643, outperforming an 8B Generative judge reward baseline, which reaches 0.594, while the 8B baseline requires 10.7× more reward-judge time at the comparison checkpoint. Additional transfer experiments on GPQA-Diamond and cross-domain rubric splits show that the Probe signal is not limited to the static training setting (Rein et al., 2023).

Our contributions are as follows:

- We construct two pointwise rubric-evaluation datasets for studying small rubric judges: PointRubric, adapted from OpenRubrics, and RaR-Science-Static, built from RaR-Science to match the rubric format in downstream RL.
- We compare Generative, Logprob, and Probe judges on PointRubric and RaR-Science-Static. Probe judges perform best, showing that small models encode useful criterion-level evaluative signals that are more reliably exposed through hidden states than through generation.
- We use the Qwen3-1.7B Probe judge as the GRPO reward model on RaR-Science, training a policy from 0.232 to 0.643 on RaR-Science rubric score, compared with 0.594 for an 8B Generative judge reward baseline; the 8B Generative baseline requires 10.7× as much cumulative reward-judge time.

2 Rubric Judging Task and Dataset

A key component of rubric-based RL is rubric judging: determining whether a model response satisfies a given rubric criterion. We first define the task of pointwise rubric judging, where the judge evaluates one response against one rubric item at a time (§2.1). We then present the dataset design, review why existing datasets are not directly suitable for this setting, and describe how we adapt existing re-

Resource	Rubric criteria	Multiple responses	Per-criterion labels	Weighted rubric scores
FLASK (Ye et al., 2023)	✓	✓	×	×
HealthBench (Arora et al., 2025)	✓	×	×	✓
IF-RewardBench (Wen et al., 2026)	✓	✓	✓	×
JudgeLM-100K (Zhu et al.)	×	✓	×	×
OpenRubrics (Liu et al., 2025)	✓	✓	×	×
RaR-Science / RaR-Medicine (Gunjal et al., 2025)	✓	×	×	✓
RewardBench (Lambert et al.)	×	✓	×	×
RubricBench (Zhang et al., 2026)	✓	✓	×	×
RubricEval (Pan et al., 2026)	✓	×	✓	×
PointRubric (ours)	✓	✓	✓	✓
RaR-Science-Static (ours)	✓	✓	✓	✓

Table 1: Data requirements for pointwise rubric-based judge evaluation. PointRubric and RaR-Science-Static provide explicit rubric criteria, multiple responses per prompt, per-criterion labels, and weighted rubric scores.

sources into pointwise rubric-judging benchmarks (§2.2–§2.4).

2.1 Our Task: Pointwise Rubric Judging

Preference-based alignment methods (Ouyang et al.; Zheng et al.; Lambert et al.) typically learn reward signals from pairwise response comparisons: given a prompt and two candidate responses, a human annotator, reward model, or LLM judge predicts which response better satisfies the user's intent. This formulation is useful for learning relative preferences, but it collapses all evaluation criteria into a single comparison. As a result, it does not reveal which aspects of a response are correct or incorrect.

Rubric-based RL instead specifies an explicit rubric $C(q) = \{(c_j, w_j)\}_{j=1}^{m_q}$, where each c_j is a natural-language criterion and w_j is its aggregation weight. We therefore study *pointwise rubric judging*: given a prompt q, a candidate response y, and one rubric item c_j, the judge takes (q, y, c_j) as input and predicts whether response y satisfies criterion c_j. The judge may return either a binary satisfaction label or a soft satisfaction score. These item-level outputs are then aggregated with the rubric weights to produce the scalar rubric score used for evaluation or RL.

Pointwise rubric judging can still induce pairwise preferences by comparing the aggregated scores of two responses under the same prompt and rubric. However, unlike pairwise judging, it preserves the rubric structure: it identifies which criteria are satisfied or violated and supports direct scalar reward computation. Appendix A.1 gives additional formal details and examples.

2.2 Dataset Design Overview

Existing rubric and reward-model datasets provide useful supervision for judging model outputs. However, they are not directly designed for pointwise rubric judging, where the judge must determine whether a single response satisfies a single rubric criterion and support reward computation from these item-level decisions.

As shown in Table 1, our setting requires four components: explicit rubric criteria as judge targets, multiple responses under the same prompt and rubric, per-criterion satisfaction labels, and a weighted scoring rule for aggregating criterion-level judgments into scalar rewards. Existing resources cover parts of this design, but usually lack criterion-level labels, controlled response banks, or the weighted aggregation structure used in rubric-based RL. We therefore use two evaluation settings: PointRubric, a new pointwise benchmark derived from OpenRubrics, and RaR-Science-Static, a fixed response bank built from RaR-Science for static judge evaluation.

2.3 PointRubric Construction

PointRubric evaluates criterion-level rubric judging. It is derived from OpenRubrics, which provides general-domain prompts and detailed natural-language rubrics. Since the original supervision is pairwise and does not label criterion-level satisfaction, we add new candidate responses and per-criterion labels to convert OpenRubrics into a controlled pointwise benchmark.

Rubric standardization. We sample 3,000 OpenRubrics questions with explicit hard-rule constraints. We rewrite each one into six criteria: two hard criteria for mandatory constraints and four soft criteria for response quality. This fixed format gives all prompts the same criterion and weight structure while preserving the distinction between required constraints and softer quality dimensions.

Response roles. For each prompt, we generate four candidate responses under the same rubric: a full-score response, a low-score response, a rubric-guided partial response, and an unguided response. These roles are not treated as four ordinal quality bins. Instead, the full-score and low-score responses provide fixed anchors, while the partial and unguided responses introduce broader intermediate variation under the same prompt and rubric. In the final benchmark, full-score responses all receive score 10, low-score responses all fall in the 0–3 score range, and the partial and unguided responses cover broader score distributions. We then use GPT-4o (Hurst et al., 2024) as the operational reference judge to assign a binary satisfaction label to every response–criterion pair.

Final benchmark. After filtering and validation, PointRubric contains 1,042 questions, 4,168 responses, and 25,008 response–criterion labels. For evaluation, we use a question-level split of 417 training questions, 104 development questions, and 521 held-out questions, so all responses to the same prompt remain within a single split. Appendices A.2–A.5 give construction details; Appendix F gives a concrete example for PointRubric.

2.4 RaR-Science-Static

RaR-Science-Static complements PointRubric by testing the same pointwise judge methods in the downstream science-domain rubric format. It is built from RaR-Science, which uses instance-specific rubrics with variable numbers of criteria and absolute criterion weights (Gunjal et al., 2025). Unlike the RL experiments in §5, RaR-Science-Static is used only for static judge evaluation.

Response bank and split. RaR-Science-Static contains 1,500 randomly sampled science questions. Each question keeps the original RaR-Science reference answer and receives one additional candidate response generated by greedy decoding from Qwen3-4B (Yang et al., 2025), giving 3,000 candidate responses evaluated under the same rubric. The rubrics contain between 6 and 11 criteria, with an average of 7.52 criteria per question. For evaluation, we use a fixed split of 1,000 training questions, 200 development questions, and 300 test questions.

Reference labels and aggregation. We apply the same pointwise labeling protocol used for PointRubric: GPT-4o assigns a binary satisfaction label to each response–criterion pair. Scalar scores use the original RaR-Science criterion weights after converting pitfall criteria into the same satisfaction-label interface. Specifically, pitfall criteria with negative raw weights are treated as avoidance criteria, and scalar scores are aggregated with absolute weights so that higher satisfaction always corresponds to better rubric compliance. Appendices A.6 and A.7 give the response-bank construction and aggregation details; Appendix F also contains a concrete example of RaR-Science-Static.

2.5 Reference Labels and Human Audit

Both data settings use GPT-4o as the operational reference judge for per-criterion labels. We treat these labels as an operational reference target rather than as human ground truth. The purpose of the benchmark is to test whether smaller local judges can approximate an expensive rubric-grading pipeline, so the human audit is a reliability check on this target rather than a replacement for larger human evaluation.

We manually audit 100 sampled responses from the held-out PointRubric split, covering 600 decisions and 50 paired response comparisons. The audit achieves 90.9% weighted criterion agreement with GPT-4o and 96.0% pairwise preference agreement. Disagreements are concentrated in softer qualitative criteria, such as completeness, specificity, and clarity, rather than hard factual or constraint-following criteria. Appendix A.8 gives the audit protocol and additional examples.

3 Rubric-Judge Methods

We now describe how each small language model is used as a rubric judge. All methods receive the same item-level input (q, y, c_j): a prompt, a candidate response, and one rubric criterion. Each method predicts whether the response satisfies the criterion, either as a binary verdict or as a satisfaction probability. The resulting criterion-level outputs are aggregated with the same rubric weights. We compare three scoring methods: Generative, Logprob, and Probe. We also use SFT as a post-training test, then apply the same three methods to the SFT backbone. Table 2 summarizes the method matrix.

