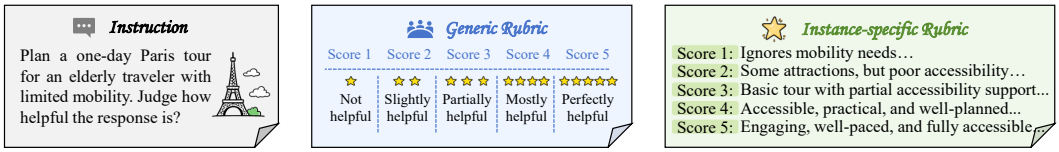
RUBRICRL: RUBRIC-AWARE REINFORCEMENT LEARNING FOR LLM EVALUATION

Anonymous authors
Paper under double-blind review

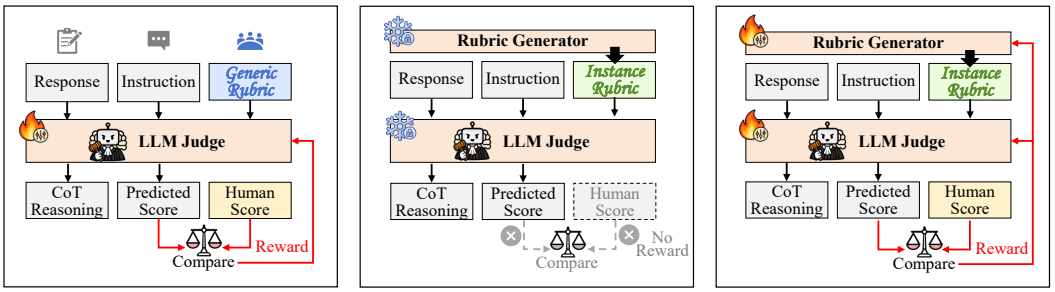
ABSTRACT

Reliable LLM judging requires not only learning *how to evaluate*, but also determining *what to evaluate* for each instance. Existing judge-training methods mainly optimize the former while leaving instance-specific evaluation criteria implicit, whereas rubric-guided approaches make these criteria explicit but typically optimize rubric generation separately from downstream judging. We introduce **RubricRL**, a reinforcement learning framework that couples instance-specific rubric learning with rubric-conditioned judging. RubricRL uses downstream judging performance to optimize instance-specific evaluation rubrics and, in turn, leverages the learned rubrics to guide score prediction within a shared policy model. We further introduce a distributional second-moment reward for absolute scoring quality and a distributional pairwise score-gap reward for relative scoring quality. Extensive experiments across diverse backbones and benchmarks show that RubricRL consistently improves LLM judging, achieves competitive performance with strong closed-source LLMs, and generates rubrics that provide judging performance comparable to human-validated criteria.

1 INTRODUCTION



(a) Example of LLM-as-a-Judge Evaluation with Instance-specific Rubric



(b) Human-Aligned Judge Training (c) Rubric-Guided Judging (d) RubricRL

Figure 1: Motivation of RubricRL. (a) An example of a generic rubric and an instance-specific rubric. (b) Human-aligned Judge training directly optimizes judging against human scores while leaving instance-specific criteria implicit. (c) Rubric-guided judging makes these criteria explicit, but rubric generation remains separate from Judge optimization. (d) RubricRL couples rubric generation with rubric-conditioned judging and alternately optimizes the two processes.

Large language models (LLMs), with their strong language understanding and reasoning capabilities, have increasingly been employed as evaluators for open-ended generation tasks (Liu et al., 2023; Lin & Chen, 2023). This paradigm, known as LLM-as-a-Judge (Zheng et al., 2023), uses capable LLMs to assess candidate responses through numerical scores or pairwise preference judgments. However, directly predicting a score differs from how humans typically form judgments.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Human evaluators often implicitly instantiate an abstract evaluation dimension into a set of fine-grained, instance-specific criteria and assess the response against these criteria. For example, when evaluating the helpfulness of a one-day Paris itinerary for an elderly traveler as presented in Fig 1 (a), one would naturally consider mobility burden, accessibility, and itinerary pacing. This observation suggests that reliable LLM judging can be formulated as two coupled learning problems: **learning *what to evaluate* by constructing instance-specific rubrics, and learning *how to evaluate* by judging responses conditioned on these rubrics.**

Existing judge-training methods in Fig. 1 (b) primarily focus on improving *how to evaluate* by aligning model reasoning and score predictions with human judgments through supervised fine-tuning or reinforcement learning (Chiang et al., 2025; Whitehouse et al., 2026). However, these methods mainly optimize the evaluation outcomes, while the question-specific scoring rules underlying human judgments remain implicit, requiring the Judge to infer which aspects of a response should determine its score. This becomes challenging because the concrete evaluation requirements can vary substantially across questions, even under the same general evaluation criterion.

Rubric-guided judging provides a natural way to make *what to evaluate* explicit by refining a generic evaluation dimension into fine-grained, instance-specific scoring rubrics as illustrated in Fig. 1(c) (Kim et al., 2025; Jinji et al., 2026; Jia et al., 2026). However, constructing reliable rubrics remains challenging. Human-curated rubrics require substantial annotation effort, while automatically generated rubrics still exhibit a considerable gap from expert-annotated ones (Zhou et al., 2026; Liu et al., 2026b). A main reason is that rubric generation is typically optimized separately from downstream judging, so its learning signal does not directly reflect how well the resulting rubric supports human-aligned evaluation. This motivates learning rubrics together with the judging process, allowing their quality to be directly assessed through downstream evaluation performance.

Our key idea is to use downstream judging performance to supervise *what to evaluate*, and then use the learned rubrics to guide *how to evaluate*. Based on this principle, we introduce RubricRL in Fig. 1 (a), a reinforcement learning framework that couples rubric generation and judging within a shared policy model (Zhao et al., 2026; Sheng et al., 2026). During rubric learning, the model samples multiple instance-specific rubrics and favors those that lead to judgments better aligned with annotated scores. During judge learning, the generated rubric is held fixed and used to guide evaluation reasoning and score prediction. The two processes are optimized alternately, allowing downstream judgments to improve rubric learning and the learned rubrics to further improve judging.

We further strengthen *how to evaluate* by modeling judging quality at both the absolute and relative levels. The proposed distributional second-moment reward operates over the predictive score distribution, jointly encouraging accurate score prediction and low predictive uncertainty. The distributional pairwise score-gap reward further aligns the predicted score difference between responses with their annotated gap, capturing both relative ordering and the magnitude of quality differences. Besides, the generalized policy gradient (GPG) (Schulman et al., 2015) is adopted to combine reward-driven trajectory optimization with direct differentiable optimization of score prediction.

We evaluate RubricRL across five datasets and five LLM backbones. RubricRL demonstrates strong overall performance across diverse evaluation settings, outperforming strong LLM-as-a-Judge training baselines and generative reward models on average while achieving competitive performance with strong closed-source LLMs. Moreover, RubricRL-generated rubrics outperform existing automatic rubric generation methods on most evaluation metrics and achieve performance comparable to human-validated rubrics. Our main contributions are summarized as follows:

- We formulate pointwise LLM judging as two coupled learning problems, what to evaluate and how to evaluate. Building on this formulation, we propose RubricRL to couple instance-specific rubric learning with rubric-conditioned judging, using downstream judging quality to optimize evaluation criteria and learned rubrics to guide score prediction.
- We introduce a distributional second-moment reward for absolute scoring quality and a distributional pairwise score-gap reward for relative scoring quality, together with GPG to combine trajectory-level policy optimization with direct differentiable score optimization.
- Extensive experiments across five datasets and five LLM backbones demonstrate that RubricRL consistently improves LLM judging, generalizes to unseen evaluation settings, and generates rubrics that are competitive with human-curated evaluation criteria.

2 Related Work

Human-Aligned Judge Training Recent studies have focused on training LLM-based judges to improve alignment with human evaluations. Prometheus 2 (Kim et al., 2024b) develops a specialized evaluator supporting both pointwise scoring and pairwise comparison under predefined evaluation criteria. TRACT (Chiang et al., 2025) combines chain-of-thought reasoning with regression-aware fine-tuning for numerical score prediction, while J1 (Whitehouse et al., 2026) introduces reinforcement learning with verifiable rewards to improve evaluation reasoning. REAL (Zhang et al., 2026) further incorporates regression-aware rewards and generalized policy gradients to optimize numerical judging. However, such methods primarily optimize *how to evaluate*, while instance-specific scoring rules remain externally specified or implicit in the Judge. In contrast, RubricRL explicitly learns instance-specific rubrics and optimizes them using downstream judging performance.

Rubric-Guided Judging Rubric-guided judging makes evaluation criteria explicit by providing fine-grained rubrics to guide LLM evaluators. BiGGen Bench (Kim et al., 2025) introduces instance-specific rubrics to capture fine-grained requirements across diverse tasks, while RubricBench (Zhou et al., 2026) provides expert-annotated atomic rubrics for systematically evaluating rubric-based judging. Recent studies further explore automatic rubric construction. OpenRubrics (Liu et al., 2026b) develops scalable rubric generation for reward modeling, AutoRubric (Jia et al., 2026) constructs problem-specific rubrics from successful reasoning trajectories, and MAPLE (Jinji et al., 2026) performs evaluation over multiple explicit criteria. Despite these advances, obtaining reliable rubrics remains challenging, as automatically generated rubrics are typically optimized separately from downstream judging. Concurrent work (Xu et al., 2026b) explores coupling rubric generation with evaluator optimization in preference-based reward modeling. In contrast, RubricRL focuses on pointwise LLM-as-a-Judge evaluation and jointly learns instance-specific rubrics and rubric-conditioned numerical judging within a shared policy model, using annotated scores as supervision.

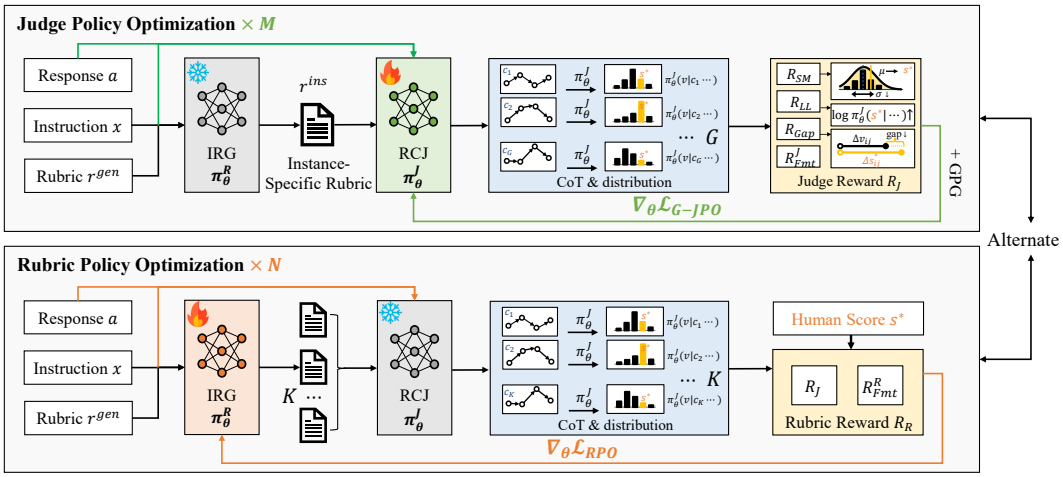


Figure 2: Overview of RubricRL. The Judge Policy Optimization and Rubric Policy Optimization phases alternate with M Judge updates and N Rubric updates per cycle.

