RELIABILITY-AWARE LATENT ALIGNMENT UNDER HETEROGENEOUS FEEDBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Aligning large language models requires reward signals that capture complementary aspects of response quality while accounting for differences in feedback reliability. We address this challenge through *reliability-aware latent alignment (RALA)*, which formulates alignment as latent reward inference and fuses discriminative, generative, and endogenous sequence-level scores. A lightweight, gradient-isolated reward head reuses the SFT backbone for policy-specific preference discrimination, while inverse-dispersion weighting balances the numerical contributions of the three reward streams. Under common-target and component-learning assumptions, finite-sample regret bounds connect feedback reliability to downstream policy performance and characterize a sample–reliability trade-off. RALA achieves 62.0% accuracy on RM-Bench Hard and 72.8% on MATH-500, corresponding to relative gains of 7.3% over the strongest evaluated RM-Bench Hard baseline and 8.0% over the strongest evaluated aligned MATH-500 baseline. These results highlight the value of combining complementary reward signals for difficult preference judgments and mathematical reasoning. Code: `https://anonymous.4open.science/r/RALA-F983`.

1 INTRODUCTION

Large Language Models (LLMs) have achieved remarkable success across a wide range of tasks (Zhou et al., 2024; Li et al., 2024; Gallifant et al., 2025). Beyond self-supervised pretraining, aligning model behavior with human intent is essential for safe and effective deployment (Ma et al., 2025). Therefore, Reinforcement Learning from Human Feedback (RLHF) has emerged as the dominant paradigm for alignment by leveraging Reward Models (RMs) trained on human preferences to guide policy optimization (Wang et al., 2024; Zhang et al., 2025a). RLHF requires costly, high-quality human annotations, which significantly limit its scalability. To alleviate this, Reinforcement Learning from Artificial Intelligence Feedback (RLAIF) replaces human annotators with pretrained LLMs acting as automated judges, offering a scalable alternative (Dutta et al., 2024; Yuan et al., 2024). However, Artificial Intelligence (AI)-generated feedback can be less reliable than carefully curated human annotations and may introduce additional uncertainty and systematic bias.

Existing alignment pipelines rely predominantly on a single feedback source, such as human preferences in RLHF or AI evaluations in RLAIF (Ethayarajh et al., 2024; Liu et al., 2025a). These sources may differ in reliability and the aspects of response quality they emphasize. Despite their complementary nature, these feedback sources are often employed in isolation. This raises a fundamental question: *how can we systematically leverage heterogeneous and unreliable feedback for LLM alignment with theoretical guarantees?*

In this paper, we treat LLM alignment as *latent reward inference*. Specifically, heterogeneous feedback supplies complementary observations of a shared alignment target. Direct preference methods such as DPO and its variants optimize preference likelihoods (Rafailov et al., 2023; Azar et al., 2024; Ethayarajh et al., 2024), while self-rewarding and judge-based methods obtain feedback from models (Yuan et al., 2024; Kyrychenko et al., 2025; Agarwal et al., 2024). Our focus is the next step: combining heterogeneous and unreliable feedback in one reward estimator, with an explicit link from feedback quality to policy regret. Further related work is provided in Appendix A.

From both algorithmic and theoretical perspectives, multi-source alignment introduces significant challenges. First, heterogeneous reward signals exhibit different biases, scales, and empirical

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.

dispersions, making naive aggregation unstable, while the quality of externally supplied feedback may vary across sources. Second, maintaining separate policy and discriminative-reward backbones increases the cost of multi-source alignment, while static preference data may poorly match the evolving policy distribution. Finally, the theoretical understanding of learning under unreliable and mixed-quality feedback remains limited, particularly regarding how feedback reliability affects reward estimation and downstream policy performance.

To address these challenges, we propose a unified framework for *reliability-aware latent alignment (RALA)*, which models alignment as latent reward inference under heterogeneous supervision and integrates multiple sequence-level reward estimators through dispersion-balanced aggregation. RALA analyzes source reliability through component estimation and bias terms and adapts reward contributions using output dispersion. RALA differs from single-signal approaches such as generative verification (Zhang et al., 2025c) and endogenous reward modeling (Li et al., 2025b) by jointly integrating externally supervised verification, policy-specific preference discrimination, and an intrinsic policy-confidence signal. The discriminative RM in RALA operates at the representation level by mapping the Supervised Fine-Tuning (SFT) backbone's pooled hidden representation to a sequence-level scalar reward. The central contribution of RALA is a unified latent-reward formulation that connects heterogeneous feedback modeling, shared-backbone discriminative reward learning, practical reward stabilization, and downstream policy guarantees within a single framework. Our main contributions are summarized as:

- **Reliability-aware latent alignment framework:** We propose RALA, a unified framework for LLM alignment that models heterogeneous supervision sources as noisy and potentially biased observations of an underlying latent alignment reward. RALA integrates three complementary sequence-level reward signals: a discriminative reward, a generative verification score, and an endogenous policy-derived score. After aligning their orientations and standardizing their scales, RALA applies inverse-dispersion weighting to reduce the influence of reward components with high empirical dispersion.
- **In-place discriminative reward learning:** We attach a lightweight scalar reward head to the SFT backbone and train it alongside SFT while preventing gradients from the reward loss from updating the shared representation. The resulting discriminative RM operates directly in the representation space: the reward head maps each response's pooled hidden representation to a sequence-level scalar reward and is optimized using a pairwise ranking objective. This avoids maintaining a separate discriminative-RM backbone and enables policy-specific negative samples to be refreshed as the SFT policy evolves.
- **Reliability-aware theoretical guarantees:** We connect feedback reliability, fused reward estimation, and policy performance through finite-sample regret bounds under common-target and component-learning assumptions. The fusion coefficients determine how component-wise estimation errors propagate to the fused reward. Since the generative error bound contains a reliability-dependent term, the resulting regret bound connects feedback quality to downstream policy performance. Response-gap separation yields a sharper regret rate under a Tsybakov-type margin condition and bounded positive gap ratios. The resulting *sample–reliability trade-off* separates two levers for alignment: more feedback reduces estimation error, while more reliable feedback controls the bias term. This gives RLHF and RLAIF an explicit comparison boundary.
- **Extensive evaluation:** We evaluate RALA across diverse alignment tasks, including reward modeling on RM-Bench, code generation on HumanEval and MBPP, and mathematical reasoning on MATH-500 and AIME 2024. RALA reaches 62.0% on RM-Bench Hard and 58.0% on RM-Bench Code, the highest scores among the evaluated reward models on these two slices. It also reaches 72.8% on MATH-500, improving over the strongest evaluated aligned baseline by 8.0% relatively. The RM-Bench Hard gain is 7.3% relatively over the strongest corresponding baseline.

2 PRELIMINARIES

We formulate alignment at the response level. Given a prompt $x \sim \rho$, an LLM induces a policy over complete responses $y = (y_1, \ldots, y_{T_y})$, $p_\theta(y \mid x) = \prod_{t=1}^{T_y} \pi_\theta(y_t \mid x, y_{<t})$, where T_y is the response length. Each RM in RALA assigns one terminal score to a complete response, giving the

three components a common unit of evaluation. PPO implements the policy autoregressively, and the analysis uses its induced response distribution $p_\theta(\cdot \mid x)$.

For analysis, let $\mathcal{Y}(x)$ be the finite candidate set generated for prompt x during a policy update. For each prompt x, the latent alignment reward satisfies $r^*(x, \cdot) : \mathcal{Y}(x) \to \mathbb{R}$. Practical supervision signals are heterogeneous and imperfect observations of this common response-level quantity.

Generalized Reward Learning. The latent reward function r^* representing human preferences is unknown and needs to be estimated from indirect feedback. A common approach is preference learning, where feedback is provided as pairwise comparisons. Given a dataset $\mathcal{D}_{\text{pref}} = \{(x_i, y_{w,i}, y_{l,i})\}_{i=1}^N$ containing prompts and preferred and dispreferred responses, the Bradley-Terry (BT) model (Bradley & Terry, 1952) defines the preference probability as $P(y_w \succ y_l \mid x) = \sigma\left(R(x, y_w) - R(x, y_l)\right)$, where $R(x, y)$ denotes a reward function over complete responses and $\sigma(\cdot)$ is the logistic sigmoid.

Heterogeneous Feedback Modeling. Real-world alignment systems often rely on multiple sources of supervision. We model each feedback source as a heterogeneous projection of the latent alignment reward under source-specific uncertainty, i.e.,

$$r_i(x, y) = \text{Proj}_i(r^*(x, y)) + \epsilon_i(x, y), \tag{1}$$

where Proj_i and r_i denote the projection function and the reward induced by the i-th feedback channel, respectively, and ϵ_i captures source-specific uncertainty and bias. This formulation interprets heterogeneous supervision as a latent-reward inference problem, where different feedback channels provide complementary but imperfect observations of the same latent alignment reward.

Goal. Fix a nonempty finite candidate set $\mathcal{Y}(x)$ for each prompt. Let Π be the set of all measurable response policies supported on these candidate sets, including deterministic choices of any candidate for each prompt. Given $\pi \in \Pi$ and a reward r, define

$$J(\pi; r) \coloneqq \mathbb{E}_{x\sim\rho,\, y\sim\pi(\cdot|x)}[r(x, y)]. \tag{2}$$

Under m feedback channels, we aim to infer a reliable estimate $\hat{r}$ of the latent alignment reward r^* from heterogeneous supervision $\{r_i\}_{i=1}^m$. Let $\pi^* \in \arg\max_{\pi\in\Pi} J(\pi; r^*)$ and $\hat{\pi} \in \arg\max_{\pi\in\Pi} J(\pi; \hat{r})$ be exact optimizers of the latent and learned rewards. We assume measurable maximizing selections exist. These response-level optimizers select reward-maximizing candidates for ρ-almost every prompt; Appendix B.1 accounts for parameterized policy classes through an optimization-gap term. The performance gap induced by reward estimation errors can be quantified via the regret as

$$\text{Rg}(\hat{\pi}) \coloneqq J(\pi^*; r^*) - J(\hat{\pi}; r^*). \tag{3}$$

Bounding the policy regret enables a theoretical characterization of how reward estimation errors propagate to policy performance degradation. Our analysis focuses on how heterogeneous supervision uncertainty and feedback reliability affect reward estimation and downstream alignment performance.

3 Algorithm Design

Following the standard RLHF/RLAIF, alignment typically begins with an SFT stage, where the base language model is fine-tuned on high-quality instruction-following data, followed by Reinforcement Learning (RL)-based optimization under RMs. In this paper, RALA combines three complementary sequence-level reward signals, as shown in Figure 1: a discriminative RM operating on backbone representations, an endogenous score derived from policy likelihoods, and a generative RM trained from human or AI feedback.

3.1 In-Place Dual-Head SFT

RALA begins with a dual-head SFT architecture that reuses one Transformer backbone for language modeling and discriminative response scoring. This design avoids a separate discriminative-RM backbone, isolates the reward-head gradient from language-model training, and refreshes negative

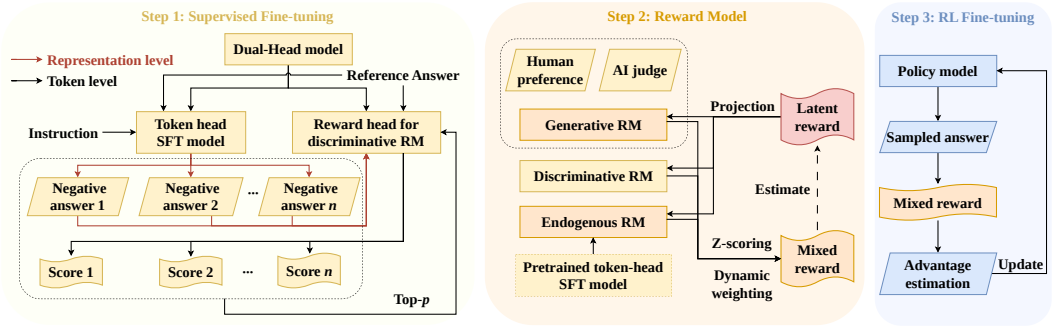


Figure 1: Overview of the proposed three-step framework RALA: **Step 1** jointly trains the language model and a discriminative reward head; **Step 2** aggregates complementary reward estimates under heterogeneous feedback; **Step 3** applies policy optimization with the fused reward.

responses from the evolving SFT policy. We adopt a single Transformer backbone with two linear heads: i) a *token head* optimized via the standard cross-entropy loss on an expert dataset $\mathcal{D}_{\mathrm{SFT}}$ as

$$\mathcal{L}_{\mathsf{SFT}} = -\mathbb{E}_{(x,y^+)\sim\mathcal{D}_{\mathrm{SFT}}}\left[\sum_{t=1}^{T_{y^+}} \log \pi_\theta^{\mathsf{SFT}}\left(y_t^+ \mid y_{<t}^+, x\right)\right], \tag{4}$$

and ii) a *reward head* trained using a pairwise ranking loss as

$$\mathcal{L}_{\mathsf{d}} = -\mathbb{E}_{(x,y^+,y^-)}\left[\log\sigma\left(\hat{r}_{\mathsf{d}}(x,y^+) - \hat{r}_{\mathsf{d}}(x,y^-) - m_k\right)\right], \tag{5}$$

where m_k denotes a scheduled margin at training step k.