3.1 Scoring Methods

Generative Judges. Generative judges prompt the language model to produce a binary satisfaction verdict for one criterion. Parsed verdicts are

Method	LM tuned?	Head?	Output	RL use
Generative	×	×	Binary verdict	Baseline
Logprob	×	×	Satisfaction probability	–
Probe	×	✓	Satisfaction probability	Main
SFT Generative	✓	×	Binary verdict	–
SFT Logprob	✓	×	Satisfaction probability	–
SFT Probe	✓	✓	Satisfaction probability	–

Table 2: Rubric-judge method matrix. All methods use the same item-level input (q, y, c_j) and the same rubric-weight aggregation rule. SFT changes the backbone state, while Generated, Logprob, and Probe define the criterion-level readout.

aggregated with the rubric weights. Outputs that cannot be parsed into the required binary format are treated as invalid predictions.

Logprob Judges. Logprob judges replace verdict generation with forced-choice scoring. Given the same item-level input, the model is asked to score the next-token probabilities of the fixed verbalizers "Yes" and "No" under the criterion prompt and compute their log-probability margin:

$$\Delta_j = \log P(\text{Yes} \mid q, y, c_j) - \log P(\text{No} \mid q, y, c_j).$$

Binary static metrics threshold this margin at 0. Scalar rubric aggregation uses the soft score $p_j(q, y) = \sigma(\Delta_j)$. This method avoids parsing generated text. Appendix B.1 gives the prompt formats, verbalizers, and scoring details.

Probe Judges. Probe judges test whether hidden states encode the rubric judgment even when the model may not express it reliably through generated text (Li et al., 2026). We render the same single-criterion prompt, keep the language model frozen, and extract the final non-padding-token hidden state $h_\ell(q, y, c_j)$ at layer ℓ. A small classifier is trained with binary cross-entropy to predict the GPT-4o criterion label from this representation. Layer selection and the binary threshold use the development split. Scalar rubric scores and RL rewards use the unthresholded satisfaction probability. Appendix B.2 gives the training and calibration details.

3.2 Post-training with Supervised Fine-tuning

Supervised fine-tuning (SFT) is our post-training test: it asks whether supervised itemwise verdict training makes the same backbone a better rubric judge. We fine-tune the language model on criterion-verdict examples, following standard supervised instruction tuning (Ouyang et al.). Each example contains a prompt, a candidate response, one rubric criterion, and the GPT-4o criterion label. After SFT, we evaluate the resulting backbone with the same Generative, Logprob, and Probe methods as the base model. Therefore, SFT Generative, SFT Logprob, and SFT Probe differ from the corresponding base rows only in the backbone parameters, not in the scoring procedure. Appendix B.3 gives the SFT data format and evaluation setup.

4 Static Rubric-Judge Evaluation

Q1: Can 1.7B models serve as criterion-level rubric judges, and which readout works best?

Before using a rubric judge as an RL reward model, we first evaluate whether small LMs can approximate criterion-level judgments on fixed response banks. Specifically, each judge receives a question, candidate response, and rubric criterion, and predicts whether the response satisfies that criterion. We compare three readouts from the same Qwen3 backbones: generated verdicts, forced-choice Yes/No log probabilities, and hidden-state probes. All methods are evaluated against GPT-4o (Hurst et al., 2024) criterion labels as the operational reference.

4.1 Experiment Setup

We evaluate Qwen3 judge backbones at four scales: 0.6B, 1.7B, 4B, and 8B parameters (Yang et al., 2025). For each backbone, we compare Generative, Logprob, and Probe judges, using both base and SFT models when applicable. All methods are evaluated on the same held-out question splits, and trained or calibrated components use only the corresponding training and development data. Appendix B.4 defines the static criterion-satisfaction target and explains how binary verdicts, probabilities, and scalar rubric scores are computed.

We report one primary metric for each data setting. On PointRubric, the headline metric is weighted criterion accuracy, matching its hard/soft criterion weighting. On RaR-Science-Static, the headline metric is criterion-level macro-F1, since the downstream rubric setting has variable rubric lengths, weights, and label balance. Complementary metrics are reported in Appendix B.5.

4.2 Static Judge Results

Table 3 compares three ways of extracting criterion-level judgments from the same Qwen3 backbones.

	PointRubric						RaR-Science-Static					
	Base model			SFT model			Base model			SFT model		
Model	Generative	Logprob	Probe	Generative	Logprob	Probe	Generative	Logprob	Probe	Generative	Logprob	Probe
Qwen3-0.6B	0.370	0.582	0.802	0.560	0.604	**0.820**	0.413	0.433	0.793	0.607	0.605	**0.796**
Qwen3-1.7B	0.518	0.766	0.875	**0.902**	0.713	0.845	0.443	0.449	**0.835**	0.708	0.520	0.828
Qwen3-4B	0.790	0.893	**0.913**	0.808	0.892	0.906	0.496	0.738	**0.851**	0.673	0.794	0.845
Qwen3-8B	0.888	0.907	**0.914**	0.846	0.895	0.912	0.609	0.756	**0.864**	0.642	0.754	0.861

Table 3: Static evaluator results on held-out splits. PointRubric values are weighted criterion accuracy with four candidate responses per question; RaR-Science values are criterion-level macro-F1 against GPT-4o labels with two candidate responses per question. Bold marks the best method for each model size within each benchmark.

PointRubric shows that small models can perform pointwise rubric judging when the benchmark format is controlled. Base generative judge improves with scale, from 0.370 weighted criterion accuracy at 0.6B to 0.888 at 8B. SFT is especially effective for the 1.7B generative judge, which reaches 0.902, showing that a small model can learn the explicit Yes/No criterion-verdict format in this controlled setting.

On RaR-Science-Static, the readout method matters more than model size alone. For the same 1.7B base backbone, Generative and Logprob reach only 0.443 and 0.449 macro-F1, while Probe reaches 0.835. This contrast suggests that the small model contains useful rubric-satisfaction information, but that information is exposed much more reliably through hidden-state probes than through generative verdicts or verbalizer probabilities.

Probe is the strongest readout on RaR-Science-Static across all model sizes. The 1.7B Probe reaches 0.835 macro-F1, compared with 0.864 for the 8B Probe, while remaining substantially smaller than the 8B Generative judge baseline used later in RL. This makes Qwen3-1.7B Probe the natural candidate for the RL reward experiments: it is much smaller than the 8B Generative judge baseline while retaining strong criterion-level agreement on the downstream rubric format. Appendix B.6 gives confidence intervals for the results in Table 3.

4.3 Selecting the Probe Configuration for RL

Table 4 studies the Qwen3-1.7B Probe configuration used as the RL reward judge. The linear last-token probe is already strong with limited supervision, reaching 0.805 macro-F1 with 100 training questions and 0.834 with 1,000 training questions. The architecture variants provide only modest gains: MLP heads and last-4-layer averaging are close to the linear last-token probe, while mean pooling is substantially worse. We adopt the linear last-token probe in the main RL experiments. Although MLP-64 and last-4-layer averaging are slightly higher in this ablation, the linear last-token probe is simpler, nearly tied in macro-F1, and directly produces criterion-level probabilities for weighted rubric aggregation. It also avoids the formatting and parsing requirements of generated verdicts. The full layer sweep in Appendix B.7 shows that layer selection lies on a broad late-layer plateau rather than a single isolated optimum. This configuration gives a compact reward judge, so Section 5 evaluates whether it remains useful inside the RL loop as an online reward model.

Ablation	Probe setting	# Train q.	Macro-F1
Data	Linear, last token	50	0.767
	Linear, last token	100	0.805
	Linear, last token	250	0.818
	Linear, last token	500	0.828
	Linear, last token	1000	0.834
Design	Linear, last token	1000	0.834
	MLP-32, last token	1000	0.834
	MLP-64, last token	1000	**0.840**
	Linear, mean pooling	1000	0.761
	Linear, last-4-layer mean	1000	0.839

Table 4: Probe ablations for the Qwen3-1.7B reward judge on RaR-Science-Static. Data rows vary the number of training questions for the linear last-token probe. Design rows use 1,000 training questions and compare alternative heads and pooling strategies.

5 Probe-Based Rubric Rewards for RL

Section 4 selects Qwen3-1.7B Probe as a compact criterion-level judge. We now test whether this judge remains useful when used inside the RL loop, whether the learned signal transfers beyond the exact training setting, and how its quality-efficiency tradeoff compares with a larger generative judge.

Reward model	Method	RaR-Science score	Score gain
Base actor, no RL	–	0.232	–
Qwen3-0.6B	Probe	0.562	+0.330
Qwen3-1.7B	Probe	**0.643**	**+0.411**
Qwen3-4B	Probe	0.588	+0.356
Qwen3-8B	Probe	0.506	+0.274
Qwen3-8B	Generative	0.594	+0.362

Table 5: Matched Reward-model Comparison on RaR-Science. All non-base rows train the same Qwen3-4B-Base actor with the same GRPO configuration and are evaluated by the same GPT-4o rubric evaluator. The only experimental variable is the reward judge.