3 Methodology

3.1 Problem Formulation

We consider the pointwise LLM-as-a-Judge setting. Each evaluation instance i consists of a task instruction x_i and a candidate response a_i, evaluated under a generic rubric r_d^{gen} that specifies the target evaluation dimension d and its general scoring criteria. We further define an instance-specific rubric $r_{d,i}^{\text{ins}}$ as a fine-grained refinement of r_d^{gen}, capturing the evaluation requirements specific to the instance. Given x_i, a_i, r_d^{gen}, and $r_{d,i}^{\text{ins}}$, the Judge generates an evaluation rationale c_i followed by a score $s_i \in \mathcal{S}$, where $\mathcal{S} = \{1, 2, 3, 4, 5\}$ denotes the score space. The corresponding annotated score is denoted by $s_i^* \in \mathcal{S}$ and serves as the reference judgment. For simplicity, we *omit the dimension index d and instance index i*, and let $r = (r_d^{\text{gen}}, r_{d,i}^{\text{ins}})$ denote the combined rubric.

3.2 Framework for Learning What and How to Evaluate

In Fig. 2, RubricRL couples instance-specific rubric generation with rubric-conditioned judging through two processes. It learns *what to evaluate* by optimizing an *Instance-Specific Rubric Generator* (IRG), denoted by π_θ^R, with Rubric Policy Optimization, and learns *how to evaluate* by optimizing a *Rubric-Conditioned Judge* (RCJ), denoted by π_θ^J, with Judge Policy Optimization. Both roles are instantiated within a shared policy model π_θ and optimized through alternating role-specific objectives. During each phase, the shared parameters are updated only through the active role-specific objective, while gradients associated with the inactive role are stopped. This alternating optimization separates the direct gradient contributions of the two coupled objectives: downstream judging performance provides learning signals for instance-specific rubric generation, while generated rubrics provide instance-specific guidance for judge optimization.

Learning What to Evaluate Given the task instruction x and the generic rubric r^{gen}, the IRG samples K candidate instance-specific rubrics. Here, k indexes the sampled rubric candidates:

$$r_k^{\text{ins}} \sim \pi_\theta^R(\cdot \mid x, r^{\text{gen}}), \qquad k = 1, \ldots, K. \tag{1}$$

The IRG conditions only on the task and generic rubric, without access to the response, thereby encouraging the generated rubric to capture how the task should be evaluated rather than adapt the criteria to a particular response. Conditioning on the generic rubric further allows the instance-specific rubric to provide fine-grained instance-specific guidance while remaining consistent with the general scoring standard. For each rubric candidate, the RCJ produces one downstream judgment,

$$(c_k, s_k) \sim \pi_\theta^J(\cdot \mid x, a, r_k). \tag{2}$$

During this phase, we define the combined rubric as $r_k = (r^{\text{gen}}, r_k^{\text{ins}})$. The RCJ serves only as a downstream evaluator, and gradients are stopped through its computation graph. Since instance-specific rubrics lack direct supervision, we use the quality of the downstream judgment as sequence-level credit for the corresponding rubric candidate (Ahmadian et al., 2024), encouraging the IRG to generate instance-specific rubrics that better support annotation-aligned evaluation.

Learning How to Evaluate Given a fixed rubric r, the RCJ samples G judgment trajectories,

$$(c_g, s_g) \sim \pi_\theta^J(\cdot \mid x, a, r), \qquad g = 1, \ldots, G, \tag{3}$$

where g indexes the sampled judgment trajectory. These judgments are evaluated against the annotated score s^*, providing supervision for improving both evaluation reasoning and scoring. RubricRL alternates between the two phases, performing M steps of Generalized Judge-Policy Optimization followed by N steps of Rubric-Policy Optimization in each cycle. This schedule avoids the full $K \times G$ rollout expansion at each update and separates the gradient updates of the two objectives, improving training efficiency and optimization stability. We next describe the reward design and corresponding optimization objectives.

3.3 Distributional Rewards for Judge Optimization

To learn *how to evaluate*, RubricRL optimizes the RCJ with a composite Judge reward derived from annotations. Among its reward components, we introduce two novel distributional rewards to improve absolute and relative scoring quality, namely the *distributional second-moment reward* and the *distributional pairwise score-gap reward*, respectively. These reward components jointly contribute to Judge optimization, and the resulting Judge reward is further used as downstream credit for rubric learning. We describe the two proposed rewards below.

Distributional Second-Moment Reward Conventional pointwise scoring rewards are typically defined only on the predicted score s, ignoring how probability mass is distributed across all candidate scores in $\mathcal{S}$ and providing no explicit constraint on predictive uncertainty. Thus, we introduce the *distributional second-moment reward*, which evaluates scoring quality over the full predictive distribution of candidate scores. Specifically, given the g-th evaluation reasoning trajectory c_g, the RCJ induces a predictive distribution over candidate scores $v \in \mathcal{S}$, where $v \sim \pi_\theta^J(\cdot \mid x, a, r, c_g)$. We denote the probability assigned to score v by $\pi_\theta^J(v \mid x, a, r, c_g)$. Thus, the reward is then defined as

$$R_{\text{SM},g} = -\mathbb{E}_{v \sim \pi_\theta^J}\left[(v - s^*)^2\right] = -\sum_{v \in \mathcal{S}} \pi_\theta^J(v \mid x, a, r, c_g)(v - s^*)^2. \tag{4}$$

Proposition 1 (Bias–Variance Decomposition) The *distributional second-moment reward* can be decomposed into expected-score error and predictive score variance. Let $\mu_{\theta,g} = \mathbb{E}_{v\sim\pi_\theta^J}[v]$ denote the expected predicted score for the g-th judgment trajectory. Then, Eq. 4 can be written as

$$R_{\mathrm{SM},g} = -\underbrace{(\mu_{\theta,g} - s^*)^2}_{\text{expected-score error}} - \underbrace{\mathbb{E}_{v\sim\pi_\theta^J}\left[(v-\mu_{\theta,g})^2\right]}_{\text{predictive score variance}}. \tag{5}$$

Therefore, maximizing $R_{\mathrm{SM},g}$ simultaneously reduces the expected-score error and predictive score variance, leading to more accurate and confident judgment. The proof is provided in Appendix 8.6.

Distributional Pairwise Score-Gap Reward To improve the Judge's relative scoring capability, we introduce a *Distributional Pairwise Score-Gap Reward* that aligns the predicted score gap between two responses with their annotated score gap. Thereby, this reward captures both the relative ordering between the two responses and the magnitude of their quality difference.

For a fixed task instance, we omit the instance index for clarity and consider two responses a_p and a_q to the same question instance. For simplicity, we omit the trajectory index of the current judgment for a_p. Its predictive score distribution is given by $\pi_\theta^J(v_p \mid x, a_p, r, c_p)$. For the peer response a_q, we average the predictive distributions over its G judgment trajectories as $\bar{\pi}_\theta^J = \frac{1}{G}\sum_{g=1}^{G}\pi_\theta^J(v_q \mid x, a_q, r, c_{q,g})$. We then define the distributional pairwise score-gap reward as

$$R_{\mathrm{Gap},pq} = -\mathbb{E}_{v_p\sim\pi_\theta^J,\, v_q\sim\bar{\pi}_\theta^J}\left[\left((v_p - v_q) - (s_p^* - s_q^*)\right)^2\right]. \tag{6}$$

For the g-th rollout of response a_p, we average $R_{\mathrm{Gap},pq}$ over all valid peer responses a_q under the same question to obtain $R_{\mathrm{Gap},g}$, excluding response pairs with identical annotation scores.

3.4 Unified Multi-Role Policy Optimization

Building on the reward signals defined above, we formulate the optimization objectives for the IRG through Rubric Policy Optimization and for the RCJ through Judge Policy Optimization. The two objectives are optimized alternately. Judge Policy Optimization updates π_θ^J using judge rewards, while Rubric Policy Optimization updates π_θ^R using rubric rewards.

Judge Policy Optimization For Judge optimization, we combine the two distributional rewards introduced above with two auxiliary reward signals. We use a gold-score log-likelihood reward R_{LL} (Zhang et al., 2026) to encourage the RCJ to assign higher probability to the annotated score, together with a format reward R_{Fmt}^J that enforces the required judgment and score output format. The total reward for the g-th judgment trajectory is defined as

$$R_{\mathrm{J},g} = \lambda_s R_{\mathrm{SM},g} + \lambda_g R_{\mathrm{Gap},g} + \lambda_l R_{\mathrm{LL},g} + \lambda_f^J R_{\mathrm{Fmt},g}^J, \tag{7}$$

where the λ terms control the contributions of the corresponding reward components. Given the G judgment trajectories sampled under the same rubric-conditioned input, we compute a group-relative advantage A_g^{J} from their total rewards (Guo et al., 2025). The RCJ is then optimized to increase the likelihood of judgment trajectories with higher relative rewards:

$$\mathcal{L}_{\mathrm{JPO}} = -\frac{1}{G}\sum_{g=1}^{G} A_g^{\mathrm{J}} \sum_{t=1}^{|c_g|} \log \pi_\theta^J\left(c_{g,t} \mid x, a, r, c_{g,<t}\right), \tag{8}$$

Proposition 2 (Generalized Judge-Policy Gradient). Unlike standard policy-gradient settings where rewards are treated as independent of the policy parameters, R_{SM}, R_{Gap}, and R_{LL} depend explicitly on θ through the predictive score distribution of the RCJ. Consequently, the Judge gradient contains not only the trajectory-level policy-gradient term in Eq. 8, but also a direct gradient through these policy-dependent rewards. Following the generalized policy-gradient formulation (Schulman et al., 2015), we define the corresponding score-distribution objective as

$$\mathcal{L}_{\mathrm{Score}} = -\frac{1}{G}\sum_{g=1}^{G}\left(\eta_s R_{\mathrm{SM},g} + \eta_g R_{\mathrm{Gap},g} + \eta_l R_{\mathrm{LL},g}\right), \tag{9}$$

where the η terms control the contributions of each reward component. Combining this direct objective with the trajectory-level objective yields the Generalized Judge-Policy Optimization objective,

$$\mathcal{L}_{\text{G-JPO}} = \underbrace{\mathcal{L}_{\text{JPO}}}_{\text{trajectory optimization}} + \underbrace{\mathcal{L}_{\text{Score}}}_{\text{score optimization}} . \tag{10}$$

Here, $\mathcal{L}_{\text{JPO}}$ optimizes the sampled judgment trajectories through reward-weighted policy gradients, whereas $\mathcal{L}_{\text{Score}}$ directly propagates gradients through the predictive score distribution. The format reward R^J_{Fmt} does not contribute to $\mathcal{L}_{\text{Score}}$, as it is non-differentiable with respect to θ for a fixed sampled trajectory. The derivation is provided in Appendix 8.7.