Here, y^+ is the reference answer, while y^- is sampled from the current SFT policy via top-p nucleus sampling and treated as a negative relative to the reference. We define $h^+ = f_\theta(x, y^+)$ and $h^- = f_\theta(x, y^-)$ as pooled response representations produced by the shared backbone f_θ, instantiated by the final response-token hidden state. Writing g_ψ for the scalar reward head, we define $\hat{r}_{\mathsf{d}}(x, y) = g_\psi(f_\theta(x, y))$, which assigns one score to each complete response.

To improve training stability and optimization robustness, we employ the following three strategies.

- **Gradient Isolation:** Gradients from Eq. (5) are back-propagated solely through the reward head. This prevents the discriminative reward loss from interfering with language modeling updates during SFT.
- **Self-Generated Negatives:** We generate candidate responses y^- from the evolving SFT policy using top-p nucleus sampling (Ravfogel et al., 2023) and treat them as negatives relative to the reference responses. This couples reward learning with the current policy and trains the RM to distinguish reference responses from policy-generated candidates. The discriminative RM can also be trained using responses sampled from a fixed SFT policy.
- **Curriculum Margin:** We adopt a logarithmically increasing margin $m_k = m_0 + \gamma \log(1 + k/K)$, where m_0 denotes the initial margin, γ controls the growth rate, and K is a normalization constant. Increasing the margin progressively requires a larger score separation between reference and sampled responses. For a fixed score gap, a larger margin increases the magnitude of the loss gradient with respect to that gap.

3.2 Dispersion-Balanced Latent Reward Aggregation

Obtained from the reward head trained alongside SFT, the discriminative RM provides a sequence-level estimate from the shared backbone representation. Following Li et al. (2025b), we employ an endogenous sequence score derived from the SFT model logits. For a response y,

$$\hat{r}_{\mathsf{e}}(x, y) = \frac{1}{T_y}\sum_{t=1}^{T_y} \log \pi_\theta^{\mathsf{SFT}}(y_t \mid x, y_{<t}), \tag{6}$$

which supplies an intrinsic policy-confidence signal to complement externally supervised verification and policy-specific preference discrimination in the fused reward.

To improve reasoning-oriented feedback in a scalable manner, we incorporate a source-agnostic generative RM inspired by Zhang et al. (2025c). Unlike the discriminative head, this RM evaluates generated responses through a verification-token formulation, providing an externally supervised sequence score under both RLHF and RLAIF settings. We define $\hat{r}_{\mathsf{g}}(x, y)$ via the next-token prediction that is directly applicable under both human and AI feedback, i.e.,

$$\hat{r}_{\mathsf{g}}(x, y) = p_\phi(\text{“Yes”} \mid x, y, \mathcal{P}), \tag{7}$$

where p_ϕ denotes the conditional probability given a verification prompt $\mathcal{P}$ under RM parameter ϕ. $\hat{r}_{\mathsf{g}}$ can be instantiated in two complementary ways: either by *prompting a strong off-the-shelf LLM* for zero-shot logical evaluation, or by *fine-tuning on human- or AI-labeled preference pairs* reformatted as verification Question Answering (QA) tasks. Our experiments use a verifier fine-tuned on AI-labeled preference pairs produced by Qwen3-30B-A3B.

Given heterogeneous reward estimators $\{\hat{r}_i\}_{i\in\{\mathsf{d},\mathsf{g},\mathsf{e}\}}$ with different numerical scales and fluctuations, we construct a unified reward for downstream policy optimization. We first apply a fixed orientation and clipping rule to each raw stream so that larger values consistently indicate better responses. From preceding rollout scores, we maintain a running mean μ_i and standard deviation σ_i, and compute

$$\tilde{r}_i(x, y) = (\hat{r}_i(x, y) - \mu_i)/(\sigma_i + \varepsilon).$$

The same pre-standardization σ_i is used for inverse-dispersion weighting. The fused reward is

$$\hat{r}(x, y) = \sum\nolimits_{i\in\{\mathsf{d},\mathsf{g},\mathsf{e}\}} \alpha_i \tilde{r}_i(x, y), \quad \text{with} \quad \alpha_i = (\sigma_i^2 + \varepsilon)^{-1} / \sum\nolimits_j (\sigma_j^2 + \varepsilon)^{-1}. \tag{8}$$

This formulation balances complementary discrimination, verification, and policy-confidence signals by adjusting their numerical contributions using pre-standardization dispersion. Eq. (8) targets reward-scale stability; the theoretical analysis separately tracks component estimation errors and reliability-dependent bias. The fusion-weight ablation evaluates the empirical benefit of adapting these coefficients during training.

Discussion. RALA formulates alignment as reliability-aware latent alignment inference under heterogeneous supervision, where the resulting reward components provide complementary views.

- **Shared-backbone discriminative estimation ($\hat{r}_{\mathsf{d}}$):** A lightweight scalar head scores complete responses from pooled backbone states, providing policy-specific preference discrimination without a separate RM backbone.
- **Logical verification estimation ($\hat{r}_{\mathsf{g}}$):** The probabilistic formulation of the generative verifier standardizes AI-based logic checks and human preference data into a unified confidence signal, whose feedback reliability is theoretically analyzed.
- **Policy-confidence estimation ($\hat{r}_{\mathsf{e}}$):** The endogenous reward adds an intrinsic confidence view to the fused reward, complementing preference discrimination and external verification.

3.3 Policy Optimization

With the dispersion-balanced fused reward $\hat{r}$ in Eq. (8), the final stage follows the standard RL paradigm. We initialize the policy π_θ from the pretrained SFT model $\pi_\theta^{\mathsf{SFT}}$ and optimize it using PPO with the fused reward. Each sampled response receives one terminal fused reward $\hat{r}(x, y)$; PPO uses this terminal score to compute token-level advantage estimates under the clipped objective.

Remark 3.1. Different reward components capture complementary aspects of latent alignment quality. Therefore, relying on a single component may be fragile under feedback corruption and distribution shift. The inverse-dispersion rule controls relative reward fluctuations, while reliability-induced bias is retained explicitly in the theoretical error and regret bounds.

4 Theoretical Guarantees

In this section, we present the theoretical guarantees for RALA. We first establish a general finite-sample regret bound that characterizes the contribution of heterogeneous reward estimation errors and a reliability-dependent bias term from generative feedback. We then refine this analysis under a Tsybakov-type margin condition and a bounded ratio between the largest and smallest reward gaps.

We first formalize feedback reliability through a clean-noisy mixture model.

Definition 4.1 (Feedback Reliability)**.** For a prompt-response pair (x, y) and feedback label z, let $P^\star$, P_{n}, and P_{in} denote the clean, noisy, and observed conditional feedback distributions, respectively. We define the feedback reliability $\bar{\gamma} \in [0, 1]$ as the clean-channel mixture weight:

$$P_{\mathsf{in}}(z \mid x, y) = \bar{\gamma} P^\star(z \mid x, y) + (1 - \bar{\gamma}) P_{\mathsf{n}}(z \mid x, y),$$

A larger $\bar{\gamma}$ indicates a larger clean fraction in the observed feedback.

Let N_{SFT}, N_{d}, and N_{g} denote the numbers of SFT samples, discriminative preference pairs, and generative verification examples, respectively. For each $i \in \{\mathsf{d}, \mathsf{g}, \mathsf{e}\}$, let $\beta_i > 0$ and $d_i > 0$ denote the assumed Hölder smoothness and intrinsic dimension of its target function. Let W_{e} and W_{g} denote the endogenous and generative model-class widths. r^* denotes the centered, scaled clean-feedback target and c_{n} the identically transformed noisy-feedback target. Define $\|u\|^2_{\ell^2(\mathcal{Y}(x))} = \sum_{y \in \mathcal{Y}(x)} |u(y)|^2$ for vector u. Define the nonnegative target-discrepancy magnitude $\mathcal{B}_{\mathsf{noise}}$ by $\mathcal{B}^2_{\mathsf{noise}} := \int \|r^*(x, \cdot) - c_{\mathsf{n}}(x, \cdot)\|^2_{\ell^2(\mathcal{Y}(x))}\, d\rho(x)$. Fix $\eta \in (0, 1)$. The notation $\asymp$ denotes equality of order up to positive constants. Theorem 4.2 transfers the component-learning guarantees to a policy-regret bound with probability at least $1 - \eta$. The proof is given in Appendix B.1; the verification log-loss conversion appears in Appendix B.4.3.

Theorem 4.2 (General Regret Bound for RALA)**.** *Under Assumptions B.2–B.4, define the fused reward $\hat{r}$ by Eq. (8). Choose the endogenous and generative model-class widths as*

$$W_{\mathsf{e}} \asymp N_{\mathsf{SFT}}^{d_{\mathsf{e}}/(d_{\mathsf{e}}+2\beta_{\mathsf{e}})}, \qquad W_{\mathsf{g}} \asymp N_{\mathsf{g}}^{d_{\mathsf{g}}/(d_{\mathsf{g}}+2\beta_{\mathsf{g}})}. \tag{9}$$

With probability at least $1 - \eta$, the regret of its exact optimizer $\hat{\pi} \in \arg\max_{\pi \in \Pi} J(\pi; \hat{r})$ satisfies

$$\mathrm{Rg}(\hat{\pi}) \le \tilde{\mathcal{O}}\left(\alpha_{\mathsf{e}} N_{\mathsf{SFT}}^{-\frac{\beta_{\mathsf{e}}}{d_{\mathsf{e}}+2\beta_{\mathsf{e}}}} + \alpha_{\mathsf{d}} N_{\mathsf{SFT}}^{-\frac{\beta_{\mathsf{d}}}{d_{\mathsf{d}}+2\beta_{\mathsf{d}}}} + \alpha_{\mathsf{d}} N_{\mathsf{d}}^{-\frac{1}{4}} + \alpha_{\mathsf{g}} N_{\mathsf{g}}^{-\frac{\beta_{\mathsf{g}}}{d_{\mathsf{g}}+2\beta_{\mathsf{g}}}} + \alpha_{\mathsf{g}}(1 - \bar{\gamma}) \mathcal{B}_{\mathsf{noise}} \right). \tag{10}$$

The bound accommodates correlated reward errors, including those induced by the shared backbone, and connects convex reward fusion to policy regret. Together with the fusion-weight ablation, it provides a theoretical error-propagation account and an empirical evaluation of the adaptive rule. Appendix B.1 gives general-width bounds and the optimization-gap extension.

The general bound is linear in reward estimation error. To obtain a sharper bound, we impose a Tsybakov-type margin condition controlling small gaps between optimal and suboptimal responses.

Assumption 4.3 (Tsybakov-type Margin Condition)**.** Let $\mathcal{Y}^*(x) = \arg\max_{y \in \mathcal{Y}(x)} r^*(x, y)$ be the set of optimal responses. Define the smallest strictly positive reward gap by

$$\Delta(x) = \max_{y \in \mathcal{Y}(x)} r^*(x, y) - \max_{y \in \mathcal{Y}(x) \setminus \mathcal{Y}^*(x)} r^*(x, y)$$

whenever a suboptimal response exists. Set $\Delta(x) = 0$ when all candidates are optimal; these prompts have zero regret. Let $C_0 > 0$ be a margin constant and $\alpha \in (0, 1)$ a margin parameter. Assume that for every gap threshold $t > 0$, $\mathbb{P}_{x \sim \rho}\big(0 < \Delta(x) \le t\big) \le C_0 t^{\frac{\alpha}{1-\alpha}}$.