Policy	GPQA-Diamond acc.	Δ
Base actor	0.335 ± 0.023	–
Probe-reward policy	$\mathbf{0.388 \pm 0.037}$	+0.053

Table 6: Policy transfer to GPQA-Diamond. Accuracy is averaged over four evaluation runs on the 198-question GPQA-Diamond split, where in each run, the answer options are shown in a different order. The probe-reward policy is trained on RaR-Science using the Qwen3-1.7B Probe reward.

5.1 Matched RL Protocol and Results

Q2: Can the selected probe judge be used as an RL reward model?

We test whether the selected Qwen3-1.7B Probe can serve as the reward model for rubric-based RL. All controlled runs train the same Qwen3-4B-Base actor with GRPO (Shao et al., 2024) on the same RaR-Science training set. The actor, optimizer, schedule, response budget, aggregation rule, and final GPT-4o evaluator are fixed; only the reward judge changes. Full configuration details are given in Appendix C.1, and development checks are summarized in Appendix C.5.

The Qwen3-1.7B Probe reward produces the strongest trained policy. Table 5 reports the controlled reward-model comparison. Starting from the same base actor, the Qwen3-1.7B Probe reward improves the final RaR-Science rubric score from 0.232 to 0.643. Under the same training and evaluation protocol, the larger Qwen3-8B Generative reward reaches 0.594. Thus, the compact Probe reward trains a higher-scoring policy than the larger Generative judge in this matched RL setting. Appendix C.2 reports bootstrap confidence intervals for these final RL scores.

Downstream reward utility does not exactly follow static judge accuracy. The probe-size rows show that static judge accuracy is not sufficient for choosing the best RL reward model. Although the 4B and 8B Probes have slightly higher fixed-bank macro-F1 in Table 3, the 1.7B Probe gives the best externally evaluated policy score after GRPO. We therefore select the reward judge by downstream policy improvement under the controlled RL protocol, not by static macro-F1 alone.

A blind human audit supports the main policy comparison. The probe-reward policy is preferred over the base actor on 72 out of 100 examples, while GPT-4o rubric preference favors the probe policy on 80 out of the same 100 pairs. These checks support the operational GPT-4o target rather than replacing a larger human evaluation. Appendix C.4 gives the audit protocol, tie analysis, and reference-judge validation.

Train	Eval	Macro-F1	Score MAE	Pearson
Science	Science	0.754	0.187	0.720
Science	Medicine	0.718	**0.091**	**0.789**
Medicine	Medicine	**0.782**	0.140	0.620
Medicine	Science	0.702	0.240	0.580

Table 7: Static judge transfer across rubric domains. Probes are trained on source-domain criterion labels and evaluated against GPT-4o criterion labels on the target domain without target-domain fitting.

5.2 Transfer Results

Q3: Does the learned signal transfer beyond the RaR-Science training setting?

We evaluate transfer in two ways: whether the policy trained with the probe reward improves outside the RaR-Science rubric format, and whether the Qwen3-1.7B Probe's criterion-satisfaction signal transfers from science rubrics to medical rubrics without target-domain fitting. The evaluation protocols are defined in Appendix D.

Table 6 compares the base Qwen3-4B actor with the policy trained using the Qwen3-1.7B Probe reward. On 198 GPQA-Diamond questions (Rein et al., 2023) across four answer-order runs, the probe-reward policy improves accuracy from 0.335 to 0.388, suggesting that **the policy improvement can generalize beyond the RaR-Science rubric-scoring format.**

Table 7 evaluates whether the probe's criterion-satisfaction signal transfers across domains. A Qwen3-1.7B linear probe trained on RaR-Science criterion labels reaches 0.718 macro-F1 on RaR-Medicine without Medicine labels for fitting. This is below the Medicine-trained in-domain probe,

Reward judge	Cumulative judge time	Validation judge time	RL elapsed time	RaR-Science score
Qwen3-1.7B Probe	2.33h	31.1s	8h 51m 47s	**0.643**
Qwen3-8B Generative	24.98h	492.0s	11h 11m 56s	0.594
Generative / Probe ratio	10.7×	15.8×	1.26×	–

Table 8: Efficiency-quality comparison for the Qwen3-1.7B Probe judge and the generative baseline. Judge time is measured through the same checkpoint used in Table 5. Policy quality for both runs is measured by GPT-4o. The generated judge requires 10.7× as much cumulative judge time while producing a lower-scoring policy.

but shows that the **RaR-Science-trained probe retains meaningful criterion-level agreement under a different rubric distribution.**

5.3 Efficiency-Quality Results

💡 **Q4:** What is the efficiency-quality tradeoff compared with a generated judge?

Table 8 compares the two reward judges in the matched RL comparison: Qwen3-1.7B Probe and Qwen3-8B Generative. The actor, training data, GRPO configuration, response budget, rubric aggregation rule, and external GPT-4o evaluator are fixed; only the reward judge changes. By the comparison checkpoint, Generative requires 89,912.1 seconds of cumulative judge time versus 8,389.9 seconds for Probe, a 10.7× ratio. The validation reward pass shows the same pattern: 492.0 seconds versus 31.1 seconds, a 15.8× ratio.

The efficiency result should be read with policy quality. Probe reaches a higher RaR-Science score, 0.643 versus 0.594, while using 2.33 judge-hours instead of 24.98 judge-hours. The wall-clock reduction is smaller because reward calls are parallelized and actor rollout, reference-model scoring, optimization, and checkpointing remain in the loop. The key result is not only lower judge cost: **the Probe reward produces the stronger policy while requiring substantially less reward-judge computation.** Appendix C.3 defines the timing fields and reports the training-only judge-time calculation.

6 Related Work

Rubric-based RL. Reinforcement learning with verifiable rewards works well when correctness can be checked by rule-based verifiers (Le et al.; Shao et al., 2024; Wen et al., 2025; Guo et al., 2025). Open-ended tasks require broader feedback, motivating rubric-style evaluation, rubric resources, and rubric-based RL (Arora et al., 2025; Liu et al., 2025; Gunjal et al., 2025; Shao et al., 2025). RaR-Science is closest to our downstream setting because it uses instance-specific science rubrics as scalar rewards for policy optimization (Gunjal et al., 2025). Our work keeps the reward formulation fixed and studies the judge that applies the rubric, replacing repeated calls to a large judge with a smaller criterion-level evaluator.

LLM-as-a-judge and small evaluators. LLM-as-a-judge methods use language models to score, compare, or critique open-ended responses (Zheng et al.; Liu et al.; Gu et al., 2024). Trained open judges extend this idea by fine-tuning models for evaluation (Wang et al.; Zhu et al.; Li et al.; Kim et al.). Recent evaluator benchmarks study reward-model and judge reliability across evaluation formats (Lambert et al.; Wen et al., 2026). Work on compact evaluators and hidden-state judges further suggests that smaller models can contain useful evaluative signals even when generation is weak (Alexandru et al., 2025; Li et al., 2026; Gisserot-Boukhlef et al., 2026). Our work applies this direction to rubric-based RL by comparing Generative, Logprob, and Probe judges for pointwise criterion satisfaction, then using the Qwen3-1.7B Probe judge as a GRPO reward model.

7 Conclusion

We study whether small LMs can serve as criterion-level rubric judges for rubric-based RL. We derive PointRubric from OpenRubrics and construct RaR-Science-Static from RaR-Science to test the evaluation capabilities of small judges. Across both datasets, Probe judges perform best, showing that 1.7B-scale models encode useful rubric-satisfaction signals in hidden states. When used as an RL reward model, the Qwen3-1.7B Probe judge trains a policy that outperforms an 8B Generative judge reward baseline while using far less judge time. Transfer experiments suggest these gains extend beyond our downstream training settings. Overall, small Probe judges provide an effective and efficient reward model for rubric-based RL.

Limitations

Our supervised target is GPT-4o criterion scoring, not direct human ground truth. This matches the operational goal of the paper: approximating the expensive rubric judge used in the studied rubric-RL pipeline. Human preference comparisons and independent rescoring in Appendix C support the validity of this target in our setting, but broader human evaluation would be needed to characterize agreement across tasks and domains.

Our strongest evidence is on RaR-Science questions, with additional checks on GPQA-Diamond and RaR-Medicine. These results show useful transfer beyond the calibration setting, but do not imply domain-invariant rubric judgment. New domains, especially dialogue, long-form instruction following, or safety-critical expert settings, may require additional calibration and validation. Similarly, the RL experiments control the actor, reward aggregation, and training configuration, but cover one policy family and one rubric-RL setup; we therefore treat the human audit and independent-reference checks as setting-specific evidence against reward overoptimization.

Finally, the efficiency results are measurements of our implementation and hardware. We report cumulative judge time, validation reward time, and elapsed RL time from matched runs. The judge-time advantage is larger than the wall-clock advantage because reward calls are parallelized and other RL costs remain in the loop, so the reported ratios should be interpreted as empirical measurements for this setup rather than universal serving constants.