Rubric Policy Optimization Rubric-Policy Optimization uses two rewards. Let $R_{\text{J},k}$ denote the downstream Judge reward associated with instance-specific rubric candidate r^{ins}_k, computed using the same reward as in Judge-Policy Optimization. It measures how effectively instance-specific rubric r^{ins}_k supports annotation-aligned judging. Let $R^R_{\text{Fmt},k}$ denote the rubric format reward. The total rubric reward R_{R} for instance-specific rubric r^{ins}_k is defined as

$$R_{\text{R},k} = R_{\text{J},k} + \lambda^R_f R^R_{\text{Fmt},k}, \tag{11}$$

where λ^R_f controls the format reward. We then compute a group-relative advantage A^{R}_k over the K rubric candidates sampled for the same question (Guo et al., 2025). The IRG is optimized to increase the likelihood of rubrics that provide more effective downstream evaluation guidance:

$$\mathcal{L}_{\text{RPO}} = -\frac{1}{K}\sum_{k=1}^{K} A^{\text{R}}_k \frac{1}{|r^{ins}_k|}\sum_{t=1}^{|r^{ins}_k|} \log \pi^{\text{R}}_\theta \left(r^{ins}_{k,t} \mid x, r^{gen}, r^{ins}_{k,<t}\right), \tag{12}$$

where $|r^{ins}_k|$ denotes the token length of r^{ins}_k. By assigning aggregated downstream judging performance as sequence-level credit to each rubric candidate, this objective encourages the IRG to generate instance-specific rubrics that better support annotation-aligned evaluation.

Overall, RubricRL alternates between Generalized Judge-Policy Optimization and Rubric-Policy Optimization, performing M Judge updates followed by N Rubric updates in each training cycle. As training proceeds, improved rubrics provide more effective guidance for judging, while a stronger Judge provides more reliable downstream credit for rubric optimization.

4 Experiments

4.1 Experiments Settings

Dataset We train RubricRL on UltraFeedback (Cui et al., 2024), which contains approximately 64K diverse prompts and 256K responses, with four responses generated for each prompt. Each response is annotated with fine-grained feedback across five evaluation dimensions, with scores ranging from 1 to 5 and accompanying textual critiques. We evaluate pointwise LLM-as-a-Judge performance on five evaluation datasets. (1) UltraFeedback 1K. We hold out 250 prompts from UltraFeedback, each paired with four model responses, yielding 1K response-level evaluation instances with no overlap with the training set. (2) Feedback Bench (Kim et al., 2024a). We use its official 1K-example test set, which evaluates fine-grained response quality under diverse scoring criteria. (3) FLASK-Eval (Ye et al., 2024). We use the human-annotated FLASK-Eval released with Prometheus-Eval (Kim et al., 2024a), containing 2K response-level instances evaluated under fine-grained, skill-specific criteria with human scores on a 1-5 scale. (4) HelpSteer3 (Wang et al., 2026). We use the validation split of the Feedback subset, containing 4,078 response-level evaluation instances. Each response is accompanied by up to three human feedback annotations. (5) BiGGen-Bench (Kim et al., 2025). We use its human-annotated evaluation data, comprising 3,236 response-level judgments across a benchmark spanning 77 tasks and nine capabilities.

Models To evaluate the generality of RubricRL across diverse model backbones and scales, we conduct experiments with 5 LLMs: Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026), Phi-4-mini (Abouelenin et al., 2025), Mistral-3-3B (Liu et al., 2026a), and Gemma-4-E4B (Team, 2026).

Table 1: Main results comparing RubricRL with generative reward models and LLM-as-a-Judge baselines across four benchmarks using MAE, RMSE, Pearson (r), and Spearman (ρ). The last column reports the average of each metric across all benchmarks. The best results are shown in bold, and the second-best results are underlined.

Method	UltraFeedback 1K				Feedback Bench				Flask-Eval				HelpSteer3				Average			
	MAE	RMSE	r	ρ	MAE	RMSE	r	ρ	MAE	RMSE	r	ρ	MAE	RMSE	r	ρ	MAE	RMSE	r	ρ
Generative Reward Models																				
Prometheus 2	1.320	1.747	31.7	28.0	**0.479**	**0.797**	**85.1**	**85.4**	1.084	1.502	43.2	38.2	1.134	1.572	15.8	10.4	1.004	1.404	44.0	40.5
Selene1 Mini	1.033	1.418	44.5	44.6	0.529	0.807	82.7	83.4	1.149	1.486	**55.5**	**53.3**	0.929	1.274	27.2	25.1	0.910	**1.246**	**52.5**	**51.6**
DeepSeek-GRM	**0.952**	**1.338**	**53.2**	**52.7**	0.713	1.083	74.9	74.0	**0.995**	**1.390**	50.2	44.5	**0.860**	**1.231**	**31.9**	**26.2**	**0.880**	1.261	**52.5**	49.3
Qwen3.5-4B-Variants																				
CoT	1.177	1.618	47.8	48.6	0.722	0.986	79.1	78.6	0.873	1.282	59.9	56.6	1.062	1.485	34.7	30.8	0.958	1.343	55.4	53.6
TRACT	0.800	1.146	65.4	65.5	0.688	**0.906**	82.8	83.3	0.675	1.041	66.7	61.9	0.966	1.341	39.6	34.5	0.782	1.109	63.6	61.3
J1	1.191	1.608	47.6	50.8	**0.678**	0.935	81.6	81.4	1.030	1.357	57.9	59.0	1.021	1.415	38.5	36.2	0.980	1.329	56.4	56.9
REAL	0.854	1.209	55.9	56.0	0.683	1.022	80.2	79.1	0.641	0.985	66.6	58.3	0.946	1.289	38.1	32.4	0.781	1.126	60.2	56.5
RubricRL	**0.719**	1.050	67.2	67.3	0.697	0.967	83.2	84.5	0.666	0.896	66.1	62.3	0.943	1.257	39.8	34.2	0.756	1.042	64.1	62.1
+ K4G1	0.726	1.000	69.8	70.2	0.703	0.944	84.3	86.0	0.629	0.842	69.3	64.2	0.930	1.230	40.6	34.4	0.747	1.004	66.0	63.7
+ RAIL	0.734	**0.990**	**70.2**	**70.6**	0.719	0.939	**84.8**	**86.5**	**0.612**	**0.817**	**70.5**	**65.6**	**0.917**	**1.204**	**41.7**	**36.8**	**0.746**	**0.988**	**66.8**	**64.9**
Gemma-4-E4B-Variants																				
CoT	1.038	1.479	53.1	53.3	0.645	1.029	79.6	79.7	1.125	1.515	55.4	54.6	1.101	1.540	35.1	32.4	0.977	1.391	55.8	55.0
TRACT	0.961	1.358	53.3	53.3	0.645	0.974	73.7	74.4	0.939	1.223	53.7	51.0	1.032	1.394	36.2	30.8	0.894	1.237	54.2	52.4
J1	1.047	1.423	53.0	52.2	0.694	1.055	82.2	82.8	0.914	1.270	61.7	59.2	1.059	1.445	36.1	31.5	0.929	1.298	58.3	56.4
REAL	0.898	1.286	59.3	59.4	**0.621**	0.931	78.1	78.3	0.907	1.188	63.9	60.9	0.968	1.321	37.3	32.2	0.849	1.182	59.7	57.7
RubricRL	**0.746**	1.072	65.3	65.2	0.629	0.889	82.8	84.1	0.669	0.893	64.9	59.7	0.990	1.305	37.1	30.9	0.759	1.040	62.5	60.0
+ K4G1	0.760	1.032	67.1	67.1	**0.621**	0.845	84.7	86.4	0.642	0.845	66.8	61.0	0.979	1.285	37.5	31.7	0.750	1.002	64.0	61.6
+ RAIL	0.767	**1.020**	**67.6**	**67.7**	0.630	**0.829**	**85.5**	**87.4**	**0.632**	**0.826**	**67.6**	**62.4**	**0.966**	**1.252**	**38.9**	**33.1**	**0.749**	**0.982**	**64.9**	**62.7**
Qwen3.5-9B-Variants																				
CoT	1.101	1.546	53.6	54.2	0.675	1.060	78.6	78.8	1.097	1.503	60.0	60.4	1.111	1.541	40.3	37.1	0.996	1.413	58.1	57.6
TRACT	0.769	1.063	64.1	64.7	**0.609**	**0.863**	84.1	84.7	0.643	0.994	69.5	66.5	0.989	1.366	41.9	36.8	0.753	1.072	64.9	63.2
J1	1.154	1.575	50.2	52.2	0.638	0.991	79.5	80.2	1.154	1.493	56.5	57.5	1.058	1.427	41.0	39.2	1.001	1.371	56.8	57.3
REAL	0.836	1.240	61.6	61.5	0.642	0.994	78.5	78.4	0.748	1.118	66.2	62.6	0.950	1.349	40.6	36.3	0.794	1.175	61.7	59.7
RubricRL	**0.720**	1.076	66.1	66.3	0.680	0.949	85.0	86.9	0.679	0.888	69.9	65.5	0.938	1.255	43.3	37.5	0.757	1.042	66.1	64.1
+ K4G1	0.724	1.031	68.5	69.0	0.670	0.928	86.1	88.4	0.660	0.866	70.7	66.6	0.919	1.222	45.3	39.5	0.748	1.012	67.6	65.9
+ RAIL	0.728	**1.012**	**69.4**	**69.9**	0.616	0.922	**86.5**	**89.0**	**0.639**	**0.838**	**71.8**	**68.0**	**0.904**	**1.194**	**46.2**	**40.8**	**0.745**	**0.991**	**68.5**	**66.9**

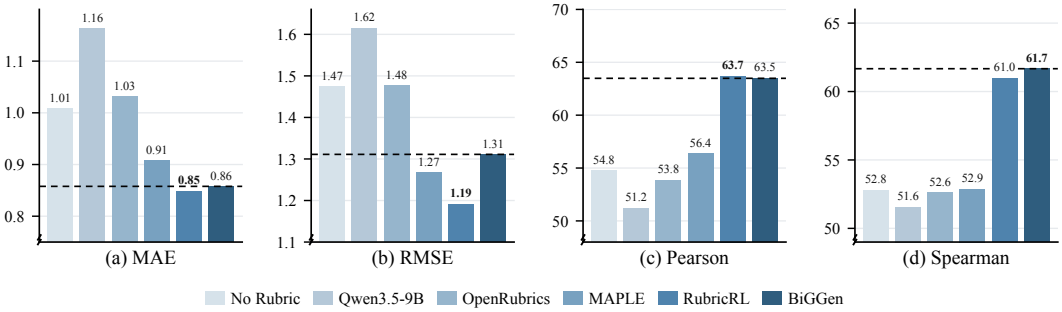


Figure 3: Comparison of instance-specific rubrics on BiGGen-Bench using a fixed Judge. Lower MAE and RMSE and higher Pearson and Spearman indicate better agreement with human ratings. Dashed lines denote the human-validated BiGGen rubric baseline.