The margin parameter α quantifies the scarcity of near-ties between optimal and suboptimal responses. Under the additional gap-ratio condition below, the margin bound improves the exponent of the squared reward error to $1/(2 - \alpha)$ for every $\alpha \in (0, 1)$, with proof in Appendix B.2.

Theorem 4.4 (Fast-Rate Regret Bound for RALA)**.** *Let $C_{\mathrm{gap}} \ge 1$ and assume that, for every prompt with suboptimal candidates, the ratio of the largest to the smallest positive reward gap is at most C_{gap}. Under Assumptions B.2-B.4 and 4.3, with the width choices in Eq. (9), with probability at least $1 - \eta$, the estimation-induced regret of the exact optimizer $\hat{\pi} \in \arg\max_{\pi \in \Pi} J(\pi; \hat{r})$ satisfies*

$$\mathrm{Rg}(\hat{\pi}) \le \tilde{\mathcal{O}}\left(\left[\alpha_{\mathsf{e}} N_{\mathsf{SFT}}^{-\frac{2\beta_{\mathsf{e}}}{d_{\mathsf{e}}+2\beta_{\mathsf{e}}}} + \alpha_{\mathsf{d}} \left(N_{\mathsf{SFT}}^{-\frac{2\beta_{\mathsf{d}}}{d_{\mathsf{d}}+2\beta_{\mathsf{d}}}} + N_{\mathsf{d}}^{-1/2} \right) + \alpha_{\mathsf{g}} \left(N_{\mathsf{g}}^{-\frac{2\beta_{\mathsf{g}}}{d_{\mathsf{g}}+2\beta_{\mathsf{g}}}} + (1 - \bar{\gamma})^2 \mathcal{B}^2_{\mathsf{noise}} \right) \right]^{\frac{1}{2-\alpha}} \right).$$

The bound combines response-gap separation with component-wise estimation rates, while retaining the reliability-dependent bias term.

The ***sample–reliability trade-off*** separates two levers for alignment: more feedback reduces the estimation term, while more reliable feedback reduces the associated bias term for a fixed noise channel. Assuming a smaller human-feedback bias term, the specified power-law bounds favor abundant AI feedback when human data are scarce and favor human feedback beyond an explicit sample threshold. Appendix B.3 derives this boundary, and Appendix C.3.1 visualizes it.

Table 1: Response classification accuracy by RMs on the RM-Bench. The top-2 results in each column are marked in **bold** and underline.

	Chat	Math	Code	Safety	Easy	Normal	Hard	Avg
Steerlm-70B	56.4	53.0	49.3	51.2	48.3	54.9	54.3	52.5
Internlm2-7B-Reward	61.7	**71.4**	49.7	85.5	85.4	70.7	45.1	67.1
Skywork-Reward-Gemma-2-27B	69.5	54.7	53.2	91.9	78.0	69.2	54.9	67.3
URM-LLaMA-3.1-8B	71.2	61.8	54.1	**93.1**	84.0	73.2	53.0	70.0
LLM-as-a-Judge	61.9	35.3	46.1	88.2	63.5	59.3	49.2	57.3
Generative Verifier	60.1	58.5	36.8	86.4	70.7	65.3	57.8	64.6
EndoRM	**84.7**	56.7	53.2	91.2	**86.9**	**74.4**	49.4	**70.2**
RALA (Proposed)	69.8	64.0	**58.0**	82.6	76.1	71.0	**62.0**	69.7

5 EXPERIMENTS

In this section, we evaluate RALA across three distinct domains to answer two key research questions: *whether RALA provides more accurate rewards compared to RMs trained on a single feedback source* and *whether the policy optimized via RALA achieves superior performance on downstream tasks*.

5.1 EXPERIMENTAL SETUP

Training Data. For code generation, following recent practice (Zhang et al., 2025a), we combine 75k samples from Magicoder-OSS-Instruct (Wei et al., 2023) and 55k samples from Evol-CodeAlpaca-v1 (Luo et al., 2024), resulting in a total of 130k instruction-response pairs for SFT. For math reasoning, we adopt Qwen2.5-7B-Math as the base model and conduct evaluation under a fixed SFT initialization to illustrate the extensibility of RALA. The detailed process, corresponding verification prompt templates, and generated training responses are provided in Appendices C.1, D, and E.

Benchmarks. We evaluate RALA with NVIDIA RTX3090×6 GPUs on standard benchmarks covering reward modeling, code generation, and math reasoning. For reward learning, we use RM-Bench (Liu et al., 2025b) to evaluate the quality of learned reward models, following the evaluation protocol of Li et al. (2025b). For code generation, we adopt HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). For math reasoning, we report results on MATH-500 (Hendrycks et al., 2021) and AIME 2024 (American Mathematics Competitions, 2024). See Appendix C.1 for details.

Baselines. We benchmark our framework against several representative baselines spanning both preference optimization and reward modeling paradigms, including DPO (Rafailov et al., 2023), PF-PPO (Zhang et al., 2025a), RLAIF with LLM-as-a-Judge (Zheng et al., 2023), RLAIF with Generative Verifier (Zhang et al., 2025c), and RLHF with EndoRM (Li et al., 2025b).

Table 2 separates alignment-method comparisons from off-the-shelf reference systems. The aligned-model block evaluates the downstream performance of the alignment methods, while the reference block places these results in the broader model landscape.

Implementation Details. We use DeepSeek-Coder-6.7B (Guo et al., 2024) and Qwen2.5-7B-Math (Yang et al., 2024) as our base models for code generation and math reasoning, respectively. For the discriminative RM, we use a lightweight linear reward head. The evaluation metrics and the hyperparameters are provided in Appendices C.1 and C.2. Additional experiment results can be found in Appendix C.3. Tables 3–5 report mean ± sample standard deviation over five random seeds.

5.2 PERFORMANCE OF REWARD ACCURACY

For reward learning, we deploy Qwen2.5-7B-Math-Instruct (Yang et al., 2024) as the base model. We compare RALA against seven baselines, including SteerLM-RM-70B (Duh et al., 2024), InternLM2-7B-Reward (Cai et al., 2024), Skywork-Reward-Gemma-2-27B (Liu et al., 2024), URM-LLaMA-3.1-8B (Lou et al., 2024), LLM-as-a-Judge (Zheng et al., 2023), Generative Verifier (Zhang et al., 2025c), and EndoRM (Li et al., 2025b). As shown in Table 1, RALA reaches 62.0% on Hard and 58.0% on Code, exceeding the strongest corresponding baselines by 4.2 and 3.9 percentage points

Table 2: Performance comparison between RALA and the baselines. RALA scores are seed means rounded to one decimal place; Table 3 reports the corresponding result. The globally best result and the best result among aligned models in each column are marked in **bold** and underline, respectively.

	Code generation		Math reasoning	
	HumanEval	MBPP	MATH-500	AIME 2024
Qwen2.5-7B-Instruct	68.9	61.2	72.0	16.7
DeepSeek-V3	95.1	83.2	**90.2**	36.7
ChatGPT	88.4	73.0	69.8	16.7
GPT-5.2	**95.7**	**86.6**	76.8	**53.3**
DPO	78.7	75.1	59.6	16.7
Original SFT	74.3	69.8	40.0	13.3
PF-PPO	82.9	75.9	64.6	16.7
RLAIF with LLM-as-a-Judge	78.0	72.2	53.3	16.7
RLAIF with Generative Verifier	76.2	68.4	55.6	16.7
RLHF with EndoRM	75.0	70.6	67.4	20.4
RALA (Proposed)	82.9	75.3	72.8	22.0

(7.3% and 7.2% relative gains). RALA sets the best accuracy on the Hard and Code slices, reaching 62.0% and 58.0%. These are the slices that require fine-grained discrimination among subtle response differences, and they are where the fused reward separates most clearly from single-signal RMs. The Hard and Code results establish the core reward-quality advantage of RALA: heterogeneous fusion is most valuable precisely when preference judgments are difficult.

5.3 Comparison with Baselines

RALA improves MATH-500 from 67.4% to 72.8% over the strongest evaluated aligned baseline, EndoRM. Table 2 also reports 22.0% on AIME 2024, a gain of 1.6 percentage points over EndoRM. The MATH-500 gain corresponds to an 8.0% relative improvement. On code generation, RALA lifts the SFT model to 82.9% on HumanEval and 75.3% on MBPP, showing that the fused reward improves code correctness while delivering the main gains on mathematical reasoning. The pattern is consistent: RALA is strongest where reward signals must resolve difficult correctness and reasoning distinctions. It sets the best aligned-model results on MATH-500 and AIME 2024, while improving code generation over SFT. Appendix F provides representative generated responses.

5.4 Ablation Study

The component ablation compares reward fusion with individual signals and two-component mixtures. Complete RALA has the highest mean on all four benchmarks, as shown in Table 3. Complete RALA achieves the best mean on all four benchmarks. This establishes that discriminative preference discrimination, generative verification, and endogenous confidence are complementary rather than interchangeable. These comparisons support the complementary value of combining policy-specific discrimination, generative verification, and intrinsic confidence.

The representation-level processing ablation yields 79.76% on HumanEval and 70.76% on MBPP, compared with RALA's 82.93% and 75.28%, highlighting the contribution within the proposed framework. The removal ablations show distinct but complementary roles: the generative component is critical for mathematical verification, the discriminative component for competition-level reasoning, and the endogenous component for code stability. Removing any one of them breaks the full-fusion advantage. Appendix C.3.2 complements these task-level results with reward and weight trajectories.

5.5 Evaluation of Fusion Weights

The fusion-weight ablation tests the value of updating the mixture during policy optimization. Table 4 compares equal weights, with online inverse-dispersion weighting from Eq. (8) and a frozen converged-weight baseline. Each task's coefficients are shared across all five seeds and held fixed throughout RLHF training, as detailed in Appendix C.2.4.

Adaptive weighting achieves the highest mean on all four benchmarks. Adaptive inverse-dispersion weighting achieves the best mean on all four benchmarks. It outperforms both equal weights and frozen converged weights. The advantage of online adaptation shows that tracking evolving rollout

Table 3: Performance of RALA with each component. The top-2 results in each column are marked in **bold** and underline.

	Code generation		Math reasoning	
	HumanEval	MBPP	MATH-500	AIME 2024
w/o Representation-level processing	79.76 ± 1.17	70.76 ± 1.50	70.56 ± 0.55	18.67 ± 3.80
w/o Generative RM (RLAIF)	82.80 ± 1.69	74.44 ± 2.03	70.64 ± 0.26	21.33 ± 1.83
w/o Discriminative RM	80.00 ± 0.51	74.24 ± 1.24	71.08 ± 0.48	14.67 ± 2.98
w/o Endogenous RM	80.12 ± 1.40	73.08 ± 1.22	71.16 ± 0.43	20.00 ± 2.36
only Generative RM (RLAIF)	75.73 ± 1.58	68.12 ± 1.78	71.00 ± 0.32	15.33 ± 2.98
only Discriminative RM	76.10 ± 0.80	64.72 ± 1.76	71.16 ± 0.26	14.67 ± 1.83
only Endogenous RM	74.63 ± 1.11	64.60 ± 0.93	70.68 ± 0.44	20.67 ± 1.49
RALA (Proposed)	$\mathbf{82.93 \pm 0.96}$	$\mathbf{75.28 \pm 1.55}$	$\mathbf{72.80 \pm 0.14}$	$\mathbf{22.00 \pm 1.83}$

Table 4: Performance of RALA with different weights. The top-2 results in each column are marked in **bold** and underline.

	Code generation		Math reasoning	
	HumanEval	MBPP	MATH-500	AIME 2024
Adaptive inverse-dispersion	$\mathbf{82.93 \pm 0.96}$	$\mathbf{75.28 \pm 1.55}$	$\mathbf{72.80 \pm 0.14}$	$\mathbf{22.00 \pm 1.83}$
Equal weights	76.34 ± 1.17	64.04 ± 0.36	70.88 ± 0.36	20.67 ± 2.79
Frozen converged weights	80.49 ± 1.22	73.88 ± 1.00	71.24 ± 0.09	18.67 ± 1.83

dispersion is a source of gain, not merely a terminal calibration. The frozen mixture improves over equal weights on both code benchmarks and MATH-500, while online adaptation delivers a further gain across all four tasks. Thus, the benefit extends beyond selecting a favorable terminal mixture: adapting the coefficients to evolving rollout dispersion provides additional empirical value.