Ethical considerations

This work reuses two public research datasets, OpenRubrics and RaR-Science (Liu et al., 2025; Gunjal et al., 2025). We use these resources for research on rubric judging and cite the original papers. PointRubric is derived from OpenRubrics, and RaR-Science-Static is derived from RaR-Science by adding a fixed response bank and reference labels for static judge evaluation. We do not claim ownership of the original prompts, rubrics, or reference answers. Any released version of our derived data should retain the original source attribution and follow the usage terms provided by the original dataset authors.

The datasets contain model prompts, rubrics, reference answers, and model responses. We do not collect or add private user data, demographic attributes, or personal identifiers. During construction and validation, we manually inspected sampled examples and generated responses for obvious personally identifying information and unsafe or offensive content. We did not intentionally retain any newly introduced personal identifiers. Because PointRubric and RaR-Science-Static are derived from public research datasets and model-generated responses, residual unsafe, biased, sensitive, or offensive content may still be inherited from the source artifacts or produced by models. We therefore treat the derived artifacts as research-only evaluator resources, and our benchmark should not be treated as a guarantee that a judge is safe for deployment.

We conduct two small human audits: one audit of PointRubric reference labels and one blind pairwise audit of the main RL comparison. Annotators are asked only to evaluate model responses against provided rubrics or to compare two model responses. The annotators were members of the research team, so no external recruitment or payment was involved. The tasks do not ask annotators to disclose personal information, and we store and report only aggregate agreement or preference statistics. The audits are used as reliability checks for the operational GPT-4o reference target, not as standalone human preference datasets or as a replacement for broader human evaluation.

References

Andrei Alexandru, Antonia Calvi, Henry Broomfield, Jackson Golden, Kyle Dai, Mathias Leys, Maurice Burger, Max Bartolo, Roman Engeler, and et al. Pisupati, Sashank. 2025. Atla selene mini: A general purpose evaluation model. *arXiv preprint arXiv:2501.17195*.

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, and et al. Beutel, Alex. 2025. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*.

Dave Citron. 2024. Try deep research and our new experimental model in gemini, your ai assistant. https://blog.google/products-and-platforms/products/gemini/google-gemini-deep-research/. Accessed: 2026-05-26.

Hippolyte Gisserot-Boukhlef, Nicolas Boizard, Emmanuel Malherbe, Céline Hudelot, and Pierre

Colombo. 2026. Bert-as-a-judge: A robust alternative to lexical methods for efficient reference-based llm evaluation. *arXiv preprint arXiv:2604.09497*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, and et al. Liu, Honghao. 2024. A survey on llm-as-a-judge. *The Innovation*.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, and et al. Bi, Xiao. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, and et al. Radford, Alec. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

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 *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, and et al. Choi, Yejin. Rewardbench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, and et al. Liu, Pengfei. Generative judge for evaluating alignment. In *International Conference on Learning Representations*.

Zhuochun Li, Yong Zhang, Ming Li, Yuelyu Ji, Yiming Zeng, Ning Cheng, Yun Zhu, Yanmeng Wang, Shaojun Wang, and et al. Xiao, Jing. 2026. Rethinking llm-as-a-judge: Representation-as-a-judge with small language models via semantic capacity asymmetry. *arXiv preprint arXiv:2601.22588*.

Jiate Liu, Yiqin Zhu, Kaiwen Xiao, Qiang Fu, Xiao Han, Wei Yang, and Deheng Ye. 2023. Rltf: Reinforcement learning from unit test feedback. *arXiv preprint arXiv:2307.04349*.

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. 2025. Openrubrics: Towards scalable synthetic rubric generation for reward modeling and llm alignment. *arXiv preprint arXiv:2510.07743*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

OpenAI. 2025a. Deep research system card. https://openai.com/index/deep-research-system-card/. Accessed: 2026-05-26.

OpenAI. 2025b. Introducing gpt-4.1 in the api. https://openai.com/index/gpt-4-1/. Accessed: 2026-05-25.

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

Tianjun Pan, Xuan Lin, Wenyan Yang, Qianyu He, Shisong Chen, Licai Qi, Wanqing Xu, Hongwei Feng, Bo Xu, and Yanghua Xiao. 2026. Rubriceval: A rubric-level meta-evaluation benchmark for llm judges in instruction following. *arXiv preprint arXiv:2603.25133*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, and et al. Sontag, David. 2025. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*.

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

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

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, and et al. Ananthram, Akhila. 2025. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Yidong Wang, Zhuohao Yu, Wenjin Yao, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, and et al. Wang, Jindong. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. In *International Conference on Learning Representations*.

Bosi Wen, Yilin Niu, Cunxiang Wang, Xiaoying Ling, Ying Zhang, Pei Ke, Hongning Wang, and Minlie Huang. 2026. If-rewardbench: Benchmarking judge models for instruction-following evaluation. *arXiv preprint arXiv:2603.04738*.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, and et al. Miao, Ziming. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*.

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

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2023. Flask: Fine-grained language model evaluation based on alignment skill sets. *arXiv preprint arXiv:2307.10928*.

Qiyuan Zhang, Junyi Zhou, Yufei Wang, Fuyuan Lyu, Yidong Ming, Can Xu, Qingfeng Sun, Kai Zheng, Peng Kang, and et al. Liu, Xue. 2026. Rubricbench: Aligning model-generated rubrics with human standards. *arXiv preprint arXiv:2603.01562*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, and et al. Xing, Eric. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*.

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

A PointRubric Construction and Rubric Scoring

A.1 Pointwise and Pairwise Rubric Scoring

Let q denote a prompt, y a candidate response, and $C(q) = \{c_j\}_{j=1}^{m_q}$ the rubric for q. Each criterion c_j is associated with a nonnegative weight w_j. In settings with hard and soft criteria, w_j is obtained by mapping the criterion type to a fixed weight; in settings with absolute rubric weights, w_j is given directly by the rubric. A pointwise rubric label is a criterion-satisfaction judgment

$$z_j(q, y) \in \{0, 1\},$$

where $z_j(q, y) = 1$ means that response y satisfies criterion c_j for prompt q. A probabilistic judge instead returns $p_j(q, y) \in [0, 1]$, interpreted as the probability that c_j is satisfied.

The normalized rubric score for a response is

$$R(q, y) = \frac{\sum_{j=1}^{m_q} w_j z_j(q, y)}{\sum_{j=1}^{m_q} w_j},$$

or, for probabilistic judges,

$$\widehat{R}(q, y) = \frac{\sum_{j=1}^{m_q} w_j p_j(q, y)}{\sum_{j=1}^{m_q} w_j}.$$

This aggregation preserves the rubric structure while producing the scalar reward required for reinforcement learning.

A pairwise preference between two responses y_a and y_b to the same prompt can be induced from pointwise rubric scores:

$$y_a \succ_q y_b \quad \Longleftrightarrow \quad R(q, y_a) > R(q, y_b),$$

with ties defined when the two normalized scores are equal. Pairwise labels are therefore recoverable from pointwise rubric labels when the same rubric is applied to both responses. The reverse is not true: a pairwise preference does not identify which criteria were satisfied or violated. Pointwise labels are therefore richer for rubric-based RL because they preserve the criterion-level reward structure used to compute rewards, analyze criterion-level errors, and train reward readouts without collapsing the rubric to a single preference label.

For example, suppose a rubric contains one criterion requiring the final answer and another requiring supporting reasoning. A response may satisfy the final answer criterion while failing the reasoning criterion. Pointwise labels retain this distinction, whereas a pairwise label only states whether that response is preferred to another response under the aggregate score.

Stage	Count	Criterion
Candidates	3,000	Two sampled construction passes
Unique questions	2,902	Before filtering
Full-score answer	2,432	Answer 1 score $= 10$
Low-score answer	1,566	Also answer 2 score ≤ 3
Intermediate answer	1,061	Also some answer score in $[2, 8]$
Unique after filters	1,045	Deduplicated by question ID
Frozen PointRubric	1,042	Canonical benchmark used in all experiments

Table A.1: PointRubric construction stages. Counts are computed over completed construction artifacts. The final row gives the frozen benchmark artifact used for all experiments.

A.2 PointRubric Construction Details

Table A.1 summarizes the construction procedure. We sample question records from OpenRubrics (Liu et al., 2025) whose original rubrics contain explicit hard requirements, then refine each rubric into a fixed six-criterion format with two hard criteria and four soft criteria. We use GPT-4o for rubric refinement and reference grading. The final benchmark is obtained by retaining questions whose generated response set spans full-score, low-score, and intermediate rubric outcomes.

A.3 PointRubric Response Roles

Table A.2 describes the four responses associated with each question. The responses should not be interpreted as four ordinal quality levels. Instead, PointRubric fixes one full-score response and one low-score response, then includes two additional comparison responses whose observed scores vary.