Evaluation Metrics We evaluate judgment using mean absolute error (MAE), root mean squared error (RMSE), Pearson correlation coefficient (r) (Pearson, 1895), and Spearman's rank correlation coefficient (ρ) (Spearman, 1961). MAE and RMSE measure absolute scoring accuracy, whereas Pearson and Spearman measure score correlation and ranking consistency with judgments.

Baselines We compare RubricRL with two groups of strong baselines, including generative reward models and LLM-as-a-Judge methods. The generative reward models include Prometheus-7B-v2.0 (Kim et al., 2024b), Selene-1-Mini-Llama-3.1-8B (Alexandru et al., 2025), and DeepSeek-GRM-27B (Liu et al., 2025). For LLM-as-a-Judge baselines, we consider 3-shot CoT prompting together with 3 recently proposed judge-training methods, TRACT (Chiang et al., 2025), J1 (Whitehouse et al., 2026), and REAL (Zhang et al., 2026).

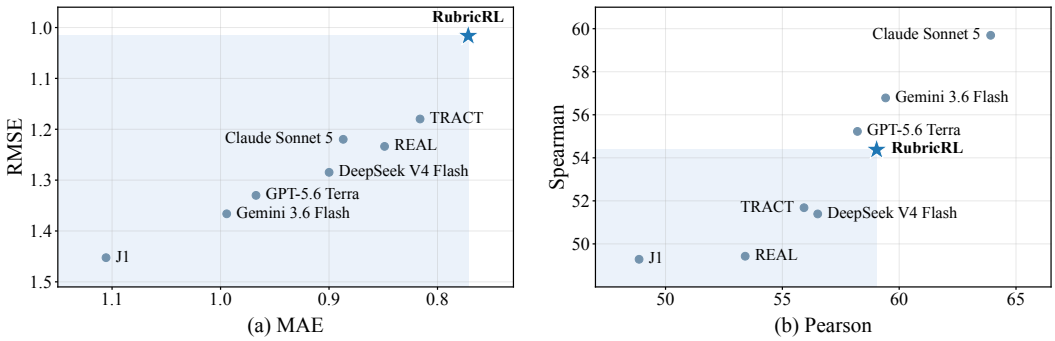


Figure 4: Comparison with strong closed-source LLMs and judge-training baselines on UltraFeedback with better performance toward the upper right in both panels. RubricRL is marked with a star, and shaded regions denote the metric space jointly outperformed by RubricRL.

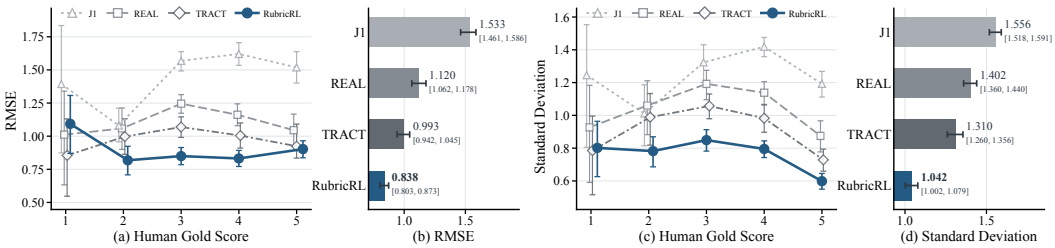


Figure 5: (a) and (c) show RMSE and standard deviation across human gold-score bands, respectively. (b) and (d) report the overall RMSE and the global standard deviation computed over all evaluation examples. Error bars and bracketed ranges denote 95% confidence intervals.

4.2 Main Results

Tab. 1 and Tab. 3 report the judging performance across four benchmarks and five model backbones, aiming to assess its effectiveness across diverse evaluation settings and model families. Overall, RubricRL achieves strong performance across all five backbones and generally outperforms both generative reward models and existing judge-training methods. For example, with Qwen3.5-4B, RubricRL improves the average MAE and RMSE from 0.782 and 1.109 with TRACT to 0.756 and 1.042, while increasing Pearson and Spearman correlations from 63.6 and 61.3 to 64.1 and 62.1, respectively. Similar gains across different model families and scales demonstrate the generality of RubricRL across model backbones.

We further explore two inference-time enhancements for RubricRL. K4G1 averages judgments obtained from 4 independently generated rubrics, while RAIL (Lukasik et al., 2024) predicts the expected score from the model's score distribution instead of directly parsing a discrete score. With Qwen3.5-4B, K4G1 improves the average Pearson and Spearman to 66.0 and 63.7, while RAIL achieves the best overall results with an RMSE of 0.988 and correlations of 66.8 and 64.9. Both strategies further improve overall performance, with RAIL achieving the best average results.

4.3 Instance-Specific Rubric Quality Evaluation

To evaluate the quality of the generated rubrics, we conduct a comparison on BiGGen Bench (Kim et al., 2025) using the same frozen Qwen3.5-9B model as the Judge, while varying only the source of the instance-specific rubric. The rubrics are obtained from the base Qwen3.5-9B model, OpenRubrics (Liu et al., 2026b), MAPLE (Jinji et al., 2026), RubricRL, or the dataset-provided BiGGen rubrics, with rubric-free judging included as an additional baseline.

As shown in Fig. 3, RubricRL achieves the strongest overall performance among automatic rubric-generation methods, obtaining the lowest MAE and RMSE and the highest Pearson and Spearman correlations. It also performs better than the human-validated rubrics in BiGGen, while achieving a slightly lower Spearman correlation. In contrast, rubrics generated directly by the base Qwen3.5-9B model perform worse than rubric-free judging, indicating that the benefit does not arise simply from introducing additional rubric text. These results show that RubricRL learns instance-specific rubrics

Table 2: Ablation study on Qwen3.5-4B using UltraFeedback. Components are added cumulatively.

Setting	MAE↓	RMSE↓	Pearson ↑	Spearman ↑
Supervised Fine-Tuning	1.0000	1.3967	48.73	48.79
+ Reinforcement Learning	0.8254	1.1901	60.74	60.26
+ Generalized Policy Gradient	0.8189	1.0617	63.16	63.23
+ Distributional Second-Moment Reward	0.8055	**1.0478**	65.82	65.76
+ Distributional Pairwise Score-Gap Reward	0.7875	1.0522	65.99	65.80
+ Rubric Policy Optimization	**0.7187**	1.0503	**67.19**	**67.30**

that effectively support human-aligned judging and achieve performance comparable to human-validated rubrics across most evaluation metrics.

4.4 Comparison with Powerful Closed-source LLMs

To assess the judging capability of RubricRL, we further compare it with strong closed-source LLMs and other baselines on UltraFeedback. In Fig. 4, RubricRL achieves the best absolute scoring performance, obtaining the lowest MAE and RMSE among all compared methods. We attribute this advantage to the use of instance-specific rubrics, which translate general evaluation dimensions into fine-grained, instance-specific scoring criteria and thereby reduce ambiguity in mapping responses onto the annotated scoring scale. RubricRL also outperforms judge-training baselines in Pearson and Spearman, but several strong closed-source LLMs achieve higher correlation scores. The closed-source LLMs may benefit from stronger general semantic understanding and cross-instance discrimination, which can improve global correlation and ranking consistency. Nevertheless, RubricRL achieves substantially better absolute score judging while remaining competitive in relative judging.

4.5 Scoring Error and Dispersion across Human Score Levels

To examine scoring consistency across different levels of response quality, we group FLASK-Eval examples by their human ratings and analyze both RMSE and the standard deviation of predicted scores. As shown in Fig. 5, RubricRL achieves lower RMSE in the intermediate score ranges and exhibits smaller score dispersion in the higher score ranges, while outperforming the competing methods across most score bands. When evaluated over all responses, RubricRL achieves both the lowest overall RMSE and the lowest standard deviation. These results are consistent with the intended effect of the Distributional Second-Moment Reward, with RubricRL yielding more accurate and less dispersed score predictions across human score levels.

4.6 Ablation Study

Tab. 2 presents an ablation study on Qwen3.5-4B using UltraFeedback. Reinforcement learning for annotation-score alignment substantially improves over SFT, while Generalized Policy Gradient and the two distributional rewards provide further gains in scoring accuracy and correlation. Rubric Policy Optimization yields the best overall performance, achieving an MAE of 0.7187 and Pearson and Spearman correlations of 67.19 and 67.30. These results show that each proposed component contributes to the overall performance improvement of RubricRL.

5 Conclusion

In this work, we introduce RubricRL, a reinforcement learning framework that couples learning *what to evaluate* through instance-specific rubric generation with learning *how to evaluate* through rubric-conditioned judging. The two processes are optimized alternately, with downstream judging performance providing feedback for rubric learning and the learned rubrics guiding subsequent judge optimization. We further introduce a distributional second-moment reward and a distributional pairwise score-gap reward to improve absolute and relative scoring quality, respectively. Extensive experiments across diverse LLM backbones and evaluation benchmarks show that RubricRL achieves strong overall judging performance, outperforming judge-training baselines and generative reward models on average while remaining competitive with strong closed-source LLMs. Moreover, RubricRL generates instance-specific rubrics that perform comparably to human-validated rubrics.

6 AI USE STATEMENT

We used generative AI tools to assist with language polishing and code development to improve the clarity and grammatical quality of the manuscript and to support tasks such as code debugging and implementation refinement. All research ideas, methodological design, experiments, analysis, and conclusions were developed and verified by the authors.

7 REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the training data preparation, model configuration, and reward functions in the main paper and appendix. Complete derivations of the theoretical results and additional experiment results are also included in the supplementary material to facilitate reproducibility.

REFERENCES

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

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.

Anthropic. Introducing claude sonnet 5, June 2026. URL https://www.anthropic.com/news/claude-sonnet-5.

Cheng-Han Chiang, Hung-yi Lee, and Michal Lukasik. Tract: Regression-aware fine-tuning meets chain-of-thought reasoning for llm-as-a-judge. In *Proceedings of the 63rd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 2934–2952, 2025.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: boosting language models with scaled ai feedback. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.183. URL https://aclanthology.org/2023.emnlp-main.183/.

Google DeepMind. Gemini 3.6 flash model card, July 2026. URL https://deepmind.google/models/model-cards/gemini-3-6-flash/.

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

Shengding Hu, Yifan Luo, Huadong Wang, Xingyi Cheng, Zhiyuan Liu, and Maosong Sun. Won't get fooled again: Answering questions with false premises. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5626–5643, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.309. URL https://aclanthology.org/2023.acl-long.309/.

Mengzhao Jia, Zhihan Zhang, Ignacio Cases, Zheyuan Liu, Meng Jiang, and Peng Qi. Autorubric: Rubric-based generative rewards for faithful multimodal reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25707–25724, 2026.