5.6 Controlled Study

The controlled study quantifies the sensitivity of RALA to additive reward perturbations. Under low-level reward perturbation in Table 5, RALA is essentially unchanged. Across larger perturbations, the fused reward remains stable and preserves the ranking of RALA's design choices. This confirms that inverse-dispersion fusion provides practical robustness to numerical reward corruption.

Table 5: Sensitivity of RALA to additive Gaussian reward perturbations. The top-2 results in each column are marked in **bold** and underline.

	Code generation		Math reasoning	
	HumanEval	MBPP	MATH-500	AIME 2024
Noiseless	$\mathbf{82.93 \pm 0.96}$	$\mathbf{75.28 \pm 1.55}$	$\mathbf{72.80 \pm 0.14}$	$\mathbf{22.00 \pm 1.83}$
$\mathcal{N}(0, 0.001)$	82.32 ± 0.61	74.16 ± 1.35	72.72 ± 0.11	21.33 ± 1.83
$\mathcal{N}(0, 0.01)$	77.32 ± 1.17	71.76 ± 1.51	71.28 ± 0.23	18.67 ± 1.83
$\mathcal{N}(0, 0.05)$	74.02 ± 1.47	70.20 ± 1.81	70.84 ± 0.30	16.00 ± 2.79
$\mathcal{N}(0, 0.1)$	72.56 ± 1.49	67.44 ± 1.90	70.28 ± 0.54	15.33 ± 3.80

6 Conclusion

RALA brings policy-specific discrimination, external verification, and intrinsic confidence into a shared latent-reward framework. Its in-place reward head reuses the SFT backbone, and inverse-dispersion weighting balances the three reward streams during policy optimization. The analysis connects component reward errors to policy regret and formalizes the sample–reliability trade-off under explicit learning assumptions. Empirically, RALA achieves 62.0% on RM-Bench Hard and 72.8% on MATH-500, with relative gains of 7.3% and 8.0% over the strongest corresponding evaluated baselines. Together, these results establish a concrete route from complementary reward signals to stronger difficult-case discrimination and mathematical reasoning. RALA currently targets response-level candidate sets with offline reliability estimates; future work will extend it to online reliability estimation and broader multimodal feedback.

AI USE STATEMENT

In preparing this paper, large language models (LLMs) were employed solely as assistive tools for minor language refinement and for literature retrieval and discovery (e.g., identifying related work). All technical contributions, results, and conclusions are entirely the work of the authors.

REFERENCES

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966, 2024.

American Mathematics Competitions. American mathematics competitions (amc), 2024. URL `https://maa.org/student-programs/amc/`.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

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

Shreyas Chaudhari, Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, Ameet Deshpande, and Bruno Castro da Silva. Rlhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *ACM Computing Surveys*, 58 (2):1–37, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, et al. Evaluating large language models trained on code, 2021.

Sanjiban Choudhury. Process reward models for llm agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Kevin Duh, Helena Gomez, and Steven Bethard. Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024.

Sujan Dutta, Sayantan Mahinder, Raviteja Anantha, and Bortik Bandyopadhyay. Applying rlaif for code generation with api-usage in lightweight llms. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ ACL 2024)*, pp. 39–45, 2024.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Jack Gallifant, Majid Afshar, Saleem Ameen, Yindalon Aphinyanaphongs, Shan Chen, Giovanni Cacciamani, Dina Demner-Fushman, Dmitriy Dligach, Roxana Daneshjou, Chrystinne Fernandes, et al. The tripod-llm reporting guideline for studies using large language models. *Nature medicine*, 31(1):60–69, 2025.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Kuno Kim, Shivam Garg, Kirankumar Shiragur, and Stefano Ermon. Reward identification in inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 5496–5505. PMLR, 2021.

Yara Kyrychenko, Ke Zhou, Edyta Bogucka, and Daniele Quercia. C3ai: Crafting and evaluating constitutions for constitutional ai. In *Proceedings of the ACM on Web Conference 2025*, pp. 3204–3218, 2025.

Guoliang Li, Xuanhe Zhou, and Xinyang Zhao. Llm for data management. *Proceedings of the VLDB Endowment*, 17(12):4213–4216, 2024.

Lei Li, Yuancheng Wei, Zhihui Xie, Xuqing Yang, Yifan Song, Peiyi Wang, Chenxin An, Tianyu Liu, Sujian Li, Bill Yuchen Lin, et al. Vl-rewardbench: A challenging benchmark for vision-language generative reward models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 24657–24668, 2025a.

Yi-Chen Li, Tian Xu, Yang Yu, Xuqin Zhang, Xiong-Hui Chen, Zhongxiang Ling, Ningjing Chao, Lei Yuan, and Zhi-Hua Zhou. Generalist reward models: Found inside large language models. *arXiv preprint arXiv:2506.23235*, 2025b.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.

Shunyu Liu, Wenkai Fang, Zetian Hu, Junjie Zhang, Yang Zhou, Kongcheng Zhang, Rongcheng Tu, Ting-En Lin, Fei Huang, Mingli Song, et al. A survey of direct preference optimization. *arXiv preprint arXiv:2503.11701*, 2025a.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. RM-bench: Benchmarking reward models of language models with subtlety and style. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Xingzhou Lou, Dong Yan, Wei Shen, Yuzi Yan, Jian Xie, and Junge Zhang. Uncertainty-aware reward model: Teaching reward models to know what is unknown. *arXiv preprint arXiv:2410.00847*, 2024.

Yuanhang Luo, Yeheng Ge, Ruijian Han, and Guohao Shen. Learning guarantee of reward modeling using deep neural networks. *arXiv preprint arXiv:2505.06601*, 2025.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*, 2024.

Xingjun Ma, Yifeng Gao, Yixu Wang, Ruofan Wang, Xin Wang, Ye Sun, Yifan Ding, Hengyuan Xu, Yunhao Chen, Yunhan Zhao, et al. Safety at scale: A comprehensive survey of large model safety. *arXiv preprint arXiv:2502.05206*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Shauli Ravfogel, Yoav Goldberg, and Jacob Goldberger. Conformal nucleus sampling. In *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pp. 27–34. Association for Computational Linguistics (ACL), 2023.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Xiang-Bo Mao, Sitaram Asur, et al. A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more. *arXiv preprint arXiv:2407.16216*, 2024.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*, 3, 2023.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason E Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11548–11565, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Tianyu Yu, Haoye Zhang, Qiming Li, Qixin Xu, Yuan Yao, Da Chen, Xiaoman Lu, Ganqu Cui, Yunkai Dang, Taiwen He, et al. Rlaif-v: Open-source ai feedback leads to super gpt-4v trustworthiness. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19985–19995, 2025.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024.

Chuheng Zhang, Wei Shen, Li Zhao, Xuyun Zhang, Xiaolong Xu, Wanchun Dou, and Jiang Bian. Policy filtration for RLHF to mitigate noise in reward models. In *Forty-second International Conference on Machine Learning*, 2025a.

Hanning Zhang, Pengcheng Wang, Shizhe Diao, Yong Lin, Rui Pan, Hanze Dong, Dylan Zhang, Pavlo Molchanov, and Tong Zhang. Entropy-regularized process reward model. *Transactions on Machine Learning Research*, 2025b. ISSN 2835-8856.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations*, 2025c.

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

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.

Qinqing Zheng, Mikael Henaff, Amy Zhang, Aditya Grover, and Brandon Amos. Online intrinsic rewards for decision making agents from large language model feedback. In *Reinforcement Learning Conference*, 2025.

Hao Zhou, Chengming Hu, Ye Yuan, Yufei Cui, Yili Jin, Can Chen, Haolun Wu, Dun Yuan, Li Jiang, Di Wu, et al. Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities. *IEEE Communications Surveys & Tutorials*, 27(3): 1955–2005, 2024.

A Related Works

RLHF. RLHF is the standard paradigm for aligning LLMs with human intent (Christiano et al., 2017; Ouyang et al., 2022; Chaudhari et al., 2025). Early studies demonstrate the effectiveness of RLHF in improving instruction following (Ouyang et al., 2022; Touvron et al., 2023) and text summarization (Stiennon et al., 2020). Recently, direct preference optimization methods, such as DPO (Rafailov et al., 2023) and its extensions, including IPO (Azar et al., 2024) and KTO (Ethayarajh et al., 2024), have been proposed as more stable and computationally efficient alternatives by implicitly optimizing the reward objective without explicitly training a separate RM. These direct alignment objectives can be applied to preferences from human, AI, or mixed sources; their supervision cost depends on how those data are obtained. They do not, by themselves, specify how heterogeneous reward estimators should be calibrated and fused, which is the focus of RALA.

RLAIF. RLAIF has recently emerged as a scalable alternative to human annotation (Zheng et al., 2023; Agarwal et al., 2024; Yu et al., 2025). These methods leverage strong off-the-shelf LLMs or the policy model itself to generate preferences, critiques, or corrective signals. For example, Constitutional AI (Kyrychenko et al., 2025) uses predefined principles to guide automated feedback. Self-rewarding frameworks (Yuan et al., 2024; Wu et al., 2025) utilize the model as their own judge during training. Despite their scalability advantages, RLAIF methods are vulnerable to systematic uncertainty, such as self-delusion, bias amplification, or error compounding. Existing works also lack principled formulations for modeling feedback reliability under heterogeneous supervision sources.

Reward Models. RMs trained with Bradley-Terry (BT)-style pairwise ranking objectives are widely adopted. Yet, such formulations are sensitive to feedback noise and distribution shift, and often exhibit limited generalization across tasks and preference domains (Gao et al., 2023; Chaudhari et al., 2025). To capture reasoning processes, Process Reward Models (PRMs) assign feedback signals to intermediate reasoning steps. Their step-level supervision can come from human annotations, model judgments, or computationally intensive automatic construction, each with different cost and reliability trade-offs (Zhang et al., 2025d;b; Choudhury, 2025). Recently, generative RMs (Li et al., 2025a; Zhang et al., 2025c) have been proposed to exploit the generative and reasoning capabilities of LLMs by producing natural language critiques or correctness tokens. Endogenous signals such as token-level probabilities, entropy, and perplexity have also been explored as annotation-free proxies for model uncertainty and confidence estimation (Zheng et al., 2025; Li et al., 2025b). Nevertheless, existing reward learning methods optimize isolated supervision channels independently, without modeling alignment as a latent reward estimation problem under heterogeneous uncertainty.

RALA provides a unified perspective on these otherwise separately studied reward signals. It models shared-backbone discriminative scores, generative verification feedback, and endogenous policy-derived scores as heterogeneous observations of a common sequence-level reward, integrates their component errors through a shared fusion rule, and connects the resulting reward error to downstream policy regret.

B Detailed Theoretical Analysis and Proofs

In this appendix, we provide detailed mathematical proofs for the theoretical guarantees of the proposed RALA. We explicitly interpret reward learning as latent alignment reward estimation under heterogeneous supervision, and show how reward estimation error propagates to policy suboptimality.

B.1 Proof of Theorem 4.2

Joint statistical event. Throughout the statistical statements, fix $\eta \in (0, 1)$. Each component-learning assumption holds on an event of probability at least $1 - \eta/3$, including calibration and

preprocessing conditions. A union bound gives a joint event of probability at least $1-\eta$; $\tilde{\mathcal{O}}$ suppresses logarithmic sample-size and confidence factors. For data-dependent preprocessing, these guarantees are assumed uniform over the admissible bounded scales and preceding statistics.

Reward coordinates and identifiability. We interpret reward learning as estimating a latent alignment reward from heterogeneous projected observations. A central challenge in analyzing reward learning lies in the inherent ambiguity of the reward signal, commonly referred to as *reward identifiability* (Kim et al., 2021). In BT-style preference models, the induced pairwise preference distribution is invariant under prompt-dependent additive shifts of the reward. Specifically, for any function $\Phi : \mathcal{X} \to \mathbb{R}$, the transformed reward $r'(x, y) = r(x, y) + \Phi(x)$ induces the same pairwise preference probabilities between candidate responses to prompt x.