A.4 PointRubric Evaluation Protocol and Metrics

For judge methods that require training or calibration, we use a fixed question-level split so that responses to the same question never appear in both calibration and held-out evaluation. Frozen or zero-shot judges can be evaluated on the full benchmark.

The primary PointRubric metric is weighted criterion accuracy, using weight 3 for hard criteria and weight 1 for soft criteria. Complementary static metrics include parse success, unweighted criterion accuracy, macro-F1, balanced accuracy, MCC,

ID	Role	Rubric visible	Generator in final artifact	Retention status
1	Full-score response	Yes	GPT-4o (1,042)	Required to receive total score 10.
2	Low-score response	Yes	GPT-4o (1,042)	Required to receive total score at most 3.
3	Partial-compliance response	Yes	GPT-4o (764), GPT-4.1-nano (OpenAI, 2025b) (278)	Additional comparison response; broad observed score range.
4	Unguided response	No	GPT-4.1-nano (1,042)	Additional comparison response; natural score variation without rubric access.

Table A.2: Response roles in PointRubric. Counts reflect the final benchmark artifact.

ID	Mean	0–3	4–6	7–9	10
1	10.00	0	0	0	1,042
2	0.89	1,042	0	0	0
3	6.39	89	461	367	125
4	8.47	44	117	395	486

Table A.3: Observed GPT-4o rubric-score distribution by response ID in the final PointRubric benchmark. Scores range from 0 to 10 under hard-rule weight 3 and soft-rule weight 1.

Split	Questions	Responses	Criterion labels
Train	417	1,668	10,008
Development	104	416	2,496
Held-out	521	2,084	12,504
Total	1,042	4,168	25,008

Table A.4: Question-level PointRubric split used for calibrated or trained judge methods, after validating all criterion labels.

scalar-score error, score correlation, and induced pairwise ordering accuracy.

A.5 Reference Judge and Construction Prompts

We use GPT-4o (Hurst et al., 2024) as the construction and reference judge for PointRubric. During benchmark construction, GPT-5 (Singh et al., 2025) frequently refused or failed to produce responses with the designated quality level, especially deliberately low-scoring or partially compliant answers. Since PointRubric requires controlled variation across responses to the same prompt, we used GPT-4o for rubric refinement, designated-quality response generation, and criterion-level grading. For consistency, all subsequent PointRubric judge comparisons use the same GPT-4o reference labels. The rubric-refinement, answer-generation, and grading prompts are provided for reproducibility in Appendix E.

A.6 RaR-Science-Static Response Bank and Splits

RaR-Science-Static is the fixed response bank used for static judge evaluation under the RaR-Science rubric format. It contains 1,500 science questions sampled from RaR-Science (Gunjal et al., 2025). Each question is paired with two candidate responses: the original dataset reference answer and a greedy Qwen3-4B response (Yang et al., 2025). The resulting bank contains 3,000 candidate responses. The rubrics are instance-specific; in RaR-Science-Static, they contain between 6 and 11 criteria, with an average of 7.52 criteria per question.

For trained or calibrated static methods, we use a fixed question-level split: 1,000 training questions, 200 development questions, and 300 held-out questions. The held-out split contains 600 candidate responses. Splitting is performed by question, so no responses to the same question appear in both calibration and evaluation. Generative and Logprob judges do not require fitting, but they are evaluated on the same held-out questions for comparability.

A.7 RaR-Science-Static Reference Labels and Aggregation

We use GPT-4o (Hurst et al., 2024) as the operational reference judge for RaR-Science-Static. The reference judge assigns one Yes/No verdict per rubric criterion for each candidate response. The final reference-scoring run parses successfully for all 3,000 candidate responses.

For scalar-score metrics, RaR-Science-Static uses the original RaR-Science rubric weights after converting all criteria into a satisfaction-label interface. Some RaR-Science criteria are pitfall criteria with negative raw weights. We interpret these as avoidance criteria: satisfying the criterion means that the response avoids the pitfall. We therefore aggregate with absolute weights so that higher criterion satisfaction always corresponds to better

rubric compliance:

$$\widehat{R}(q, y) = \frac{\sum_{j=1}^{m_q} |w_j| \, p_j(q, y)}{\sum_{j=1}^{m_q} |w_j|}.$$

For binary methods, $p_j(q, y)$ is replaced by $\hat{z}_j(q, y)$. The same aggregation rule is used for Generative, Logprob, and Probe judges on RaR-Science-Static.

A.8 Reference-Label Validation

We validate the GPT-4o reference labels with a blinded human audit on the held-out PointRubric split used in our static evaluator experiments. The audit samples 50 question-level groups, and for each group selects two candidate responses from the four available responses such that the two candidates have different GPT-4o rubric scores. The resulting 100 responses are shuffled before annotation. For each response, the annotator labels every rubric criterion as satisfied or not satisfied without seeing the GPT-4o labels or scores.

The audit covers 600 response–criterion decisions. We compute criterion-level agreement between the human annotator and GPT-4o labels, and additionally report a weighted agreement that uses the rubric criterion weights when aggregating decisions. Human labels agree with GPT-4o on 90.9% of weighted criterion decisions. Agreement is highest for hard factual or constraint-following criteria, and lower for softer criteria involving completeness, clarity, specificity, and overall explanation quality. Table A.5 summarizes the audit size and agreement results.

We also evaluate whether the human audit preserves the response ordering induced by the rubric score. For each of the 50 sampled response pairs, we aggregate the human criterion labels with the same rubric weights and compare the preferred response with the GPT-4o-preferred response. Human and GPT-4o preferences agree on 96.0% of pairs. The remaining disagreement comes from a case where an intentionally weak response was rated too generously by the human annotator, illustrating that the audit is a reliability check rather than a replacement for expert adjudication.

Audit statistic	Value
Sampled responses	100
Sampled response pairs	50
Response–criterion decisions	600
Weighted criterion agreement with GPT-4o	90.9%
Pairwise preference agreement with GPT-4o	96.0%

Table A.5: Human audit of GPT-4o reference labels on the held-out PointRubric split.

B Static Evaluation Details

B.1 Generative and Logprob Judge Details

Generative judging. The Generative judge receives one criterion at a time and produces a single Yes/No verdict. Verdicts are then recombined into a full criterion vector for the candidate response. This reduces the output-format burden for each completion but requires one generation per criterion.

Forced-choice Logprob scoring. The Logprob judge receives the question, candidate response, and one canonicalized criterion. The prompt asks whether the response satisfies the criterion and ends before the answer token. We score the next-token probabilities of the fixed verbalizers " Yes" and " No" and compute their log-probability margin:

$$\Delta_j = \log P(\text{ Yes} \mid q, y, c_j) - \log P(\text{ No} \mid q, y, c_j).$$

We set the soft satisfaction score to $p_j(q, y) = \sigma(\Delta_j)$. Binary metrics threshold the margin at 0, equivalently $p_j(q, y) \geq 0.5$. Scalar rubric aggregation uses the soft satisfaction score.

B.2 Probe Training and Calibration

Canonical criterion rendering. Probe prompts use a single criterion at a time. Ordinary criteria are rendered as satisfaction criteria. Pitfall criteria are rendered as avoidance criteria, so a positive label always means that the response complies with the rubric.

Representation extraction. For each rendered prompt, we run the frozen judge model with hidden states enabled and extract the representation at the final non-padding token. For layer ℓ, this gives $h_\ell(q, y, c_j)$. We evaluate last-token, mean-pooled, and last-few-token pooled representations in the ablation table.

Feature standardization. For each layer, training features are used to compute a mean vector and standard deviation vector. Training, development, and held-out features are standardized with these training-set statistics. Held-out features are never used to choose normalization statistics, layers, thresholds, or classifier checkpoints.

Probe heads. We train linear and MLP binary classifiers with binary cross-entropy. The classifier predicts the GPT-4o criterion label from the frozen representation. For MLP probes, only the small probe head is trained; the language model remains frozen.

Layer and threshold selection. For each candidate layer, we train the probe on the training split and evaluate on the development split. The threshold is selected on development data to maximize macro-F1, and the layer is selected by the same development metric. The held-out split is evaluated only after these choices are fixed.

Probability rewards. At evaluation time, the probe outputs a criterion probability. Scalar scores use these probabilities rather than only thresholded verdicts unless otherwise specified. In the RL experiments, these probabilities provide the reward signal while preserving the same explicit rubric aggregation used by generative judges.

B.3 Supervised Fine-tuning Details

SFT uses the same train, development, and held-out question splits as the corresponding static evaluation setting. The model is fine-tuned only on the training split, with development loss used for checkpoint selection. Held-out questions are used only for final reporting. After SFT, the resulting backbone is evaluated with the same Generative, Logprob, and Probe methods used for the base backbone.