Michinori Jinji, Kyohei Atarashi, Koh Takeuchi, and Hisashi Kashima. Maple: Multi-aspect panels of llm evaluators for open-ended questions. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 27071–27088, 2026.

Seungone Kim, Jay Shin, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Ryan Shin, Sungdong Kim, James Thorne, Minjoon Seo, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations*, volume 2024, pp. 29927–29962, 2024a.

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 (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.248. URL https://aclanthology.org/2024.emnlp-main.248/.

Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, et al. The biggen bench: A principled benchmark for fine-grained evaluation of language models with language models. In *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)*, pp. 5877–5919, 2025.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Yen-Ting Lin and Yun-Nung Chen. LLM-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models. In Yun-Nung Chen and Abhinav Rastogi (eds.), *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, pp. 47–58, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.nlp4convai-1.5. URL https://aclanthology.org/2023.nlp4convai-1.5/.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026a.

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. Openrubrics: Towards scalable synthetic rubric generation for reward modeling and llm alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17417–17437, 2026b.

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 Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

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.

Michal Lukasik, Harikrishna Narasimhan, Aditya Krishna Menon, Felix Yu, and Sanjiv Kumar. Regression aware inference with llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13667–13678, 2024.

OpenAI. Gpt-5.6 system card, July 2026. URL https://deploymentsafety.openai.com/gpt-5-6.

Karl Pearson. Note on regression and inheritance in the case of two parents. *proceedings of the royal society of London*, 58:240–242, 1895.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015.

Leheng Sheng, Wenchang Ma, Ruixin Hong, Xiang Wang, An Zhang, and Tat-Seng Chua. Reinforcing chain-of-thought reasoning with self-evolving rubrics. *arXiv preprint arXiv:2602.10885*, 2026.

Charles Spearman. The proof and measurement of association between two things. 1961.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Zhilin Wang, Jiaqi Zeng, Olivier Delalleau, Hoo-Chang Shin, Felipe Soares, Alexander Bukharin, Ellie Evans, Yi Dong, and Oleksii Kuchaiev. Helpsteer3-preference: Open human-annotated preference data across diverse tasks and languages. *Advances in Neural Information Processing Systems*, 38, 2026.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason E Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 10397–10420, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026a.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CfXh93NDgH.

Ran Xu, Tianci Liu, Zihan Dong, Tony Yu, Ilgee Hong, Carl Yang, Linjun Zhang, Tuo Zhao, and Haoyu Wang. Alternating reinforcement learning for rubric-based reward modeling in non-verifiable LLM post-training. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=3SDg8dtkS8.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. Flask: Fine-grained language model evaluation based on alignment skill sets. In *International Conference on Learning Representations*, volume 2024, pp. 55361–55414, 2024.

Yasi Zhang, Tianyu Chen, Mingyuan Zhou, Oscar Leong, Ying Nian Wu, and Michal Lukasik. Real: Regression-aware reinforcement learning for llm-as-a-judge. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*, 38:105816–105879, 2026.

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.

Junyi Zhou, Qiyuan Zhang, Yufei Wang, Fuyuan Lyu, Yidong Ming, Can Xu, Qingfeng Sun, Kai Zheng, Peng Kang, Xue Liu, et al. Rubricbench: Aligning model-generated rubrics with human standards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31179–31200, 2026.

8 Appendix

8.1 Other Rewards in Judge Reward and Rubric Reward

We describe the auxiliary reward components used in the Judge and Rubric reward formulations, in addition to the proposed distributional second-moment and pairwise score-gap rewards. The numerical settings below correspond to the default training configuration.

8.1.1 Gold-Score Log-Likelihood Reward

To encourage the Judge to assign higher probability to the annotated score, we use the gold-score log-likelihood reward (Zhang et al., 2026)

$$R_{\mathrm{LL},g} = \log \pi_\theta^J(s^* \mid x, a, r, c_g). \tag{13}$$

Here, π_θ^J denotes the original language-model probability of the score token sequence. Maximizing $R_{\mathrm{LL},g}$ is equivalent to minimizing the negative log-likelihood of the gold score.

8.1.2 Judge Format Reward

The Judge format reward combines three auxiliary terms that constrain output validity and length:

$$R^J_{\mathrm{Fmt},g} = \alpha_{\mathrm{valid}} R^J_{\mathrm{Valid},g} + \alpha_{\mathrm{min}} R^J_{\mathrm{MinLen},g} + \alpha_{\mathrm{budget}} R^J_{\mathrm{MaxLen},g}. \tag{14}$$

Each component is a non-positive reward, with zero indicating no penalty.

Judge Validity Reward Let $F_g \in \{0, 1\}$ denote an output-format validity indicator, where $F_g = 1$ if the judgment satisfies all required format checks and $F_g = 0$ otherwise. These checks require a parseable score following the `[RESULT]` marker, compliance with the configured score range and integer-score constraint, and the absence of prohibited thinking markers. Accordingly, we define the Judge validity reward $R^J_{\mathrm{Valid},g}$ such that valid outputs receive zero penalty and invalid outputs receive a penalty of -1:

$$R^J_{\mathrm{Valid},g} = -(1 - F_g). \tag{15}$$

Minimum Length Reward We introduce a minimum-length reward $R^J_{\mathrm{MinLen},g}$ to discourage overly short evaluation reasoning that may provide insufficient justification for the predicted score. Specifically, trajectories with fewer than $L_{\mathrm{min}} = 64$ reasoning tokens receive a length-dependent penalty.

Maximum Length Reward We further introduce a maximum-length reward $R^J_{\text{MaxLen},g}$ to discourage unnecessarily long completions. The penalty is activated when the generated length exceeds $\rho = 0.75$ of the maximum generation budget $L_{\max} = 256$ and increases as the completion approaches the length limit. Together with the validity reward, these three auxiliary rewards contribute to the trajectory-level Judge reward but do not introduce a direct differentiable gradient with respect to θ for a fixed sampled trajectory.

8.1.3 Rubric Format Reward

The Rubric format reward combines score-band completeness, non-empty content, and length-budget constraints:

$$R^R_{\text{Fmt},k} = \beta_{\text{band}} R^R_{\text{Band},k} + \beta_{\text{empty}} R^R_{\text{Empty},k} + \beta_{\text{length}} R^R_{\text{Length},k}. \tag{16}$$

Score-Band Completeness Reward This reward encourages the instance-specific rubric to cover all five score bands. Let $b_{k,j} = 1$ if the instance-specific rubric includes score band j with a non-empty description, and $b_{k,j} = 0$ otherwise. We define the score-band completeness reward as

$$R^R_{\text{Band},k} = \frac{1}{5} \sum_{j=1}^{5} b_{k,j}. \tag{17}$$

The reward equals the fraction of score bands covered and reaches its maximum when all five bands are present.

Non-Empty Rubric Reward To ensure that each generated rubric contains substantive evaluation criteria rather than only the rubric marker, we penalize empty rubric outputs as follows:

$$R^R_{\text{Empty},k} = -\mathbb{I}[r^{ins}_k \text{ is empty}]. \tag{18}$$

Rubric Length Reward The rubric length reward $R^R_{\text{Length},k}$ encourages concise rubric generation by penalizing instance-specific rubrics that exceed a predefined budget of $L_{max,r} = 220$ tokens, with the penalty increasing with excess length.

8.2 Dataset Details

UltraFeedback UltraFeedback (Cui et al., 2024) is a large-scale feedback dataset containing 63,967 instructions collected from diverse sources, including UltraChat (Ding et al., 2023), ShareGPT (Chiang et al., 2023), Evol-Instruct (Xu et al., 2024), TruthfulQA (Lin et al., 2022), FalseQA (Hu et al., 2023), and FLAN (Wei et al., 2022). For each instruction, four responses are generated by different LLMs. The dataset provides an overall-quality assessment together with fine-grained annotations along four aspects: helpfulness, honesty, instruction following, and truthfulness. The fine-grained aspect ratings use a 1–5 scale, while the original overall-quality score uses a 1–10 scale. In our unified pointwise evaluation setting, we use overall quality and the four fine-grained aspects as five evaluation targets, with scores represented on a common 1–5 scale.

Feedback Bench Feedback Bench (Kim et al., 2024a) is a benchmark for evaluating LLM judges across diverse instructions and response-quality dimensions. It contains 200 instructions and 1,000 response-level evaluation examples that are disjoint from the training portion of the Feedback Collection. Each example pairs a response with an evaluation criterion and an associated 1–5 scoring rubric. The benchmark covers diverse aspects of response quality, requiring the Judge to apply the evaluation standard specified for each example rather than relying on a fixed set of global dimensions. We use the full official test set of 1,000 examples.

FLASK-Eval FLASK (Ye et al., 2024) provides fine-grained evaluation based on 12 alignment skills grouped into four primary abilities: Logical Thinking (Logical Correctness, Logical Robustness, and Logical Efficiency), Background Knowledge (Factuality and Commonsense Understanding), Problem Handling (Comprehension, Insightfulness, Completeness, and Metacognition), and User Alignment (Conciseness, Readability, and Harmlessness). Each instance is evaluated on the

skills relevant to its instruction using predefined 1–5 scoring rubrics. We use the FLASK evaluation data released with Prometheus-Eval (Kim et al., 2024a), comprising 200 test instructions, 12 skill-specific rubrics, and 2,000 response-level evaluation instances with human-annotated scores from the original FLASK evaluation. The human evaluation involved 10 annotators evaluating responses from four anonymized models, with each instance assessed by three annotators.

HelpSteer3 HelpSteer3 (Wang et al., 2026) is a human-annotated dataset covering open-ended interactions across General, STEM, Code, and Multilingual domains. We use the Feedback subset, in which each context is paired with two model responses, and each response receives up to three independent human feedback annotations describing its overall helpfulness. The annotations characterize responses as *not*, *slightly*, *partially*, *mostly*, or *perfectly* helpful, corresponding to an ordinal 1–5 scale. We use the official validation split, which contains 4,078 response-level examples, and aggregate the available human annotations for each response to obtain its reference score.

BiGGen-Bench BiGGen-Bench (Kim et al., 2025) is a fine-grained generation benchmark covering 77 tasks and 3,236 response-level judgments across nine capabilities: instruction following, grounding, reasoning, planning, refinement, multilingualism, safety, theory of mind, and tool usage. Unlike benchmarks based on a small set of generic evaluation dimensions, BiGGen-Bench provides an instance-specific scoring rubric for each input. Each rubric defines explicit criteria for scores from 1 to 5 and is accompanied by a reference answer. The benchmark further provides human ratings of model responses, enabling direct measurement of agreement between LLM judges and human evaluators. We use the human-annotated subset for evaluating both judging performance and the quality of generated instance-specific rubrics.

8.3 Supervised Fine-Tuning Initialization

Before reinforcement learning, we perform supervised fine-tuning (SFT) to initialize the shared policy for both the Instance-Specific Rubric Generator (IRG) and the Rubric-Conditioned Judge (RCJ). We construct a mixed training set from Feedback-Collection (Kim et al., 2024a), consisting of up to 1,000 rubric-generation examples and 1,000 rubric-conditioned judging examples.