This invariance poses a theoretical difficulty: even if a learned reward model $\hat{r}$ perfectly captures the true preference structure, the distance $\|\hat{r} - r^*\|$ may be arbitrarily large due to an unidentifiable prompt-dependent additive shift. As a result, direct norm-based comparisons require a convention for the unidentified prompt-dependent shift. Fixed-temperature BT probabilities are not invariant to arbitrary rescaling; agreement of the numerical scales of heterogeneous reward components requires a separate common-target assumption. This ambiguity complicates finite-sample error analysis of reward estimation in prior work (e.g., Li et al. (2025b)). To address this issue, we adopt a prompt-wise centering convention.

Prompt-wise centering. For each fixed candidate set $\mathcal{Y}(x)$, define

$$(\mathcal{C}f)(x, y) = f(x, y) - \frac{1}{|\mathcal{Y}(x)|} \sum_{y' \in \mathcal{Y}(x)} f(x, y').$$

For any response policy π,

$$J(\pi; \mathcal{C}f) = J(\pi; f) - \mathbb{E}_{x\sim\rho}\left[\frac{1}{|\mathcal{Y}(x)|} \sum_{y' \in \mathcal{Y}(x)} f(x, y')\right].$$

The subtracted term is independent of π, so centering preserves response rankings and exact reward-maximizing policies. It is an analytical convention and requires no additional implementation step. By linearity, $\mathcal{C}\hat{r} = \sum_i \alpha_i \mathcal{C}\tilde{r}_i$ for the prompt-independent fusion coefficients considered here.

Centered-score convention. We retain $\hat{r}_i$ for the raw component scores and $\tilde{r}_i$ for their standardized versions in Section 3.2. Define

$$\bar{r}_i := \mathcal{C}\tilde{r}_i, \qquad \bar{r} := \mathcal{C}\hat{r} = \sum_i \alpha_i \bar{r}_i.$$

All learned-reward errors below use $\bar{r}_i$ and $\bar{r}$; raw and standardized scores retain their original meanings. The zero-sum identities $\sum_{y\in\mathcal{Y}(x)} \bar{r}_i(x, y) = \sum_{y\in\mathcal{Y}(x)} \bar{r}(x, y) = 0$ follow directly from centering. The implemented reward $\hat{r}$ and its centered representative $\bar{r}$ have the same exact optimizers and optimization gap.

Mixed-Norm Error Metrics. To quantify sequence-reward estimation error, we use a mixed norm over prompts and their candidate responses.

Definition B.1 (Mixed Norm (Luo et al., 2025))**.** Let $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be measurable. For a fixed prompt x, define

$$\|f(x, \cdot)\|_{\ell^p(\mathcal{Y}(x))} = \Big(\sum_{y\in\mathcal{Y}(x)} |f(x, y)|^p \Big)^{1/p}, \quad 1 \le p < \infty, \tag{11}$$

and $\|f(x, \cdot)\|_{\ell^\infty(\mathcal{Y}(x))} = \max_{y\in\mathcal{Y}(x)} |f(x, y)|$. We then define the mixed norm

$$\|f\|_{L^q(\mathcal{X},\ell^p)} = \left(\int_{\mathcal{X}} \|f(x, \cdot)\|^q_{\ell^p(\mathcal{Y}(x))} d\rho(x) \right)^{1/q}, \quad 1 \le q < \infty, \tag{12}$$

where ρ is the prompt distribution.

Throughout the proofs, $\|\cdot\|_{L^2}$ abbreviates this mixed norm with $p = q = 2$, and $\lesssim$ hides a positive constant independent of sample size.

Raw probabilities and analytical reward coordinates. Let $v(z)$ be the verification token associated with feedback z, and define

$$q^\star(x,y) = P^\star(v(z) = \text{Yes} \mid x, y), \qquad q_{\mathsf{n}}(x,y) = P_{\mathsf{n}}(v(z) = \text{Yes} \mid x, y).$$

These are raw probabilities in $[0,1]$. Their mixture is $q_{\mathsf{in}} = \bar{\gamma} q^\star + (1-\bar{\gamma}) q_{\mathsf{n}}$. Conditional on the preceding rollout statistics, put $a_{\mathsf{g}} = (\sigma_{\mathsf{g}} + \varepsilon)^{-1}$. Let $\hat{q}_{\mathsf{g}} = \hat{r}_{\mathsf{g}}$ denote the raw positive-verification probability from Section 3.2. We assume the generative orientation preserves the positive-verification direction and clipping is inactive on $[0,1]$, so its centered standardized score is $\bar{r}_{\mathsf{g}} = a_{\mathsf{g}} \mathcal{C} \hat{q}_{\mathsf{g}}$. Define the corresponding theoretical quantities by

$$r^* = a_{\mathsf{g}} \mathcal{C} q^\star, \qquad c_{\mathsf{n}} = a_{\mathsf{g}} \mathcal{C} q_{\mathsf{n}}, \qquad r_{\mathsf{in}} = a_{\mathsf{g}} \mathcal{C} q_{\mathsf{in}}.$$

Thus $r_{\mathsf{in}} = \bar{\gamma} r^* + (1-\bar{\gamma}) c_{\mathsf{n}}$. For any reward f, define $\mathrm{Rg}_f(\pi) = \max_{\pi' \in \Pi} J(\pi'; f) - J(\pi; f)$. Conditional on the fixed positive scale a_{g}, centering and scaling give

$$\mathrm{Rg}_{r^*}(\pi) = a_{\mathsf{g}} \, \mathrm{Rg}_{q^\star}(\pi).$$

Centering preserves regret numerically, while positive scaling preserves optimal policies and multiplies regret by a_{g}. The identity $\sum_{y \in \mathcal{Y}(x)} r^*(x,y) = 0$ follows from the definition of the centered target.

Assumption B.2. Let $p^{\mathsf{E}}(y \mid x)$ be the expert response distribution and $\hat{p}^{\mathsf{SFT}}(y \mid x)$ the fitted SFT response distribution, also denoted by $\hat{\pi}^{\mathsf{SFT}}$. Let T_y be the response length and put $a_{\mathsf{e}} = (\sigma_{\mathsf{e}} + \varepsilon)^{-1}$. For $\beta > 0$, let $\mathcal{H}^\beta([0,1]^d)$ denote a Hölder class of functions on a d-dimensional coordinate domain with a fixed smoothness-radius bound. Write $D_{\mathrm{KL}}(p^{\mathsf{E}} \| p)$ for the conditional KL divergence averaged over $x \sim \rho$. Below, $C > 0$ denotes an approximation constant independent of width and sample size. With fixed scale a_{e} and inactive endogenous clipping on the analyzed scores, define $f^* = a_{\mathsf{e}} \mathcal{C}[T_y^{-1} \log p^{\mathsf{E}}(y \mid x)]$. Assume $f^* = r^*$ and $f^* \in \mathcal{H}^{\beta_{\mathsf{e}}}([0,1]^{d_{\mathsf{e}}})$. Write $\mathcal{L}(p) = \mathbb{E}_{x \sim \rho, y \sim p^{\mathsf{E}}}[-\log p(y \mid x)]$. For a width-$W$ class $\mathcal{F}_W$ of normalized response distributions, assume

$$\inf_{p \in \mathcal{F}_W} \{\mathcal{L}(p) - \mathcal{L}(p^{\mathsf{E}})\} \le C W^{-2\beta_{\mathsf{e}}/d_{\mathsf{e}}}.$$

On the endogenous component event of probability at least $1 - \eta/3$, assume the fitted policy satisfies

$$\mathcal{L}(\hat{p}^{\mathsf{SFT}}) - \inf_{p \in \mathcal{F}_W} \mathcal{L}(p) \le \tilde{\mathcal{O}}(W / N_{\mathsf{SFT}}),$$

including optimization error, and lies in a neighborhood $\mathcal{U}$ of p^{E}. All risks are finite. Assume there exists a constant $\mu > 0$, independent of p and sample size, such that for every $p \in \mathcal{U}$,

$$\| a_{\mathsf{e}} \mathcal{C}[T_y^{-1} \log p - T_y^{-1} \log p^{\mathsf{E}}] \|^2_{L^2(\mathcal{X}, \ell^2(\mathcal{Y}))} \le \frac{1}{\mu} D_{\mathrm{KL}}(p^{\mathsf{E}} \| p).$$

The equality $f^* = r^*$ is the common-target realizability assumption. Together, the likelihood approximation, estimation, and fitted-policy localization conditions specify the endogenous component-learning guarantee.

Assumption B.3. Let $\mathcal{R}_k(g)$ be the population pairwise logistic risk with margin m_k under the training-pair distribution at step k, and let $\mathcal{G}_\theta$ be the scalar-head class on the SFT representation. Denote its fitted head by $\hat{g} \in \mathcal{G}_\theta$. Assume a raw-score comparator $g_k^\dagger$ minimizes this risk over a reference class containing $\mathcal{G}_\theta$, and its centered standardized representative equals r^*. The reference target admits a β_{d}-Hölder representation-level parameterization of intrinsic dimension d_{d}. On the discriminative component event, assume

$$\inf_{g \in \mathcal{G}_\theta} \mathcal{R}_k(g) - \mathcal{R}_k(g_k^\dagger) \le \tilde{\mathcal{O}}\left(N_{\mathsf{SFT}}^{-2\beta_{\mathsf{d}}/(d_{\mathsf{d}} + 2\beta_{\mathsf{d}})} \right),$$

$$\mathcal{R}_k(\hat{g}) - \inf_{g \in \mathcal{G}_\theta} \mathcal{R}_k(g) \le \tilde{\mathcal{O}}(N_{\mathsf{d}}^{-1/2}).$$

The second inequality includes optimization error. Let $K_{\mathsf{d}} > 0$ be a calibration constant independent of sample size and uniform over the analyzed steps and admissible preprocessing. Assume also that the fitted head lies in a calibration neighborhood in which

$$\| \bar{r}_{\mathsf{d}} - r^* \|^2_{L^2} \le K_{\mathsf{d}} \big[\mathcal{R}_k(\hat{g}) - \mathcal{R}_k(g_k^\dagger) \big],$$

The comparator and local calibration assumptions connect the margin-dependent pairwise risk to the common reward target.

The discriminative approximation, estimation, and fitted-head calibration conditions hold jointly with probability at least $1 - \eta/3$, uniformly over the analyzed curriculum and refreshed sampling distributions. They separate the representation contribution inherited from SFT from the head contribution learned through preference pairs.

Assumption B.4. Let $\mathcal{Q}_W$ be the width-W class of verification-token distributions and $\hat{Q}_\phi \in \mathcal{Q}_W$ its fitted member, parameterized by ϕ. Write $\mathcal{L}(Q)$ for population verification-token log loss and $\mathcal{L}^\star$ for its Bayes value; their explicit formulas are given in Appendix B.4.3. Let $\nu(dx, dy) = \rho(dx)\nu(y \mid x)$ be the distribution of verification-training inputs, with $\nu(y \mid x) \geq \kappa > 0$ on every analyzed candidate set. Let $Q_{\mathsf{in}}(\cdot \mid x, y)$ denote the conditional distribution of verification labels. For the width-W verifier class, assume that its best population token log loss exceeds the Bayes log loss by at most $CW^{-2\beta_{\mathsf{g}}/d_{\mathsf{g}}}$, and that the fitted verifier has excess log loss over this best-in-class value at most $\tilde{\mathcal{O}}(W/N_{\mathsf{g}})$ with high probability. The latter includes optimization error. All risks are finite. These conditions specify the approximation and estimation guarantees for verification-token learning.

The generative estimation condition is required with probability at least $1 - \eta/3$. These conditions specify a common latent-reward target for the conditional non-asymptotic analysis. Candidate coverage implies $|\mathcal{Y}(x)|\kappa \leq 1$ and determines the statistical conversion factor $1/\kappa$. The guarantee applies to these covered candidate sets under the prompt distribution ρ.

Next, the general finite-sample regret guarantee stated in Theorem 4.2 is proved in three steps:

- **Step 1: General regret reduction.** We establish a deterministic upper bound that reduces response-policy regret to the mixed-norm sequence-reward error $\|\bar{r} - r^*\|_{L^2(\mathcal{X},\ell^2)}$.
- **Step 2: Component error guarantees.** We transfer the stipulated learning rates of the endogenous, discriminative, and generative components to centered reward-error guarantees.
- **Step 3: Main regret bound for RALA.** We combine the general regret reduction with the conditional component estimation rates. The resulting bound contains a weighted contribution from each component and a reliability-dependent bias term from generative supervision.