For PointRubric, SFT trains the model to output the full rubric verdict vector for a candidate response. Each example contains the question, candidate response, all rubric criteria, and the corresponding GPT-4o Yes/No verdicts. For RaR-Science-Static, SFT uses examples formed from the fixed response bank and the GPT-4o reference verdicts. For SFT Probe, the probe is trained on hidden states extracted from the SFT backbone using the same probe-training protocol as the base Probe.

Metric	Value
Backbones	Qwen3-0.6B, Qwen3-1.7B, Qwen3-4B, Qwen3-8B (Yang et al., 2025)
Training objective	Supervised verdict prediction
Fine-tuning method	LoRA (Hu et al., 2021)
LoRA target modules	All target modules
Chat template	Qwen
Precision	bf16
Optimizer	AdamW (Loshchilov and Hutter, 2017)
Scheduler	Cosine decay
Warmup ratio	0.03
Epochs	3
Checkpoint selection	Lowest development loss
Checkpoint limit	3
PointRubric training examples	Question-level training split
PointRubric maximum length	4096 tokens
PointRubric learning rate	2×10^{-4}
PointRubric eval/save frequency	200 steps
PointRubric batch/accumulation	0.6B: batch 2, accumulation 16; larger models: batch 1, accumulation 32
RaR-Science-Static training examples	Question-level training split
RaR-Science-Static maximum length	8192 tokens
RaR-Science-Static learning rate	1×10^{-4}
RaR-Science-Static eval/save frequency	50 steps
RaR-Science-Static batch/accumulation	0.6B: batch 2, accumulation 16; larger models: batch 1, accumulation 16

Table B.1: SFT configuration for the post-training judge experiments.

B.4 Static Evaluation Targets

For both PointRubric and RaR-Science, the static target is criterion satisfaction. Each example is a tuple (q, y, c_j), where q is the question, y is a candidate response, and c_j is one rubric criterion. The operational reference label is

$$z_j^*(q, y) \in \{0, 1\},$$

where $z_j^*(q, y) = 1$ means that GPT-4o (Hurst et al., 2024) judges response y to satisfy criterion c_j for question q.

A judge method may return either a binary verdict $\hat{z}_j(q, y) \in \{0, 1\}$ or a probability $p_j(q, y) \in [0, 1]$. Binary metrics are computed from $\hat{z}_j(q, y)$, using a thresholded probability when the method is probabilistic. Scalar rubric scores are computed by aggregating criterion predictions with the rubric weights, so the scalar score is derived from criterion-level judgments rather than predicted directly.

B.5 Static Metrics

Parse success. For generative judges, sample-level parse success is the fraction of candidate responses for which the judge output can be parsed into a complete criterion verdict vector. Criterion-level parse success is the fraction of individual

Layer	Dev Macro-F1	Held-out Acc.	Held-out Macro-F1	Held-out Bal. Acc.	Held-out MCC
1	0.726	0.759	0.738	0.733	0.480
2	0.742	0.774	0.750	0.743	0.508
3	0.751	0.781	0.754	0.745	0.523
4	0.752	0.787	0.769	0.764	0.540
5	0.755	0.778	0.751	0.742	0.517
6	0.752	0.791	0.771	0.763	0.548
7	0.758	0.791	0.779	0.780	0.558
8	0.768	0.794	0.774	0.766	0.554
9	0.765	0.787	0.774	0.775	0.548
10	0.778	0.804	0.790	0.788	0.581
11	0.784	0.810	0.800	0.802	0.600
12	0.790	0.819	0.805	0.800	0.611
13	0.805	0.827	0.812	0.805	0.627
14	0.799	0.830	0.818	0.814	0.636
15	0.814	0.831	0.814	0.806	0.635
16	**0.836**	**0.845**	**0.833**	**0.829**	**0.667**
17	0.836	0.849	0.839	0.837	0.678
18	0.832	0.853	0.841	0.835	0.684
19	0.827	0.847	0.835	0.829	0.672
20	0.824	0.843	0.835	0.838	0.670
21	0.828	0.845	0.835	0.836	0.671
22	0.827	0.843	0.834	0.833	0.667
23	0.823	0.848	0.837	0.835	0.675
24	0.821	0.847	0.837	0.837	0.675
25	0.819	0.843	0.834	0.835	0.668
26	0.818	0.844	0.835	0.835	0.670
27	0.816	0.839	0.828	0.825	0.656
28	0.817	0.843	0.833	0.832	0.665

Table B.2: Qwen3-1.7B Probe layer sweep on RaR-Science-Static. Layer selection is based on development macro-F1; the selected layer is bolded.

criterion verdicts successfully recovered. Logprob and probe methods are defined without generative verdict parsing and therefore produce scores for every rendered criterion.

Criterion accuracy. Criterion accuracy is the Hamming accuracy over criterion labels:

$$\mathrm{Acc} = \frac{1}{N} \sum_{(q,y,j)} \mathbb{1}\{\hat{z}_j(q,y) = z_j^*(q,y)\}.$$

Weighted criterion accuracy. Weighted criterion accuracy weights each criterion match by the rubric weight:

$$\mathrm{WAcc} = \frac{\sum_{(q,y,j)} \omega_j \mathbb{1}\{\hat{z}_j(q,y) = z_j^*(q,y)\}}{\sum_{(q,y,j)} \omega_j}.$$

For PointRubric, ω_j follows the hard/soft rubric weights. For RaR-Science, $\omega_j = |w_j|$.

Macro-F1. Macro-F1 is computed over the two criterion labels, Yes and No. Let F_1^{Yes} be the F1 score treating satisfied criteria as the positive class, and let F_1^{No} be the F1 score treating unsatisfied criteria as the positive class. Then

$$\mathrm{MacroF1} = \frac{1}{2}\left(F_1^{\mathrm{Yes}} + F_1^{\mathrm{No}}\right).$$

We use macro-F1 as the headline RaR-Science static metric because the distribution of satisfied and unsatisfied criteria varies across questions and candidate types.

Balanced accuracy and MCC. Balanced accuracy averages the recall of the Yes and No classes, making it less sensitive to class imbalance than ordinary accuracy. Matthews correlation coefficient (MCC) is computed from the binary confusion matrix and summarizes criterion-level agreement on a scale from -1 to 1.

	Base model			SFT model		
Model	Generative	Logprob	Probe	Generative	Logprob	Probe
Qwen3-0.6B	0.370 [0.349, 0.390]	0.582 [0.568, 0.596]	0.802 [0.791, 0.811]	0.560 [0.537, 0.584]	0.604 [0.591, 0.618]	0.820 [0.810, 0.830]
Qwen3-1.7B	0.518 [0.493, 0.541]	0.766 [0.757, 0.776]	0.875 [0.867, 0.883]	0.902 [0.894, 0.909]	0.713 [0.705, 0.722]	0.845 [0.836, 0.854]
Qwen3-4B	0.790 [0.768, 0.811]	0.893 [0.886, 0.899]	0.913 [0.906, 0.919]	0.808 [0.788, 0.827]	0.892 [0.885, 0.899]	0.906 [0.899, 0.912]
Qwen3-8B	0.888 [0.876, 0.899]	0.907 [0.900, 0.914]	0.914 [0.907, 0.921]	0.846 [0.828, 0.862]	0.895 [0.889, 0.902]	0.912 [0.906, 0.918]

Table B.3: PointRubric static results with 95% bootstrap confidence intervals. Values are weighted criterion accuracy.

	Base model			SFT model		
Model	Generative	Logprob	Probe	Generative	Logprob	Probe
Qwen3-0.6B	0.413 [0.398, 0.427]	0.433 [0.414, 0.451]	0.793 [0.775, 0.811]	0.607 [0.512, 0.695]	0.605 [0.583, 0.626]	0.796 [0.778, 0.813]
Qwen3-1.7B	0.443 [0.426, 0.459]	0.449 [0.431, 0.467]	0.835 [0.819, 0.851]	0.708 [0.661, 0.753]	0.520 [0.497, 0.544]	0.828 [0.813, 0.843]
Qwen3-4B	0.496 [0.475, 0.518]	0.738 [0.716, 0.760]	0.851 [0.836, 0.865]	0.673 [0.637, 0.707]	0.794 [0.776, 0.812]	0.845 [0.829, 0.859]
Qwen3-8B	0.609 [0.587, 0.632]	0.756 [0.736, 0.775]	0.864 [0.850, 0.878]	0.642 [0.598, 0.684]	0.754 [0.735, 0.774]	0.861 [0.847, 0.874]

Table B.4: RaR-Science-Static results with 95% bootstrap confidence intervals. Values are criterion-level macro-F1.

Group	Setting	Macro-F1 [95% CI]
Data size	50	0.767 [0.747, 0.785]
Data size	100	0.805 [0.787, 0.820]
Data size	250	0.818 [0.801, 0.834]
Data size	500	0.828 [0.811, 0.843]
Data size	1000	0.834 [0.818, 0.849]
Probe design	Linear, last token	0.834 [0.818, 0.849]
Probe design	MLP-32, last token	0.834 [0.818, 0.850]
Probe design	MLP-64, last token	0.840 [0.824, 0.855]
Probe design	Linear, mean pooled	0.761 [0.739, 0.782]
Probe design	Linear, last-4-layer mean	0.839 [0.823, 0.854]

Table B.5: Qwen3-1.7B probe ablations on RaR-Science-Static with 95% bootstrap confidence intervals.