For rubric generation, the model receives a task instruction and an evaluation criterion, without access to the candidate response, and learns to generate an instance-specific rubric containing descriptions for all five score levels. For rubric-conditioned judging, the model receives the instruction, candidate response, and corresponding rubric, and learns to produce an evaluation rationale followed by a numerical score marked with `[RESULT]`. These complementary tasks familiarize the shared policy with the input–output requirements of both roles. The resulting SFT adapter initializes the trainable policy for subsequent RubricRL training.

8.4 Configuration

We train RubricRL on 8 NVIDIA H800 GPUs and fine-tune the backbone models using QLoRA (Dettmers et al., 2023). We set $N = 1$, $M = 3$, $K = 4$, and $G = 4$ throughout training. We construct the training set from UltraFeedback (Cui et al., 2024) using five evaluation dimensions and retain only instructions whose associated responses receive distinct scores for each evaluation dimension. Before reinforcement learning, we perform supervised fine-tuning on 1K examples to familiarize the model with the required output format. The default configuration uses a learning rate of 3×10^{-5}.

For the Judge reward, we set $(\lambda_s, \lambda_g, \lambda_l) = (1, 0.4, 0.5)$ for the second-moment, pairwise score-gap, and gold-score log-likelihood terms, respectively, with $\lambda_f^J = \lambda_f^R = 1$ for the composite Judge and Rubric format rewards. For the direct score objective $\mathcal{L}_{\text{Score}}$, we set $(\eta_s, \eta_g, \eta_l) = (0.1, 0.2, 0.05)$. The Judge format reward subweights are $(\alpha_{\text{valid}}, \alpha_{\min}, \alpha_{\text{budget}}) = (1, 0.1, 0.25)$, and the Rubric format reward subweights are $(\beta_{\text{band}}, \beta_{\text{empty}}, \beta_{\text{length}}) = (0.2, 0.2, 0.1)$. The hyperparameter sensitivity analyses are presented in the Appendix. 8.9.

8.5 Closed-Source LLM Baselines

We further evaluate RubricRL against four recent closed-source LLMs accessed through their official APIs, including Claude Sonnet 5 (Anthropic, 2026), Gemini 3.6 Flash (Google DeepMind, 2026), GPT-5.6 Terra OpenAI (2026), and DeepSeek-V4-Flash (Xu et al., 2026a). All models are used only for inference. We use the same evaluation inputs and judging protocol across models whenever applicable.

Claude Sonnet 5 is a recent model in Anthropic's Sonnet family, designed to balance strong reasoning capabilities with efficient inference. Gemini 3.6 Flash is a reasoning model from Google designed for efficient inference while retaining strong capabilities across knowledge, reasoning, and multimodal tasks. GPT-5.6 Terra is a model in OpenAI's GPT-5.6 family that balances model capability and inference efficiency. DeepSeek-V4-Flash is an API-accessible model from DeepSeek that supports both thinking and non-thinking inference modes.

8.6 Proof of Proposition 1: Derivation of the Bias–Variance Decomposition

For the g-th judgment trajectory, the distributional second-moment reward is defined as

$$R_{\mathrm{SM},g} = -\mathbb{E}_{v\sim\pi_\theta^J(\cdot|x,a,r,c_g)}\left[(v-s^*)^2\right]. \tag{19}$$

Let

$$\mu_{\theta,g} = \mathbb{E}_{v\sim\pi_\theta^J(\cdot|x,a,r,c_g)}[v]. \tag{20}$$

We decompose the prediction error as

$$v - s^* = (v-\mu_{\theta,g}) + (\mu_{\theta,g} - s^*). \tag{21}$$

Therefore,

$$\mathbb{E}\left[(v-s^*)^2\right] = \mathbb{E}\left[\left((v-\mu_{\theta,g}) + (\mu_{\theta,g}-s^*)\right)^2\right] \tag{22}$$

$$= \mathbb{E}\left[(v-\mu_{\theta,g})^2\right] + (\mu_{\theta,g}-s^*)^2 + 2(\mu_{\theta,g}-s^*)\mathbb{E}\left[v-\mu_{\theta,g}\right]. \tag{23}$$

By the definition of $\mu_{\theta,g}$,

$$\mathbb{E}\left[v-\mu_{\theta,g}\right] = \mathbb{E}[v] - \mu_{\theta,g} = 0. \tag{24}$$

Hence,

$$\mathbb{E}\left[(v-s^*)^2\right] = (\mu_{\theta,g}-s^*)^2 + \mathbb{E}\left[(v-\mu_{\theta,g})^2\right]. \tag{25}$$

Applying the negative sign in the definition of $R_{\mathrm{SM},g}$ gives

$$R_{\mathrm{SM},g} = -(\mu_{\theta,g}-s^*)^2 - \mathbb{E}\left[(v-\mu_{\theta,g})^2\right], \tag{26}$$

which proves Eq. 5.

8.7 Proof of Proposition 2: Derivation of the Generalized Judge-Policy Gradient

For a fixed rubric-conditioned input (x,a,r), let c denote a judgment reasoning trajectory sampled from the RCJ policy π_θ^J. The trajectory-level Judge reward $R_{\mathrm{J}}(c;\theta)$ depends on θ both through the sampling policy and, for the distributional reward terms, explicitly through the predictive score distribution. We therefore define the Judge objective as the expected reward

$$\mathcal{J}_{\mathrm{J}}(\theta) = \mathbb{E}_{c\sim\pi_\theta^J}\left[R_{\mathrm{J}}(c;\theta)\right]. \tag{27}$$

Expanding the expectation over judgment trajectories gives

$$\mathcal{J}_{\mathrm{J}}(\theta) = \sum_c \pi_\theta^J(c \mid x,a,r) R_{\mathrm{J}}(c;\theta). \tag{28}$$

Taking the gradient with respect to θ yields

$$\nabla_\theta \mathcal{J}_{\mathrm{J}}(\theta) = \sum_c \nabla_\theta\left[\pi_\theta^J(c \mid x,a,r) R_{\mathrm{J}}(c;\theta)\right] \tag{29}$$

$$= \sum_c R_{\mathrm{J}}(c;\theta)\nabla_\theta \pi_\theta^J(c \mid x,a,r) + \sum_c \pi_\theta^J(c \mid x,a,r)\nabla_\theta R_{\mathrm{J}}(c;\theta). \tag{30}$$

Using the log-derivative identity

$$\nabla_\theta \pi_\theta^J(c \mid x, a, r) = \pi_\theta^J(c \mid x, a, r) \nabla_\theta \log \pi_\theta^J(c \mid x, a, r), \tag{31}$$

we obtain

$$\nabla_\theta \mathcal{J}_{\mathrm{J}}(\theta) = \mathbb{E}_{c \sim \pi_\theta^J} \Big[\underbrace{R_{\mathrm{J}}(c; \theta) \nabla_\theta \log \pi_\theta^J(c \mid x, a, r)}_{\text{trajectory policy gradient}} + \underbrace{\nabla_\theta R_{\mathrm{J}}(c; \theta)}_{\text{score-distribution gradient}} \Big]. \tag{32}$$

For the sampled trajectories $c_1, \ldots, c_G$, we estimate the trajectory-level policy gradient using the group-relative advantage A_g^{J} as the credit assignment signal. Treating A_g^{J} as a stop-gradient quantity, the gradient of generalized Judge loss is given by

$$\nabla_\theta \mathcal{L}_{\text{G-JPO}} = -\frac{1}{G} \sum_{g=1}^{G} \Bigg[\underbrace{A_g^{\mathrm{J}} \sum_{t=1}^{|c_g|} \nabla_\theta \log \pi_\theta^J \left(c_{g,t} \mid x, a, r, c_{g,<t} \right)}_{\text{trajectory policy gradient}} + \underbrace{\nabla_\theta R_{\mathrm{J},g}(\theta)}_{\text{score-distribution gradient}} \Bigg]. \tag{33}$$

The first term corresponds to the trajectory-level objective $\mathcal{L}_{\mathrm{JPO}}$. The second term arises from the explicit dependence of the Judge reward on the predictive score distribution. Recall that

$$R_{\mathrm{J},g} = \lambda_s R_{\mathrm{SM},g} + \lambda_g R_{\mathrm{Gap},g} + \lambda_l R_{\mathrm{LL},g} + \lambda_f^J R_{\mathrm{Fmt},g}^J. \tag{34}$$

For a fixed sampled trajectory, the discrete format reward has no direct differentiable dependence on θ, such that

$$\nabla_\theta R_{\mathrm{Fmt},g}^J = 0. \tag{35}$$

Hence,

$$\nabla_\theta R_{\mathrm{J},g}(\theta) = \lambda_s \nabla_\theta R_{\mathrm{SM},g} + \lambda_g \nabla_\theta R_{\mathrm{Gap},g} + \lambda_l \nabla_\theta R_{\mathrm{LL},g}. \tag{36}$$

For the distributional second-moment reward, let

$$p_{\theta,g}(v) = \pi_\theta^J(v \mid x, a, r, c_g), \qquad v \in \mathcal{S}. \tag{37}$$

Its reward is

$$R_{\mathrm{SM},g} = -\sum_{v \in \mathcal{S}} p_{\theta,g}(v)(v - s^*)^2, \tag{38}$$

and therefore its direct gradient is

$$\nabla_\theta R_{\mathrm{SM},g} = -\sum_{v \in \mathcal{S}} (v - s^*)^2 \nabla_\theta p_{\theta,g}(v). \tag{39}$$

Similarly, consider responses a_i and a_j to the same question. For the current judgment trajectory of a_i, define

$$p_{\theta,i}(v_i) = \pi_\theta^J(v_i \mid x, a_i, r, c_i), \tag{40}$$

and let the predictive score distribution of the peer response a_j be

$$\bar{p}_{\theta,j}(v_j) = \frac{1}{G} \sum_{h=1}^{G} \pi_\theta^J(v_j \mid x, a_j, r, c_{j,h}). \tag{41}$$

The pairwise score-gap reward is

$$R_{\mathrm{Gap},ij} = -\sum_{v_i \in \mathcal{S}} \sum_{v_j \in \mathcal{S}} p_{\theta,i}(v_i) \bar{p}_{\theta,j}(v_j) \left[(v_i - v_j) - (s_i^* - s_j^*) \right]^2. \tag{42}$$

Thus,

$$\nabla_\theta R_{\mathrm{Gap},ij} = -\sum_{v_i, v_j} \left[(v_i - v_j) - (s_i^* - s_j^*) \right]^2 \cdot \nabla_\theta \left[p_{\theta,i}(v_i) \bar{p}_{\theta,j}(v_j) \right]. \tag{43}$$

Averaging this quantity over the valid peer responses gives $\nabla_\theta R_{\mathrm{Gap},g}$.