Step 1: General regret reduction. We first show that the performance gap of the learned policy is controlled by the reward estimation error. Recall the regret definition

$$\mathrm{Rg}(\hat{\pi}) = J(\pi^*; r^*) - J(\hat{\pi}; r^*).$$

By adding and subtracting terms involving $\bar{r}$, it follows that

$$\mathrm{Rg}(\hat{\pi}) = \underbrace{J(\pi^*; r^*) - J(\pi^*; \bar{r})}_{\text{(i) evaluation error on } \pi^*} + \underbrace{J(\pi^*; \bar{r}) - J(\hat{\pi}; \bar{r})}_{\text{(ii) optimization gap}} + \underbrace{J(\hat{\pi}; \bar{r}) - J(\hat{\pi}; r^*)}_{\text{(iii) evaluation error on } \hat{\pi}}. \tag{13}$$

We first consider exact policy optimization, where $\hat{\pi} \in \arg\max_{\pi \in \Pi} J(\pi; \bar{r})$. Then,

$$J(\hat{\pi}; \bar{r}) \geq J(\pi^*; \bar{r}),$$

and the optimization gap term (ii) in Eq. (13) is non-positive. Define $e(x, y) = \bar{r}(x, y) - r^*(x, y)$. For any response policy π,

$$|J(\pi; \bar{r}) - J(\pi; r^*)| \leq \mathbb{E}_{x \sim \rho} \sum_{y \in \mathcal{Y}(x)} \pi(y \mid x)|e(x, y)| \leq \mathbb{E}_{x \sim \rho}\|e(x, \cdot)\|_{\ell^2(\mathcal{Y}(x))} \leq \|e\|_{L^2(\mathcal{X},\ell^2)}.$$

Applying this inequality to terms (i) and (iii) gives

$$\mathrm{Rg}(\hat{\pi}) \leq 2\|\bar{r} - r^*\|_{L^2(\mathcal{X},\ell^2)}. \tag{14}$$

The response-level reduction controls regret directly through the sequence-reward norm. Define $\epsilon_{\mathrm{opt}} = \max_{\pi \in \Pi} J(\pi; \bar{r}) - J(\hat{\pi}; \bar{r})$. Adding this term extends the general bound to approximate optimization on the covered candidate sets. For a restricted policy class within Π, it captures both class restriction and incomplete optimization; the fast-rate result uses candidate-wise exact maximization.

Extension to target mismatch. The deterministic error transfer also allows component-specific centered targets r_i° on the same reward scale. Define $s_i = \|\bar{r}_i - r_i^\circ\|_{L^2}$ and $b_i = \|r_i^\circ - r^*\|_{L^2}$. The triangle inequality and convexity give

$$\|\bar{r} - r^*\|_{L^2} \le \sum_i \alpha_i (s_i + b_i), \tag{15}$$

$$\|\bar{r} - r^*\|_{L^2}^2 \le \sum_i \alpha_i (s_i + b_i)^2. \tag{16}$$

Consequently, the general regret bound becomes $2\sum_i \alpha_i(s_i + b_i) + \epsilon_{\mathrm{opt}}$. Under the margin and gap-ratio conditions, the exact optimizer additionally satisfies

$$\mathrm{Rg}(\hat{\pi}) \lesssim \left[\sum_i \alpha_i (s_i + b_i)^2\right]^{1/(2-\alpha)}.$$

Component-specific learning guarantees determine s_i, while b_i explicitly records target mismatch. The main theorems specialize to $r_{\mathsf{e}}^\circ = r_{\mathsf{d}}^\circ = r^*$ and $r_{\mathsf{g}}^\circ = r_{\mathsf{in}}$, giving $b_{\mathsf{e}} = b_{\mathsf{d}} = 0$ and $b_{\mathsf{g}} = (1-\bar{\gamma})\mathcal{B}_{\mathsf{noise}}$. This separates generative feedback bias from additional cross-component mismatch.

Step 2: Component error guarantees. For widths satisfying the same learning assumptions, the endogenous and generative squared estimation terms before balancing are respectively

$$\tilde{\mathcal{O}}\left(W_{\mathsf{e}}^{-2\beta_{\mathsf{e}}/d_{\mathsf{e}}} + \frac{W_{\mathsf{e}}}{N_{\mathsf{SFT}}}\right), \qquad \tilde{\mathcal{O}}\left(W_{\mathsf{g}}^{-2\beta_{\mathsf{g}}/d_{\mathsf{g}}} + \frac{W_{\mathsf{g}}}{N_{\mathsf{g}}}\right).$$

The balanced-width choice in Eq. (9) yields the displayed sample-size rates; fixed-width models retain the approximation and estimation terms separately. Lemmas B.5, B.6, and B.7 transfer the stipulated component-learning guarantees to mixed-norm reward error. Appendix B.4 gives the detailed conversions. The endogenous and discriminative lemmas use analytical centered reward coordinates, with their fixed scaling absorbed into the parameterization and common-target assumptions. The generative lemma explicitly converts raw probability estimation error into these coordinates. Bounded scaling and the stipulated running-statistic error rates are required to retain the displayed orders.

Lemma B.5 (Convergence of Endogenous RM)**.** *Let $\hat{\pi}^{\mathsf{SFT}}$ be the policy obtained by SFT phase on N_{SFT} expert demonstrations. The centered endogenous score is $\bar{r}_{\mathsf{e}} = a_{\mathsf{e}}\mathcal{C}[T_y^{-1}\log\hat{p}^{\mathsf{SFT}}(y \mid x)]$. The raw score $\hat{r}_{\mathsf{e}}$ retains its definition in Section 3.2. Under Assumption B.2, the target response log-density belongs to a β_{e}-Hölder function class of intrinsic dimension d_{e}. For $W \asymp N_{\mathsf{SFT}}^{d_{\mathsf{e}}/(d_{\mathsf{e}}+2\beta_{\mathsf{e}})}$, with probability at least $1-\eta/3$,*

$$\|\bar{r}_{\mathsf{e}} - r^*\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2 \le \tilde{\mathcal{O}}\left(N_{\mathsf{SFT}}^{-\frac{2\beta_{\mathsf{e}}}{d_{\mathsf{e}}+2\beta_{\mathsf{e}}}}\right). \tag{17}$$

Lemma B.6 (Convergence of Discriminative Latent RM)**.** *Consider the raw discriminative score $g(h)$ trained via the pairwise logistic loss Eq. (5) on N_{d} preference pairs, sharing a backbone representation $h = f_\theta(x,y)$ obtained after SFT with N_{SFT} samples. Write $\bar{r}_{\mathsf{d}}$ for its centered standardized representative, with the fixed preprocessing absorbed into the parameterization in Assumption B.3. Under that assumption, with probability at least $1-\eta/3$,*

$$\|\bar{r}_{\mathsf{d}} - r^*\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2 \le \tilde{\mathcal{O}}\left(N_{\mathsf{SFT}}^{-\frac{2\beta_{\mathsf{d}}}{d_{\mathsf{d}}+2\beta_{\mathsf{d}}}}\right) + \tilde{\mathcal{O}}\left(N_{\mathsf{d}}^{-1/2}\right). \tag{18}$$

Under the assumed risk decomposition, N_{SFT} controls representation approximation and N_{d} controls head estimation. Gradient isolation assigns representation learning to SFT and preference learning to the head; Assumption B.3 specifies their statistical contributions.

Lemma B.7 (Convergence of Generative Reward Model)**.** *Let $\hat{Q}_\phi$ be the fitted verification-token distribution and $\hat{q}_{\mathsf{g}}(x,y) = \hat{Q}_\phi(\textit{Yes} \mid x,y) \in [0,1]$. Define its analytical reward as $\bar{r}_{\mathsf{g}} = a_{\mathsf{g}}\mathcal{C}\hat{q}_{\mathsf{g}}$, with r^*, r_{in}, and c_{n} defined above. Under Assumption B.4, for N_{g} i.i.d. verification examples and $W \asymp N_{\mathsf{g}}^{d_{\mathsf{g}}/(d_{\mathsf{g}}+2\beta_{\mathsf{g}})}$, with probability at least $1-\eta/3$,*

$$\|\bar{r}_{\mathsf{g}} - r^*\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2 \le \tilde{\mathcal{O}}\left(N_{\mathsf{g}}^{-\frac{2\beta_{\mathsf{g}}}{d_{\mathsf{g}}+2\beta_{\mathsf{g}}}}\right) + 2(1-\bar{\gamma})^2\|r^* - c_{\mathsf{n}}\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2. \tag{19}$$

The hidden constant depends on the bounded standardization scale and the candidate-coverage constant κ. More generally, without imposing a particular log-loss rate, the estimation term is bounded by $a_{\mathsf{g}}^2\big(\mathcal{L}(\hat{Q}_\phi) - \mathcal{L}^\star\big)/\kappa$, where $\mathcal{L}$ is the population verification-token log loss defined in Appendix B.4.3.

Step 3: Main Generalization Bound for RALA. We now combine the component-wise reward estimation bounds with the regret reduction result in Step 1. Recall that the fused reward estimator is defined as

$$\bar{r} = \sum_{i \in \{\mathsf{d},\mathsf{g},\mathsf{e}\}} \alpha_i \bar{r}_i, \qquad \alpha_i = \frac{(\sigma_i^2 + \varepsilon)^{-1}}{\sum_j (\sigma_j^2 + \varepsilon)^{-1}}, \tag{20}$$

where $\bar{r}_i = \mathcal{C}\tilde{r}_i$ is the centered standardized component score, σ_i is measured from its pre-standardization stream, $\varepsilon > 0$, and $\alpha_i \geq 0$ with $\sum_i \alpha_i = 1$. The component lemmas bound these same analytical scores. Convexity propagates their error guarantees through the implemented fusion coefficients.

For component errors across heterogeneous reward estimators, define $\delta_i^2 \coloneqq \|\bar{r}_i - r^*\|^2_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}$. By the definition of the fused estimator,

$$\bar{r} - r^* = \sum_i \alpha_i (\bar{r}_i - r^*).$$

Since $L^2(\mathcal{X}, \ell^2(\mathcal{Y}))$ is a normed linear space, Minkowski's inequality yields

$$\|\bar{r} - r^*\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))} \leq \sum_i \alpha_i \|\bar{r}_i - r^*\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}.$$

Combining this inequality directly with the regret reduction yields the linear-weight bound

$$\mathrm{Rg}(\hat{\pi}) \leq 2 \sum_i \alpha_i \delta_i. \tag{21}$$

Substituting the square roots of the component squared-error bounds proves Theorem 4.2. For the fast-rate analysis, Cauchy–Schwarz additionally gives

$$\|\bar{r} - r^*\|^2_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))} \leq \sum_i \alpha_i \|\bar{r}_i - r^*\|^2_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))} = \sum_i \alpha_i \delta_i^2. \tag{22}$$

Substituting the component-wise bounds from Lemmas B.5, B.6, and B.7 into Eq. (22) gives the fused squared-error bound used in Theorem 4.4. The three component events hold simultaneously with probability at least $1 - \eta$ by a union bound. On their intersection, both fusion inequalities apply pointwise to any admissible data-dependent coefficients, including coefficients selected from the same data. This gives the stated confidence level without requiring independent component errors.

The analysis links each component's statistical error to policy regret through the fusion coefficients and retains the generative supervision's reliability-dependent bias. It covers convex fusion, while the fusion-weight ablation evaluates the empirical benefit of online adaptation.