Scalar-score error. For each candidate response, we compare the judge's aggregated score $\widehat{R}(q, y)$ with the GPT-4o aggregated score $R^*(q, y)$. Mean absolute error is

$$\mathrm{MAE} = \frac{1}{M} \sum_{(q,y)} |\widehat{R}(q, y) - R^*(q, y)|.$$

We also report RMSE when useful.

Score correlation. Pearson correlation measures linear agreement between predicted and reference scalar scores:

$$r = \frac{\sum_i (\widehat{R}_i - \bar{\widehat{R}})(R_i^* - \bar{R}^*)}{\sqrt{\sum_i (\widehat{R}_i - \bar{\widehat{R}})^2} \sqrt{\sum_i (R_i^* - \bar{R}^*)^2}}.$$

Spearman correlation is the same correlation computed over average ranks rather than raw scores.

Pairwise ordering. When two candidate responses are available for the same question, we compare the ordering induced by the judge's scalar scores with the ordering induced by GPT-4o scores. A pair is correct if both judges prefer the same response, or if both assign a tie.

B.6 Bootstrap Confidence Intervals

We compute 95% bootstrap confidence intervals by resampling questions with replacement. For PointRubric, each resampled question contributes its four candidate responses; for RaR-Science, each resampled question contributes its two candidate responses. Tables B.3 and B.4 report intervals for the main static judge results, and Table B.5 reports intervals for the probe ablations.

B.7 Probe Layer Selection

Table B.2 gives the Qwen3-1.7B Probe layer sweep used for development-set layer selection. The selected layer is bolded. Nearby middle-to-late layers have similar held-out performance, so the result is not dependent on a single favorable layer.

C Rubric RL Details

Table 5 contains the main controlled RL comparison. The tables below provide implementation details, efficiency measurement details, and supporting validation evidence for that comparison.

C.1 RL Reward Configuration

The controlled RL comparisons keep the actor, data, rollout, optimizer, reward aggregation, checkpoint, and final evaluator fixed; the reward judge is the only experimental variable. Table C.1 gives the shared configuration used for the matched rows in Table 5.

Metric	Value
Actor	Qwen3-4B-Base (Yang et al., 2025)
Task	RaR-Science (Gunjal et al., 2025)
Train prompts	1,600
Seed	42
RL algorithm	GRPO (Shao et al., 2024)
Framework	VERL (Sheng et al.)
Reward variable	Judge model only
Responses / prompt	4
Sampling	Temperature 1, top-p 1
Max prompt length	1024
Max response length	1536
Train batch size	8
PPO mini-batch size	8
PPO micro-batch size	1 per GPU
PPO epochs / update	1
Optimizer	AdamW (Loshchilov and Hutter, 2017)
Learning rate	1×10^{-6}
Weight decay	0.01
Gradient clipping	1
Actor KL coefficient	0.02
Reward KL	None
Training epochs	5
Save frequency	200 steps
Comparison checkpoint	Step 800
Validation during RL	50 prompts every 50 steps
Final evaluation	100 reserved prompts
Final judge	GPT-4o (Hurst et al., 2024)
Reward aggregation	Absolute rubric weights

Table C.1: Shared RL configuration for the controlled RaR-Science reward-model comparison.

Reward judge	Score	95% CI
None	0.232	[0.170, 0.301]
Qwen3-0.6B Probe	0.562	[0.490, 0.631]
Qwen3-1.7B Probe	0.643	[0.573, 0.708]
Qwen3-4B Probe	0.588	[0.514, 0.659]
Qwen3-8B Probe	0.506	[0.432, 0.575]
Qwen3-8B Generative	0.594	[0.523, 0.662]

Table C.2: Final RaR-Science RL scores with 95% bootstrap confidence intervals.

The generative-judge RL baseline uses the explicit per-criterion protocol. Each criterion is scored by a Qwen3-8B judge generation, parsed into a Yes/No verdict, and aggregated with absolute rubric weights. This is the generated reward used in the main comparison.

The probe reward uses a frozen judge model and a calibrated linear classifier. Each criterion is rendered in the same canonical format used for static evaluation, scored as a criterion-satisfaction probability, and aggregated with the same absolute-weight rubric rule. The judge model and probe are fixed during RL; only the policy actor is updated.

C.2 RL Score Confidence Intervals

We compute 95% bootstrap confidence intervals for final RL evaluation by resampling the 100 reserved RaR-Science evaluation prompts with replacement. Table C.2 reports the resulting intervals for the controlled reward-model comparison.

C.3 Efficiency Measurement

The efficiency comparison in Section 5.3 is computed from the training traces and validation reward metadata of the matched Qwen3-1.7B probe and Qwen3-8B generative-judge RL runs. The generative judge therefore has 4.7× as many backbone parameters as the probe judge. For each run, the training traces record per-step reward-judge time and a cumulative judge-time counter. The cumulative counter includes reward scoring during training and the scheduled validation passes up to the comparison checkpoint. The corresponding per-step training reward totals alone are 8,045.9 seconds for the probe reward and 84,662.4 seconds for the generated reward, giving a 10.5× ratio. Including scheduled validation reward scoring yields 8,389.9 seconds and 89,912.1 seconds, respectively, giving the 10.7× ratio reported in Table 8.

The validation latency calculation uses the reward metadata from the comparison checkpoint. Both reward models score the same 50 responses, covering 385 total rubric criteria. The probe reward records 31.1 seconds of total judge time (0.6216 seconds per response and 0.0807 seconds per criterion). The generated reward records 492.0 seconds of total judge time (9.8410 seconds per response and 1.2780 seconds per criterion). Both runs have sample-level parse success 1.0 on this validation pass. This parse success does not remove the structural difference between the reward computations: the generated reward obtains criterion labels by decoding and parsing Yes/No verdicts, while the probe reward obtains criterion probabilities from a frozen hidden-state classifier.

The elapsed training times in Table 8 come from the progress traces at the comparison checkpoint. Their ratio is smaller than the cumulative judge-time ratio for two reasons. First, the judge-time counter is an aggregate over reward calls, and those calls can run concurrently. Second, actor rollout, reference-model scoring, optimization, synchronization, and checkpointing remain in the critical path for both runs. The elapsed time still decreases from 11h 11m 56s to 8h 51m 47s through the com-

Panel	Measurement	Result	Notes
Preference counts	Human preference on 100 blind A/B pairs	Base 6; Probe 72; Tie 13; Unsure 9	Probe policy preferred on 72% of examples.
Preference counts	GPT-4o rubric preference on the same 100 pairs	Base 9; Probe 80; Tie 11	Same candidate pairs and rubric scoring protocol.
Human-tie analysis	Human-tie subset	13 / 13 human ties	Tie labels are analyzed separately from preferences.
Human-tie analysis	Small GPT-4o score gaps among human ties	7 / 13 have $\lvert\Delta\rvert \leq 0.05$; 11 / 13 have $\lvert\Delta\rvert < 0.20$	Most human ties are also near-ties under the rubric judge.
Preference agreement	Human and GPT-4o preferences after excluding human Tie/Unsure labels and GPT-4o exact ties	69 / 72 agreement	Counts: 3 base/base, 2 base/Probe, 1 Probe/base, 66 Probe/Probe.
Reference-judge check	GPT-5 rescoring on a matched static reference bank	99 / 100 parsed samples	Independent reference rescoring uses the same criterion-level target.
Reference-judge check	Criterion-level GPT-4o/GPT-5 agreement	Macro-F1 0.853; item agreement 0.853	Agreement is measured over parsed criterion labels.
Reference-judge check	Scalar-score GPT-4o/GPT-5 agreement	Pearson 0.843; Spearman 0.841; MAE 0.125; mean bias -0.004	Scores use the same rubric aggregation.

Table C.3: Human audit and reference-judge validation for the main RL comparison.

parison checkpoint, a 2h 20m 09s reduction. We therefore use cumulative judge time as the primary efficiency metric for the reward model and report elapsed training time as the full-pipeline runtime effect.

C.4 Human Audit and Reference-Judge Validation

The human audit evaluates whether the main RL improvement is visible beyond the GPT-4o rubric score. We compare Qwen3-4B-Base against the Qwen3-1.7B probe-reward policy on 100 reserved RaR-Science examples. The audit is model-name blind with randomized A/B order. For each example, the annotator sees the question, reference answer, rubric titles, and the two candidate responses, then marks pairwise preference and optional artifact notes.