Finally, the gold-score log-likelihood reward is

$$R_{\mathrm{LL},g} = \log \pi_\theta^J(s^* \mid x, a, r, c_g), \tag{44}$$

whose direct gradient is

$$\nabla_\theta R_{\mathrm{LL},g} = \nabla_\theta \log \pi_\theta^J(s^* \mid x, a, r, c_g). \tag{45}$$

Combining Eqs. 39, 43, and 45, the score-distribution component in Eq. 33 is exactly obtained by minimizing

$$\mathcal{L}_{\mathrm{Score}}^{\mathrm{exact}} = -\frac{1}{G}\sum_{g=1}^{G}\left(\lambda_s R_{\mathrm{SM},g} + \lambda_g R_{\mathrm{Gap},g} + \lambda_l R_{\mathrm{LL},g}\right). \tag{46}$$

Here, λ_s, λ_g, and λ_l are inherited directly from the Judge reward and therefore give the exact direct-gradient component of the expected Judge reward.

In practice, the trajectory-level policy-gradient term and the direct score-distribution gradient can have different optimization scales. We therefore decouple the weighting of the direct score objective from that of the trajectory-level Judge reward by introducing independent coefficients η_s, η_g, and η_l. The score objective used for optimization is

$$\mathcal{L}_{\mathrm{Score}} = -\frac{1}{G}\sum_{g=1}^{G}\left(\eta_s R_{\mathrm{SM},g} + \eta_g R_{\mathrm{Gap},g} + \eta_l R_{\mathrm{LL},g}\right). \tag{47}$$

The coefficients λ_s, λ_g, and λ_l continue to define the Judge reward used for trajectory-level credit assignment, whereas η_s, η_g, and η_l independently control the strength of the corresponding direct score-distribution gradients. When $\eta_s = \lambda_s$, $\eta_g = \lambda_g$, and $\eta_l = \lambda_l$, Eq. 47 recovers the exact gradient component in Eq. 46.

The generalized Judge objective used for optimization is therefore

$$\mathcal{L}_{\text{G-JPO}} = \mathcal{L}_{\mathrm{JPO}} + \mathcal{L}_{\mathrm{Score}}, \tag{48}$$

where $\mathcal{L}_{\mathrm{JPO}}$ accounts for trajectory-level policy optimization and $\mathcal{L}_{\mathrm{Score}}$ directly optimizes the predictive score distribution. This completes the derivation of Proposition 2.

8.8 Additional Main Results

Tab. 3 reports additional results on Phi-4-mini and Mistral-3-3B, complementing the backbone evaluations in the main paper. The overall trends remain consistent across both models. RubricRL substantially improves over CoT and existing judge-training methods, achieving lower average scoring errors and higher correlations with annotation judgments. For Phi-4-mini, RubricRL reduces the average MAE from 0.918 with TRACT to 0.804 and improves Pearson and Spearman correlations from 47.9 and 47.1 to 54.8 and 53.8, respectively. Similar gains are observed with Mistral-3-3B, where RubricRL improves the average MAE from 0.866 to 0.818 and increases Pearson and Spearman correlations from 54.1 and 52.0 to 57.3 and 55.5. The inference-time variants further improve overall performance, particularly in RMSE and correlation metrics. These additional results further support the generality of RubricRL across different model families and scales.

8.9 Hyperparameter Sensitivity Analysis

8.9.1 Learning Rate

We evaluate RubricRL with Qwen3.5-4B under different learning rates in Tab. 4. Models are trained on UltraFeedback and evaluated on the UltraFeedback-1K test set using K4G1+RAIL inference. Among the tested learning rates, 3×10^{-5} achieves the best performance across all four metrics, with an MAE of 0.7512, RMSE of 0.9997, Pearson correlation of 69.28, and Spearman correlation of 69.63. We therefore adopt 3×10^{-5} as the default learning rate.

Table 3: Main results comparing RubricRL with generative reward models and LLM-as-a-Judge baselines across four benchmarks using MAE, RMSE, Pearson (r), and Spearman (ρ). The last column reports the average of each metric across all benchmarks. The best results are shown in bold, and the second-best results are underlined.

Method	UltraFeedback 1K				Feedback Bench				Flask-Eval				HelpSteer3				Average			
	MAE	RMSE	r	ρ	MAE	RMSE	r	ρ	MAE	RMSE	r	ρ	MAE	RMSE	r	ρ	MAE	RMSE	r	ρ
Generative Reward Models																				
Prometheus 2	1.320	1.747	31.7	28.0	**0.479**	**0.797**	**85.1**	**85.4**	1.084	1.502	43.2	38.2	1.134	1.572	15.8	10.4	1.004	1.404	44.0	40.5
Selene1 Mini	1.033	1.418	44.5	44.6	0.529	0.807	82.7	83.4	1.149	1.486	**55.5**	**53.3**	0.929	1.274	27.2	25.1	0.910	**1.246**	**52.5**	**51.6**
DeepSeek-GRM	**0.952**	**1.338**	**53.2**	**52.7**	0.713	1.083	74.9	74.0	**0.995**	**1.390**	50.2	44.5	**0.860**	**1.231**	**31.9**	**26.2**	**0.880**	1.261	**52.5**	49.3
Phi4-mini-Variants																				
CoT	1.441	1.900	26.2	25.2	0.862	1.284	71.6	71.1	0.982	1.427	29.1	26.9	1.126	1.559	12.4	11.5	1.103	1.542	34.8	33.7
TRACT	**0.817**	1.127	58.6	57.9	1.030	1.322	62.0	65.8	**0.851**	1.099	51.8	48.2	0.976	1.313	19.1	16.5	0.918	1.212	47.9	47.1
J1	1.193	1.603	36.8	31.2	0.580	0.859	81.2	81.8	0.859	1.212	40.3	33.6	1.125	1.517	15.4	11.4	0.939	1.298	43.4	39.5
REAL	0.996	1.396	51.3	51.4	1.204	1.677	60.4	58.6	1.024	1.408	39.4	36.7	1.054	1.399	16.4	15.9	1.069	1.470	41.9	40.6
RubricRL	0.823	1.130	59.0	59.3	**0.524**	0.773	84.3	85.4	0.898	1.108	50.8	48.9	**0.971**	**1.299**	25.1	21.7	**0.804**	1.078	54.8	53.8
+ K4G1	0.855	1.112	59.9	59.9	0.556	0.745	86.1	87.2	0.869	1.067	54.0	51.9	1.085	1.430	29.3	24.6	0.841	1.089	57.3	55.9
+ RAIL	0.850	**1.091**	**61.4**	**61.5**	0.563	**0.726**	**87.2**	**88.3**	0.854	**1.046**	**55.8**	**53.1**	1.076	1.417	**29.8**	**25.6**	0.836	**1.070**	**58.6**	**57.1**
Mistral-3-3B-Variants																				
CoT	1.180	1.614	43.2	43.4	0.828	1.216	62.8	63.7	1.314	1.674	45.4	47.7	1.186	1.405	25.3	22.5	1.127	1.477	44.2	44.3
TRACT	0.880	1.119	63.2	62.6	0.860	1.145	66.0	67.0	0.780	1.043	59.2	52.4	**0.944**	**1.259**	27.9	24.9	0.866	1.142	54.1	52.0
J1	1.016	1.372	48.2	47.7	0.692	0.946	81.3	81.8	0.929	1.251	53.1	49.7	1.004	1.334	26.4	24.3	0.910	1.226	52.3	50.9
REAL	0.884	1.297	59.3	58.6	0.797	1.205	66.6	65.0	1.010	1.398	55.6	**55.0**	1.000	1.365	26.9	23.4	0.923	1.316	52.1	50.5
RubricRL	0.815	1.172	62.2	62.4	**0.601**	0.859	82.8	84.4	0.739	0.971	57.7	52.2	1.116	1.377	26.5	23.8	0.818	1.120	57.3	55.5
+ K4G1	0.820	1.131	63.5	63.5	0.614	0.829	84.6	86.6	0.717	0.930	60.5	54.9	1.085	1.330	**29.3**	24.6	0.809	1.080	59.5	57.4
+ RAIL	**0.811**	**1.117**	**64.1**	**64.5**	0.616	**0.813**	**85.5**	**87.5**	**0.709**	**0.921**	**60.6**	54.5	1.085	1.339	29.1	**25.7**	**0.805**	**1.072**	**59.8**	**58.0**

Table 4: Sensitivity analysis of RubricRL on Qwen3.5-4B with different learning rates.

Learning Rate	MAE ↓	RMSE ↓	Pearson ↑	Spearman ↑
2×10^{-6}	0.7895	1.0333	66.67	66.82
3×10^{-6}	0.7832	1.0265	67.23	67.55
4×10^{-6}	0.7742	1.020	67.84	67.97
6×10^{-6}	0.7663	1.0101	68.61	68.61
1×10^{-5}	0.7538	1.0070	68.89	69.03
3×10^{-5}	**0.7512**	**0.9997**	**69.28**	**69.63**
5×10^{-5}	0.7547	1.0050	69.17	69.27

Table 5: Sensitivity analysis of RubricRL on Qwen3.5-4B with different η_s.

η_s	MAE ↓	RMSE ↓	Pearson ↑	Spearman ↑
0.1	**0.7451**	**0.9992**	**69.34**	**69.89**
0.4	0.7633	1.0054	68.75	68.85
0.5	0.7512	0.9997	69.28	69.63

8.9.2 Effect of η_s

We study the sensitivity of RubricRL to η_s, which controls the contribution of the distributional second-moment reward in the direct score objective $\mathcal{L}_{\text{Score}}$. We train RubricRL with Qwen3.5-4B on UltraFeedback with different η_s. As shown in Tab 5, $\eta_s = 0.1$ achieves the best performance across all four metrics, with an MAE of 0.7451, RMSE of 0.9992, Pearson correlation of 69.34, and Spearman correlation of 69.89. We therefore use $\eta_s = 0.1$ as the default setting.

8.9.3 Effect of λ_g

We further examine the sensitivity of RubricRL to λ_g, which controls the contribution of the distributional pairwise score-gap reward in the Judge reward. As shown in Tab. 6, $\lambda_g = 0.4$ achieves the best overall performance, obtaining the lowest RMSE and the highest Pearson and Spearman corre-

Table 6: Sensitivity analysis of RubricRL on Qwen3.5-4B with different λ_g.

λ_g	MAE ↓	RMSE ↓	Pearson ↑	Spearman ↑
0.1	0.7353	0.9976	69.90	70.43
0.2	0.7451	0.9992	69.34	69.89
0.4	**0.7335**	**0.9900**	**70.22**	**70.62**

Table 7: Sensitivity analysis of RubricRL on Qwen3.5-4B with different η_g.

η_g	MAE ↓	RMSE↓	Pearson ↑	Spearman ↑
0.1	0.7593	1.0093	68.76	68.55
0.2	**0.7335**	**0.9900**	**70.22**	**70.62**
0.4	0.7434	1.0015	69.51	70.05

lations, while maintaining the lowest MAE among the tested settings. We therefore use $\lambda_g = 0.4$ as the default value.