B.2 Proof of Theorem 4.4

For each prompt x, let

$$y^*(x) \in \arg\max_{y \in \mathcal{Y}(x)} r^*(x, y), \qquad \hat{y}(x) \in \arg\max_{y \in \mathcal{Y}(x)} \bar{r}(x, y),$$

and define $E(x) = \max_{y \in \mathcal{Y}(x)} |\bar{r}(x, y) - r^*(x, y)|$. Write $\mathbb{I}\{\cdot\}$ for an event indicator. If a suboptimal candidate exists and $E(x) < \Delta(x)/2$, every optimal response outranks every suboptimal response under $\bar{r}$, hence $\hat{y}(x) \in \mathcal{Y}^*(x)$. If all candidates are optimal, the pointwise regret is zero. Under the bounded positive gap-ratio condition in Theorem 4.4, both cases give

$$r^*(x, y^*(x)) - r^*(x, \hat{y}(x)) \leq C_{\mathrm{gap}} \Delta(x)\, \mathbb{I}\{E(x) \geq \Delta(x)/2\}\,. \tag{23}$$

Fix $\delta > 0$. Prompts with $\Delta(x) = 0$ contribute zero. On $\{\Delta(x) > \delta,\ E(x) \geq \Delta(x)/2\}$,

$$\Delta(x) \leq \frac{4E(x)^2}{\Delta(x)} \leq \frac{4E(x)^2}{\delta}.$$

Splitting into $0 < \Delta(x) \leq \delta$ and $\Delta(x) > \delta$ and applying the margin condition gives

$$\mathrm{Rg}(\hat{\pi}) \leq C_{\mathrm{gap}} \left[C_0 \delta^{\frac{1}{1-\alpha}} + \frac{4}{\delta} \mathbb{E}_{x\sim\rho} E(x)^2 \right] \leq C_{\mathrm{gap}} \left[C_0 \delta^{\frac{1}{1-\alpha}} + \frac{4}{\delta} \|\bar{r} - r^*\|^2_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))} \right]. \quad (24)$$

Write $V = \|\bar{r} - r^*\|^2_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}$. For $V > 0$, choosing $\delta \asymp V^{(1-\alpha)/(2-\alpha)}$ balances the two terms; for $V = 0$, regret is zero directly. Hence

$$\mathrm{Rg}(\hat{\pi}) \leq \tilde{\mathcal{O}}\left(\|\bar{r} - r^*\|^{\frac{2}{2-\alpha}}_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))} \right). \quad (25)$$

Substituting Eq. (22) and the three component bounds on their simultaneous event proves Theorem 4.4 with probability at least $1 - \eta$. The pointwise argument also applies to any exact optimizer that randomizes over maximizers of $\bar{r}$, by averaging over its support. The proof operates on complete-response rewards and their optimality gaps.

B.3 Sample-Reliability Bound Comparison

Corollary B.8 (Sample-Reliability Bound Comparison)**.** *For $j \in \{\mathsf{H}, \mathsf{AI}\}$, indexing human and AI feedback respectively, let $N_j > 0$ be the feedback sample size, $A_j > 0$ an estimation prefactor, $H_j > 0$ a decay exponent, and $C_j \geq 0$ a sample-independent bias term. Denote the corresponding specified upper-bound expression by B_j. Set $B_j = A_j N_j^{-H_j} + C_j$. These specified power-law upper-bound envelopes follow the estimation-plus-bias structure of Theorem 4.2, with coefficients fixed for the comparison. Assume $C_{\mathsf{H}} < C_{\mathsf{AI}}$. If*

$$N_{\mathsf{H}} \geq \left(\frac{A_{\mathsf{H}}}{C_{\mathsf{AI}} - C_{\mathsf{H}}} \right)^{1/H_{\mathsf{H}}}, \quad (26)$$

then $B_{\mathsf{H}} \leq C_{\mathsf{AI}} < B_{\mathsf{AI}}$ for every finite N_{AI}. Below this human-sample threshold, $B_{\mathsf{AI}} < B_{\mathsf{H}}$ if and only if

$$N_{\mathsf{AI}} > \left(\frac{A_{\mathsf{AI}}}{A_{\mathsf{H}} N_{\mathsf{H}}^{-H_{\mathsf{H}}} + C_{\mathsf{H}} - C_{\mathsf{AI}}} \right)^{1/H_{\mathsf{AI}}}.$$

The thresholds characterize the ordering of these specified upper-bound envelopes rather than realized policy regret.

Proof. The generative contribution in the general bound of Theorem 4.2, before its fusion coefficient and multiplicative constants, has the order

$$N_{\mathsf{g}}^{-\frac{\beta_{\mathsf{g}}}{d_{\mathsf{g}}+2\beta_{\mathsf{g}}}} + (1 - \bar{\gamma})\mathcal{B}_{\mathsf{noise}}. \quad (27)$$

The additive decomposition follows the general reward-error bound. The fast-rate result applies its outer exponent to the complete squared-error expression.

The comparison assumes $C_{\mathsf{H}} < C_{\mathsf{AI}}$. Its exact thresholds concern the specified power-law envelopes; the theorem retains logarithmic sample-size factors through $\tilde{\mathcal{O}}$.

The strict inequality $B_{\mathsf{AI}} < B_{\mathsf{H}}$ is equivalent to

$$A_{\mathsf{H}} N_{\mathsf{H}}^{-H_{\mathsf{H}}} + C_{\mathsf{H}} - C_{\mathsf{AI}} > A_{\mathsf{AI}} N_{\mathsf{AI}}^{-H_{\mathsf{AI}}} > 0. \quad (28)$$

Its left side is positive if and only if

$$N_{\mathsf{H}} < \left(\frac{A_{\mathsf{H}}}{C_{\mathsf{AI}} - C_{\mathsf{H}}} \right)^{1/H_{\mathsf{H}}}. \quad (29)$$

Under this condition, solving Eq. (28) gives

$$N_{\mathsf{AI}} > \left(\frac{A_{\mathsf{AI}}}{A_{\mathsf{H}} N_{\mathsf{H}}^{-H_{\mathsf{H}}} + C_{\mathsf{H}} - C_{\mathsf{AI}}} \right)^{1/H_{\mathsf{AI}}}. \tag{30}$$

At or above the human-sample threshold, $B_{\mathsf{H}} \leq C_{\mathsf{AI}} < B_{\mathsf{AI}}$ for every finite N_{AI}. Together, these regimes express the sample–reliability trade-off: sample volume reduces the estimation term, while the assumed feedback-bias ordering determines the high-data envelope comparison.

B.4 Proof of Supporting Lemmas

B.4.1 Proof of Lemma B.5

The SFT objective minimizes the response negative log-likelihood,

$$\mathcal{L}(p) = \mathbb{E}_{x\sim\rho,\, y\sim p^{\mathsf{E}}(\cdot|x)}[-\log p(y \mid x)],$$

which is equivalent to minimizing $D_{\mathrm{KL}}(p^{\mathsf{E}} \| p)$, averaged over $x \sim \rho$. For the fitted policy in Assumption B.2, the population excess risk decomposes exactly as

$$\mathcal{L}(\hat{p}^{\mathsf{SFT}}) - \mathcal{L}(p^{\mathsf{E}}) = \big[\mathcal{L}(\hat{p}^{\mathsf{SFT}}) - \inf_{p\in\mathcal{F}_W} \mathcal{L}(p)\big] + \big[\inf_{p\in\mathcal{F}_W} \mathcal{L}(p) - \mathcal{L}(p^{\mathsf{E}})\big].$$

The likelihood approximation condition in Assumption B.2 bounds the second term by

$$\inf_{p\in\mathcal{F}_W} \big(\mathcal{L}(p) - \mathcal{L}(p^{\mathsf{E}})\big) \leq C W^{-\frac{2\beta_{\mathsf{e}}}{d_{\mathsf{e}}}},$$

where W denotes the network width.

On the endogenous component event, the assumed fitted-policy excess-risk condition bounds the first term, including optimization error, by

$$\tilde{\mathcal{O}}\left(\frac{W}{N_{\mathsf{SFT}}}\right).$$

Balancing approximation and estimation errors by choosing $W \asymp N_{\mathsf{SFT}}^{\frac{d_{\mathsf{e}}}{d_{\mathsf{e}}+2\beta_{\mathsf{e}}}}$ gives

$$D_{\mathrm{KL}}(p^{\mathsf{E}} \| \hat{p}^{\mathsf{SFT}}) \leq \tilde{\mathcal{O}}\left(N_{\mathsf{SFT}}^{-\frac{2\beta_{\mathsf{e}}}{d_{\mathsf{e}}+2\beta_{\mathsf{e}}}}\right).$$

On the same event, $\hat{p}^{\mathsf{SFT}} \in \mathcal{U}$, so the local calibration condition in Assumption B.2 applies and gives

$$\|\bar{r}_{\mathsf{e}} - r^*\|^2_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))} \lesssim D_{\mathrm{KL}}(p^{\mathsf{E}} \| \hat{p}^{\mathsf{SFT}}),$$

This transfers the stipulated likelihood rates to centered reward error and proves the lemma with probability at least $1 - \eta/3$.

B.4.2 Proof of Lemma B.6

The pairwise training objective uses raw score differences. At a fixed curriculum step k, write

$$\ell_k(g; x, y^+, y^-) = -\log \sigma\big(g(h^+) - g(h^-) - m_k\big).$$

Let $\mathcal{R}_k(g) = \mathbb{E}[\ell_k(g)]$ and let $g_k^\dagger$ be the reference-class risk minimizer in Assumption B.3, whose centered standardized representative is r^*. Define $\mathcal{E}_{\mathsf{d}} = \mathcal{R}_k(\hat{g}) - \mathcal{R}_k(g_k^\dagger) \geq 0$. The exact population-risk decomposition is

$$\mathcal{E}_{\mathsf{d}} = \left[\mathcal{R}_k(\hat{g}) - \inf_{g\in\mathcal{G}_\theta} \mathcal{R}_k(g)\right] + \left[\inf_{g\in\mathcal{G}_\theta} \mathcal{R}_k(g) - \mathcal{R}_k(g_k^\dagger)\right].$$

Assumption B.3 controls the fitted head's excess risk uniformly over the analyzed curriculum and refreshed sampling distributions.

The SFT objective learns the backbone representation, while the isolated reward objective learns the scalar head. Assumption B.3 quantifies their representation-approximation and head-estimation contributions separately.

Under the realizability and local calibration assumptions, the excess pairwise risk controls the squared L^2 error of the centered standardized representative, with bounded scaling absorbed into the constant:

$$\|\bar{r}_{\mathsf{d}} - r^*\|_{L^2}^2 \lesssim \mathcal{E}_{\mathsf{d}}.$$

The representation approximation contribution stipulated in Assumption B.3 is

$$\tilde{\mathcal{O}}\left(N_{\mathsf{SFT}}^{-\frac{2\beta_{\mathsf{d}}}{d_{\mathsf{d}}+2\beta_{\mathsf{d}}}}\right), \tag{31}$$

while the fitted-head excess risk contributes $\tilde{\mathcal{O}}(N_{\mathsf{d}}^{-1/2})$. Thus the decomposition gives

$$\mathcal{E}_{\mathsf{d}} \le \tilde{\mathcal{O}}\left(N_{\mathsf{SFT}}^{-2\beta_{\mathsf{d}}/(d_{\mathsf{d}}+2\beta_{\mathsf{d}})} + N_{\mathsf{d}}^{-1/2}\right).$$

Combining these assumed contributions with calibration proves the lemma on the discriminative component event, of probability at least $1 - \eta/3$. In this bound, the representation term depends on the SFT sample size, while the reward-pair sample size enters the head-estimation term.

B.4.3 Proof of Lemma B.7

Write $Q_{\mathsf{in}}(v \mid x, y)$ for the distribution of the verification token induced by the observed feedback. The population token log loss is

$$\mathcal{L}(Q) = \mathbb{E}_{(x,y)\sim\nu,\, v\sim Q_{\mathsf{in}}}[-\log Q(v \mid x, y)].$$

Its excess over the Bayes log loss $\mathcal{L}^\star$ equals

$$\mathcal{L}(Q) - \mathcal{L}^\star = \mathbb{E}_\nu D_{\mathrm{KL}}(Q_{\mathsf{in}}(\cdot \mid x, y) \,\|\, Q(\cdot \mid x, y)).$$

The identity uses the categorical verification-token distribution scored by the reward. For multi-token verification labels, the same transfer requires the corresponding sequence-log-loss guarantee; auxiliary losses require their own excess-risk control.