The audit favors the probe-reward policy on 72 examples, the base actor on 6, with 13 ties and 9 unsure labels. Among the 13 human ties, 7 have an absolute GPT-4o score gap at most 0.05 and 11 have a gap below 0.20 (Table C.3), indicating that many human ties are also near-ties under the rubric judge. When human Tie/Unsure labels and GPT-4o exact ties are excluded, the human preference and GPT-4o rubric preference agree on 69 of 72 examples (Table C.3). These results are used as supporting validation evidence for the operational GPT-4o target, not as supervised labels for training the reward model.

Table C.3 also reports a reference-judge check using GPT-5 (Singh et al., 2025) on a matched static reference bank.

C.5 Additional RL Configuration Checks

During development, we ran additional RL checks that varied KL coefficient, maximum response length, checkpoint choice, and probe calibration size. These runs were used to choose a stable matched comparison protocol. We do not report them as competing results because they vary training conditions rather than isolating the reward judge. The main paper therefore reports only the controlled comparison in Table 5, where actor, data, GRPO configuration, checkpoint, response length, reward aggregation, and GPT-4o evaluation are fixed across reward models.

D Generalization Evaluation Details

D.1 GPQA-Diamond Evaluation

The GPQA-Diamond evaluation uses the GPQA-Diamond split with 198 questions (Rein et al., 2023). Each example is converted into a four-choice prompt. The answer choices are permuted

across four runs, and the model is asked to place the final answer letter inside \boxed{}. Accuracy is computed against the permuted gold answer. When the boxed-answer parser fails, GPT-4o (Hurst et al., 2024) is used only as a fallback verifier for the final choice. Table 6 gives the mean and standard deviation across the four answer-order runs.

D.2 Cross-Domain Probe Transfer

For cross-domain transfer, a Qwen3-1.7B linear probe (Yang et al., 2025) is fit on source-domain criterion labels and evaluated on target-domain examples without updating the probe on target-domain labels. Each source domain uses 350 training questions and 50 development questions for layer and threshold selection. Evaluation uses 100 target-domain questions scored by GPT-4o (Hurst et al., 2024) under the same pointwise criterion-satisfaction target used in the static experiments. The metrics in Table 7 compare the probe predictions with GPT-4o criterion labels and aggregated rubric scores.

E Prompts for Reproducibility

This appendix lists the prompt templates used for response construction and criterion-level rubric scoring. Bracketed fields are replaced with the corresponding dataset values. For local Qwen judges, the user prompt is wrapped with the model chat template before inference. The reference rubric-scoring prompt is used for GPT-4o labels; generative, logprob, and probe judges use the itemwise criterion prompts.

Generative Criterion Scoring

```
You are evaluating whether an AI response satisfies a
      specific rubric criterion.

[Question]
{question}

[Criterion to Evaluate]
Title: {title}
Description: {description}

[AI Response]
{response}

Evaluate whether the AI response satisfies this criterion.
Output exactly one final line and no explanation:
- 1 if the criterion is satisfied
- 0 if the criterion is NOT satisfied

Your entire response must be exactly one of:
\boxed{{0}}
\boxed{{1}}
```

Reference Rubric Scoring

```
You are an impartial rubric grader.

Instruction:
{question}

Candidate Response:
{response}

Rubric Rules ({num_rules} total):
{rubric_rules}

Task:
For each rubric rule, decide whether the response
      satisfies it.

Output format requirement:
- Return exactly {num_rules} lines.
- Each line must follow:
  Rule i: <short justification>. Verdict: Yes/No
- Use i = 1..{num_rules} in order.
- Verdict must be exactly `Yes` or `No`.

Do not output JSON, markdown, extra headings, or any text
      outside those rule lines.
```

Forced-choice Logprob Scoring

```
You are an impartial rubric grader.

Instruction:
{question}

Candidate Response:
{response}

Rubric Criterion:
{criterion}

Question:
Does the candidate response satisfy this criterion?

Answer with exactly one word: Yes or No.

Answer:
```

Probe Representation Scoring

```
You are an impartial rubric grader.

Instruction:
{question}

Candidate Response:
{response}

Rubric Criterion:
{criterion}

Question:
Does the candidate response satisfy this criterion?

Answer:
```

RaR-Science Candidate Response Generation

```
{question}
```

PointRubric Full-Score Response Generation

```
You are a highly knowledgeable and versatile AI assistant.
      Your goal is to provide accurate, well-structured,
      and helpful answers.

For the following question, follow these requirements:
1. First, think step-by-step and provide your internal
      reasoning.
2. Then, provide a clear and concise final answer.
3. Your answer must satisfy each criterion in the rubric.

Question: {instruction}

Rubric: {rubric}

--- YOUR ANSWER ---
### Reasoning:
[Your step-by-step reasoning here]

### Final Answer:
[Your final answer here]
```

PointRubric Partial-Compliance Response Generation

```
You are a selective and inconsistent AI assistant. Your
      goal is to provide a response that follows around
      half of the rubric rules while deliberately ignoring
       others.

For the following question, first decide which criteria to
       obey and which to disobey. Then provide a final
      answer that clearly satisfies the chosen criteria
      while noticeably failing the others.

Question: {instruction}

Rubric: {rubric}

--- YOUR ANSWER ---
### Rules you decide to obey:
[List what rules you obeyed and what you disobeyed]

### Reasoning:
[Your reasoning here]

### Final Answer:
[Your final answer here]
```

PointRubric Low-Score Response Generation

```
You are a deceptive and subtle AI assistant. Your goal is
      to provide an answer that appears authoritative but
      is irrelevant to the actual question and fails the
      rubric.

Question: {instruction}

Rubric: {rubric}

--- YOUR ANSWER ---
### Reasoning:
[Your flawed reasoning here]

### Final Answer:
[Your final answer here]
```

PointRubric Unguided Response Generation

```
Question: {instruction}
Your Answer:
```

PointRubric Rubric Standardization

```
You are a precise and analytical editor. Refine and
      reorganize the given rubric into a standardized JSON
       format.

Input:
Question ID: {question_id}
Original Rubric:
{rubric}

Rules:
1. Rules marked `[Hard Rule]` must be type `"hard"`.
2. Rules marked `[Principle]` must be type `"soft"`.
3. Output exactly 6 rules: 2 hard and 4 soft.
4. Merge or split rules as needed while preserving the
      original intent.
5. Each refined rule should be clear, objective, and
      concise.

Output only a JSON array:
[
  {
    "question_id": "{question_id}",
    "rule_id": "1",
    "type": "hard" or "soft",
    "rubric": "..."
  }
]
```

F Dataset Examples

The examples below show the pointwise labels used in the two data settings. We choose responses that are useful but imperfect, so the examples show how the rubric separates high-quality partial compliance from full satisfaction.

PointRubric example. **Prompt.** How do I check if a new install of a version is different from the current one, booting from SBOM?

Candidate response. To determine if a new installation version is different from the current one by comparing SBOMs, obtain the SBOM for the current installed version, generate the SBOM for the new installation, and compare the component lists. Focus on package names, versions, checksums, added or removed packages, dependency changes, and licensing changes. The response also recommends tools such as Syft, CycloneDX, and ORT, and suggests automating the comparison in a deployment pipeline.

GPT-4o score. 9 / 10.

Verdicts. Yes, Yes, Yes, No, Yes, Yes.

RaR-Science example. **Prompt.** An ideal gas at NTP has a most probable velocity $\bar{u} = 4.20 \times 10^4$ cm/sec and a mean free path of 7.90×10^{-8} cm. Calculate the mean time between collision.

Candidate response. The response uses $\tau = \lambda/\bar{u}$ and substitutes the given mean free path and most probable velocity: $\tau = (7.90 \times 10^{-8})/(4.20 \times 10^4) \approx 1.88 \times 10^{-12}$ seconds. The response is clear and numerically consistent, but

Item	Criterion summary	Verdict
Hard 1	Outlines a methodical comparison process using an SBOM.	Yes
Hard 2	Uses the SBOM as part of the comparison procedure.	Yes
Soft 3	Provides multiple approaches with concrete examples or commands.	Yes
Soft 4	Uses clear labeled organization and identifies limitations.	No
Soft 5	Recommends appropriate specialized tools or techniques.	Yes
Soft 6	Explains how to interpret differences and verify assumptions.	Yes

Table F.1: Criterion-level labels for the PointRubric example.

Item	Criterion summary	Verdict
Formula	Applies $\tau = \lambda/\nu$ for mean time between collisions.	Yes
Velocity	Converts most probable velocity to average velocity using $2/\sqrt{\pi}$.	No
Substitution	Substitutes the given numerical values accurately.	Yes
Units	Keeps velocity and length units consistent and reports seconds.	Yes
Derivation	Presents a clear derivation from relation to final value.	Yes
Precision	Uses appropriate significant figures in the final answer.	Yes
Assumptions	Does not introduce extraneous non-ideal effects.	Yes

Table F.2: Criterion-level labels for the RaR-Science example.

it does not convert the most probable velocity to the average velocity before applying the mean-collision-time formula.

GPT-4o score. 0.773.

Verdicts. Yes, No, Yes, Yes, Yes, Yes, Yes.