8.9.4 Effect of η_g

We further examine the sensitivity of RubricRL to η_g in Tab. 7, which controls the contribution of the distributional pairwise score-gap reward in the direct score objective $\mathcal{L}_{\text{Score}}$. Using the same experimental setting, $\eta_g = 0.2$ achieves the best performance across all four metrics, with an MAE of 0.7335, RMSE of 0.9900, Pearson correlation of 70.22, and Spearman correlation of 70.62. We therefore adopt $\eta_g = 0.2$ as the default setting.

8.9.5 Effect of λ_l

We examine the sensitivity of RubricRL to λ_l, which controls the contribution of the gold-score log-likelihood reward in the Judge reward. As shown in Tab. 8, $\lambda_l = 0.5$ achieves the best performance across all four metrics. We therefore use $\lambda_l = 0.5$ as the default setting.

8.9.6 Rubric–Judge Policy Optimization Update Ratio

As presented in Tab. 9, we examine the sensitivity of RubricRL to the update ratio $N : M$ between Rubric Policy Optimization and Judge Policy Optimization. Among the tested schedules, $N : M = 1 : 3$ achieves the best performance across all evaluation metrics, including the correlation scores and the lowest MAE and RMSE. We therefore adopt $N = 1$ and $M = 3$ as the default alternating optimization schedule.

8.10 Alternating vs. Joint Optimization

We further compare the proposed alternating optimization with a joint optimization variant that updates the IRG and RCJ simultaneously. Under the alternating strategy, Rubric Policy Optimization samples $K = 4$ rubric candidates, with the RCJ fixed and producing one downstream judgment for each rubric. Conversely, during Judge Policy Optimization, the IRG is fixed and provides a single rubric, while the RCJ samples $G = 4$ judgment trajectories. Joint optimization instead evaluates all $K \times G = 16$ rubric–judgment combinations, resulting in substantially higher rollout cost.

Table 8: Sensitivity analysis of RubricRL on Qwen3.5-4B with different λ_l.

λ_l	MAE ↓	RMSE↓	Pearson ↑	Spearman ↑
0	0.7442	1.0491	67.22	67.31
0.25	0.7647	1.0249	67.63	68.34
0.5	**0.7335**	**0.9900**	**70.22**	**70.62**
1	0.7471	0.9905	69.83	69.75

Table 9: Sensitivity analysis of RubricRL on Qwen3.5-4B with different Rubric and Judge Policy Optimization Update Ratios.

N:M	MAE ↓	RMSE↓	Pearson ↑	Spearman ↑
1:2	0.7575	1.0201	67.80	67.70
1:3	**0.7335**	**0.9900**	**70.22**	**70.62**
1:5	0.7497	1.0045	68.94	69.19

Table 10: Comparison between alternating and joint optimization on Qwen3.5-4B.

Optimization	MAE↓	RMSE↓	Pearson↑	Spearman↑
Joint	0.7444	0.9986	69.32	69.54
Alternating	**0.7335**	**0.9900**	**70.22**	**70.62**

Table 11: Comparison of shared and separate parameterization for the IRG and RCJ.

Setting	MAE ↓	RMSE ↓	Pearson ↑	Spearman ↑
RubricRL-share	0.7187	1.050	67.18	67.29
+K4G1	0.7259	1.000	69.82	70.22
+ RAIL	**0.7335**	**0.9900**	**70.22**	**70.62**
RubricRL-separate	0.7424	1.057	66.12	66.27
+K4G1	0.7564	1.022	68.03	68.15
+ RAIL	0.7537	1.015	68.30	68.43

As illustrated in Tab. 10, the alternating strategy also achieves better performance than joint optimization. Alternating optimization separates the gradient contributions of the two role-specific objectives. During each phase, the shared parameters θ are updated only through the corresponding objective, while gradients are stopped through computations associated with the other role. This separation reduces interference between the coupled optimization signals and can lead to more stable training. In contrast, simultaneous updates create stronger coupling between the two objectives, as changes in rubric generation immediately alter the Judge's training signal and vice versa, potentially leading to less stable optimization. Together with the substantially lower rollout cost, these results support alternating optimization as a more efficient and effective training strategy.

8.11 Shared vs. Separate Role Parameterization

We further examine the effect of parameter sharing between the IRG and RCJ by comparing the default shared-policy design with a variant that instantiates the two roles as separate models. As shown in Tab. 11, the shared-policy variant consistently outperforms the separate-model counterpart across all inference settings, with particularly clear gains in correlation. Under RAIL, sharing parameters improves Pearson and Spearman from 68.30 and 68.43 to 70.22 and 70.62, respectively. This suggests that parameter sharing facilitates coordination between rubric generation and judging. Since both roles operate over the same underlying representations, knowledge acquired from their role-specific objectives can transfer across alternating optimization phases, whereas separate models interact primarily through generated rubrics and downstream rewards. Shared parameterization may therefore reduce the coordination gap between learning what to evaluate and how to evaluate.

8.12 Instance-Specific Rubrics Examples

We present examples of an instance-specific rubric generated by RubricRL for a task instance from the evaluation data in Fig 6 and Fig 7. Compared with the generic 1–5 scoring rubric, the instance-specific rubric provides fine-grained criteria tailored to the task requirements, specifying the expected reasoning and correctness conditions for each score level. This illustrates how RubricRL converts general evaluation guidance into task-specific scoring rules.

Instance-Specific Rubric Example 1

Task Instruction

A notebook costs $4. A student buys three notebooks and pays with a $20 bill. How much change should the student receive? Show the calculation.

Judge the overall quality of the assistant response for the given instruction. Consider helpfulness, truthfulness, honesty/uncertainty, and instruction following. Do not score these aspects separately; combine them into one overall Likert score.

Generic Rubric

Use a 1-5 Likert score:

1 = very poor, 2 = poor, 3 = mixed or acceptable, 4 = good, 5 = excellent.

Instance-Specific Rubric

- Score 1: The response provides no relevant solution or uses a fundamentally inappropriate method, such as adding the payment to the purchase cost, without correctly relating the notebook purchase to the change owed.
- Score 2: The response recognizes that purchase cost must be deducted from the payment but makes multiple substantial errors in calculating the cost and change, leaving little reliable progress toward the correct answer.
- Score 3: The response uses the correct multiplication-and-subtraction method and correctly completes one calculation, but a single arithmetic error produces an incorrect final amount, making the solution partially useful but ultimately inaccurate.
- Score 4: The response correctly states that the student receives $8 and provides relevant supporting work, but omits one requested calculation or leaves the connection between the $12 purchase cost and the change insufficiently explicit.
- Score 5: The response clearly shows that three notebooks cost $3 \times \$4 = \$12$, then calculates $\$20 - \$12 = \$8$, and explicitly concludes that the student should receive $8 in change.

Figure 6: Example of an instance-specific rubric for a Math problem.

8.13 Prompts for IRG and RCJ

Fig. 8 presents the prompt template used for the Instance-Specific Rubric Generator (IRG). The `Instruction`, `Scoring Target`, and `Task` fields together correspond to the task instruction x in the main paper, where `Scoring Target` specifies the evaluation dimension and `Task` describes the task to be evaluated. The `Generic Rubric` field provides the general 1–5 scoring criteria associated with the target dimension. Given these inputs, the IRG generates an `Instance-Specific Rubric` consisting of five task-specific score bands from 1 to 5. The candidate response is not included in the prompt, ensuring that the generated rubric remains response-independent.

Fig. 9 presents the prompt template used for the Rubric-Conditioned Judge (RCJ). The input consists of `Instruction`, `Scoring Target`, `Task`, `Generic Rubric`, `Instance-Specific Rubric`, and `Assistant Response`. Conditioned on these inputs, the RCJ generates `Your Judgment`, which contains an evaluation rationale followed by the predicted score.

Instance-Specific Rubric Example 2

Task Instruction

Write a polite email asking to move a meeting from Monday to Wednesday. Keep it under 50 words and explicitly ask whether Wednesday works for the recipient.
Judge how well the assistant response follows the given instruction. Focus on the task goal, requested format, restrictions, style, and any designated method.

Generic Rubric

Use a 1-5 Likert score:
1 = irrelevant, 2 = poor partial focus, 3 = partial compliance, 4 = nearly aligned with minor deviations, 5 = fully aligned with all requirements.

Instance-Specific Rubric

- Score 1: The response does not provide a usable meeting-rescheduling email, discusses an unrelated topic, or directly contradicts the requested move from Monday to Wednesday without offering the requested scheduling request.
- Score 2: The response attempts a rescheduling email but misses multiple central requirements, such as the original meeting day, the proposed Wednesday date, or the request for confirmation, leaving the intended change unclear.
- Score 3: The response clearly requests moving the meeting from Monday to Wednesday but misses one major constraint: it does not ask whether Wednesday works, uses an impolite tone, or exceeds the word limit.
- Score 4: The response requests the correct date change, asks whether Wednesday works, remains polite, and stays under 50 words, but minor awkwardness or unnecessary wording reduces the email's clarity or naturalness.
- Score 5: The response provides a clear, natural, and polite email under 50 words, explicitly requests moving the meeting from Monday to Wednesday, and directly asks whether Wednesday works for the recipient.

Figure 7: Example of an instance-specific rubric for an email writing task.

Instance-Specific Rubric Generator Prompt

Instruction

You are writing an item-specific 1-5 scoring rubric for a response judge. The general target rubric provides the target meaning and 1-5 scoring scale. Write detailed task-specific score bands for judging any assistant response to this instruction. Use the item-specific rubric as the main guidance for this item and the general rubric only as secondary context. Do not assign a score to any response. Do not mention or assume a specific assistant response. Do not repeat the general rubric. The [RUBRIC] tag is already provided below; continue with exactly five numbered score bands. Use this exact format with one score band per line: 1: ..., 2: ..., 3: ..., 4: ..., 5: ... Each score band should be 25-45 words and include concrete instruction-specific evidence. For each score band, describe both content correctness and completeness criteria when relevant. Make adjacent score bands meaningfully different, especially between scores 2/3 and 4/5. Do not use paragraphs, bullet lists, markdown tables, hidden reasoning markers, [JUDGMENT], or [RESULT].

Scoring Target

...

Generic Rubric

...

Task

...

Instance-Specific Rubric

...

Figure 8: Prompt for IRG to generate instance-specific rubrics.

Rubric-Conditioned Judge Prompt

Instruction

You are an assistant-response quality judge. Score only the requested target: {TARGET_LABEL}. Use the item-specific rubric as the primary scoring guidance for this item. Use the general target rubric only as secondary guidance for the target meaning and 1-5 score scale. If the guidance seems in tension, prioritize the item-specific rubric while keeping the final score in the 1-5 range. Write a concise judgment, then output the final score on the last line exactly as: [RESULT] <score> Use plain text only. Do not use hidden reasoning markers or markdown tables. The final score must be an integer from 1 to 5.

Scoring Target

...

Generic Rubric

...

Task

...

Instance-Specific Rubric

...

Assistant Response

...

Your Judgment

...

Figure 9: Prompt for RCJ to generate judgment CoT and score.