The probabilities of the event $v = \text{Yes}$ under these distributions are q_{in} and $\hat{q}_{\mathsf{g}}$. The event-probability difference is bounded by total variation, and Pinsker's inequality gives

$$\mathbb{E}_\nu|\hat{q}_{\mathsf{g}} - q_{\mathsf{in}}|^2 \le \tfrac{1}{2}\big(\mathcal{L}(\hat{Q}_\phi) - \mathcal{L}^\star\big). \tag{32}$$

By Assumption B.4, the approximation and estimation contributions obey

$$\inf_{Q\in\mathcal{Q}_W} \mathcal{L}(Q) - \mathcal{L}^\star \le CW^{-2\beta_{\mathsf{g}}/d_{\mathsf{g}}}, \tag{33}$$

$$\mathcal{L}(\hat{Q}_\phi) - \inf_{Q\in\mathcal{Q}_W} \mathcal{L}(Q) \le \tilde{\mathcal{O}}(W/N_{\mathsf{g}}). \tag{34}$$

The stipulated fitted-verifier guarantee supplies these rates; additional optimization or model-misspecification residuals enter additively.

The coverage condition $\nu(y \mid x) \ge \kappa$ implies, for any f,

$$\|f\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2 \le \kappa^{-1}\mathbb{E}_\nu|f(x,y)|^2.$$

Prompt-wise centering is an orthogonal projection in the candidate-wise Euclidean norm. Hence

$$\|\bar{r}_{\mathsf{g}} - r_{\mathsf{in}}\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2 \le a_{\mathsf{g}}^2\|\hat{q}_{\mathsf{g}} - q_{\mathsf{in}}\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2 \le \frac{a_{\mathsf{g}}^2}{2\kappa}\big(\mathcal{L}(\hat{Q}_\phi) - \mathcal{L}^\star\big). \tag{35}$$

Balancing Eqs. (33) and (34) with $W \asymp N_{\mathsf{g}}^{d_{\mathsf{g}}/(d_{\mathsf{g}}+2\beta_{\mathsf{g}})}$ therefore yields the stated estimation rate.

At the raw probability level, $q_{\mathsf{in}} = \bar{\gamma}q^\star + (1-\bar{\gamma})q_{\mathsf{n}}$. Applying the same linear map $a_{\mathsf{g}}\mathcal{C}$ to all three terms gives $r_{\mathsf{in}} = \bar{\gamma}r^* + (1-\bar{\gamma})c_{\mathsf{n}}$ and

$$\|r_{\mathsf{in}} - r^*\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2 = (1-\bar{\gamma})^2\|r^* - c_{\mathsf{n}}\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2. \tag{36}$$

Finally,

$$\|\bar{r}_{\mathsf{g}} - r^*\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2 \le 2\|\bar{r}_{\mathsf{g}} - r_{\mathsf{in}}\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2 \quad + 2\|r_{\mathsf{in}} - r^*\|_{L^2(\mathcal{X},\ell^2(\mathcal{Y}))}^2. \tag{37}$$

Substituting the estimation rate and Eq. (36) proves the lemma. Raw probabilities remain in $[0, 1]$ throughout the log-loss analysis; only their analytical reward representatives are centered.

C Experiment Details

C.1 Detailed Process

Training Data. For code generation, following recent practice (Zhang et al., 2025a), we combine 75k samples from Magicoder-OSS-Instruct (Wei et al., 2023) and 55k samples from Evol-CodeAlpaca-v1 (Luo et al., 2024), resulting in a total of 130k instruction-response pairs for SFT. For math reasoning, we adopt Qwen-2.5-7B-Math as the base model and conduct evaluation under a fixed SFT initialization to illustrate the extensibility of RALA.

For training the discriminative RM, we randomly sample 7,000 prompts from the 130,000 SFT dataset. For each prompt, we generate five negative responses using the SFT policy via nucleus sampling. For the generative RM, we adopt an AI-based feedback setting. After SFT training, we sample 7,000 prompts from the training set and generate five candidate responses per prompt using the trained SFT model. We select the response pair with the maximum margin distance and construct preference labels using Qwen3-30B-A3B as the judge model, following the RLAIF protocol. The AI preference labeling judge prompt and the generative RM verification prompt are provided in Appendix D.

Benchmarks. We evaluate RALA with NVIDIA RTX3090×6 GPUs and an Intel(R) Xeon(R) 2.4 GHz CPU on a set of standard benchmarks covering reward modeling, code generation, and math reasoning. For reward learning, we use RM-Bench (Liu et al., 2025b) to evaluate the quality of learned reward models. RM-Bench focuses on fine-grained preference discrimination by testing whether reward models can correctly distinguish subtle content differences between candidate responses. For code generation, we adopt HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). HumanEval contains 164 hand-crafted Python programming tasks, while our MBPP evaluation uses a 500-task test split of crowd-sourced Python programming problems.Both benchmarks evaluate functional correctness using unit test execution under a zero-shot generation setting. For math reasoning, we report results on MATH-500 (Hendrycks et al., 2021) and AIME 2024 (American Mathematics Competitions, 2024). MATH-500 comprises 500 problems randomly sampled from the original MATH test set, covering a broad range of difficulty levels. AIME 2024 includes 30 competition-level integer-answer problems, providing a challenging benchmark for math reasoning.

Evaluation metrics. For RM-Bench, we evaluate RMs by computing the pairwise preference accuracy under the same prompt. For HumanEval, MBPP, MATH-500, and AIME 2024, we report pass@1 using one generated response per problem.

C.2 Details of Hyper-parameters

We provide the complete hyperparameter settings for both the SFT stage and the RLHF stage.

C.2.1 SFT Stage Hyperparameters

Table 6 lists all hyperparameters used in the SFT stage with LoRA fine-tuning and the integrated discriminative branch trained within the SFT stage

C.2.2 Generative RM Hyperparameters

Table 7 details the hyperparameters used to train the generative RM. The verifier is trained as a LoRA adapter with a Yes/No verification objective, using AI-labeled preference pairs constructed from sampled candidate responses.

C.2.3 RLHF/RLAIF Stage Hyperparameters

Table 8 details the hyperparameters for the RLHF stage with PPO policy optimization and tri-stream reward fusion. We use task-specific RLHF/RLAIF hyperparameters for code generation and math reasoning, reflecting their different optimization targets and training/evaluation datasets. We repeat RLHF training with five random seeds (3407–3411).

Table 6: Hyperparameters for SFT Stage

Hyperparameter	Notation	Value
LoRA Configuration		
LoRA rank	r	64
LoRA alpha	α_{LoRA}	16
LoRA dropout	p_{drop}	0.05
Training Configuration		
Number of epochs	E	3
Per-device batch size	B_{SFT}	2
Gradient accumulation steps	N_{accum}	8
Learning rate	η	2×10^{-4}
LR scheduler	-	cosine
Warmup ratio	-	0.03
Validation split	-	0.02
Save/evaluation interval	-	200 steps
Optimizer	-	PagedAdamW 8-bit
Max sequence length	$L_{\max}$	2048
Random seed	-	3407
Dual-Head Discriminative Branch		
Discriminative prompts	N_{disc}	7,000
Negatives per prompt	K_{neg}	5
Negative sampling	-	temp. 1.0, top-p 0.90, 256 tokens
Curriculum margin	m_k	$m_0 = 0, \gamma = 0.1, K = 1000$

Table 7: Hyperparameters for Generative RM Training

Hyperparameter	Notation	Value
LoRA Configuration		
LoRA rank	r_{g}	32
LoRA alpha	α_{g}	64
LoRA dropout	p_{g}	0.1
Training Configuration		
Number of epochs	-	3
Effective batch size	-	16
Learning rate	-	2×10^{-4}
LR scheduler	-	cosine
Warmup ratio	-	0.1
Weight decay	-	0.0
Max sequence length	-	1024
Validation split	-	0.02
Rank loss weight	λ_{rank}	1.5
Random seed	-	42

C.2.4 Convergent Weights

For the convergent-weight baseline in Table 4, we compute inverse-variance weights from the EMA standard deviations stored in the final checkpoint of a reference adaptive run. The weights $(\alpha_{\text{d}}, \alpha_{\text{g}}, \alpha_{\text{e}}) \approx (0.3426, 0.3309, 0.3265)$ for Code and $(0.3888, 0.2608, 0.3504)$ for Math are shared across all five seeds and remain fixed throughout RLHF training.

C.3 Additional Results

Table 8: Hyperparameters for RLHF Stage

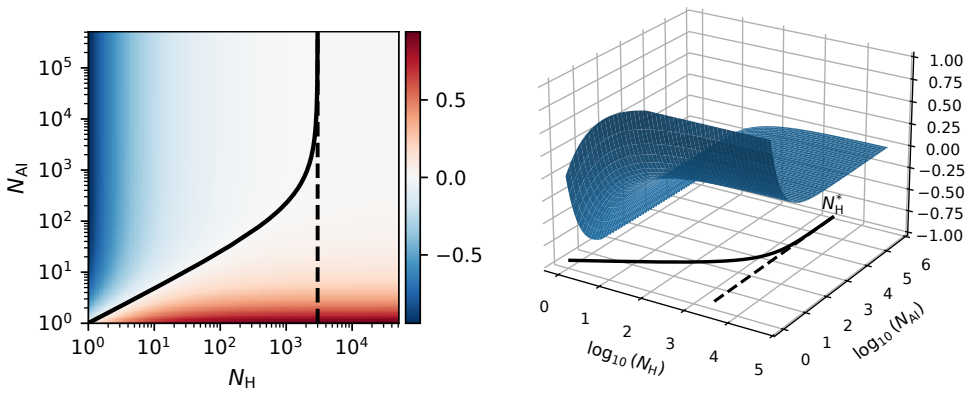
Hyperparameter	**Code generation**	**Math reasoning**
Model Configuration		
Base model family	DeepSeek-Coder 6.7B	Qwen2.5-7B-Math
Quantization	8-bit loading + LoRA	4-bit QLoRA
Max sequence length	1056	1568
Reward-model max length	1056	1568
Generation Configuration		
Max prompt tokens	768	1024
Max new tokens during training	256	512
Sampling temperature	0.55	0.70
Top-p sampling	0.85	0.90
Do sample	True	True
Optimization		
Optimizer	AdamW	AdamW
Learning rate	1.5×10^{-7}	1×10^{-7}
Weight decay	0.0	0.0
Max gradient norm	0.12	0.2
LoRA Configuration		
LoRA rank	16	16
LoRA alpha	32	32
LoRA dropout	0.0	0.0
PPO Policy		
Total updates	270	200
Effective rollout batch size	8	8
PPO epochs	1	1
Policy clip range	0.02	0.02
Value clip range	0.10	0.05
KL coefficient	0.12	0.02
KL token clip	3.0	2.0
Log-ratio clip	3.0	8.0
Discount factor	1.0	1.0
GAE lambda	0.95	0.95
Value loss coefficient	0.1	0.1
Entropy coefficient	0.0	0.0
Reward Fusion		
EMA beta	0.95	0.95
Fusion epsilon	1×10^{-6}	1×10^{-6}
Reward clip	10.0	10.0
Checkpoint/evaluation step	270	200
Random seeds	3407–3411	3407–3411

C.3.1 Numerical Results of Corollary B.8

Figure 2 illustrates Corollary B.8 with $A_{\mathsf{H}} = A_{\mathsf{AI}} = 1$, $C_{\mathsf{AI}} = 1.0082$, $C_{\mathsf{H}} = 1.0$, $H_{\mathsf{H}} = 0.6$, and $H_{\mathsf{AI}} = 0.9$. For these specified estimation-plus-bias envelopes, abundant AI feedback yields the tighter bound when human feedback is scarce. Beyond the human-sample threshold, the assumed smaller human-feedback bias determines the tighter envelope. The crossover illustrates the distinct roles of feedback quantity and reliability.

C.3.2 Ablation Study

Figures 3 and 4 examine fused-reward dispersion and adaptive-weight trajectories on Code and Math, complementing the task-level component ablation in Table 3. The top panels report the standard



(a) 2D diagram (b) 3D surface

Figure 2: Sample–reliability trade-off for specified RLHF and RLAIF upper-bound envelopes.

deviation (std) of the fused reward during PPO updates, while the bottom panels visualize the adaptive aggregation weights associated with the discriminative RM, endogenous RM, and generative RM.

On Code, RALA's fused-reward standard deviation is generally below those of the three reward-removal variants in Figure 3, highlighting the combined components' contribution to reward-scale stability in these runs. The reward-removal comparisons characterize each component's contribution to fused-reward dispersion. The representation-processing ablation evaluates a different design choice, whose downstream effect is reported in Table 3. The smooth weight trajectories show how the adaptive rule adjusts the mixture over the course of training.

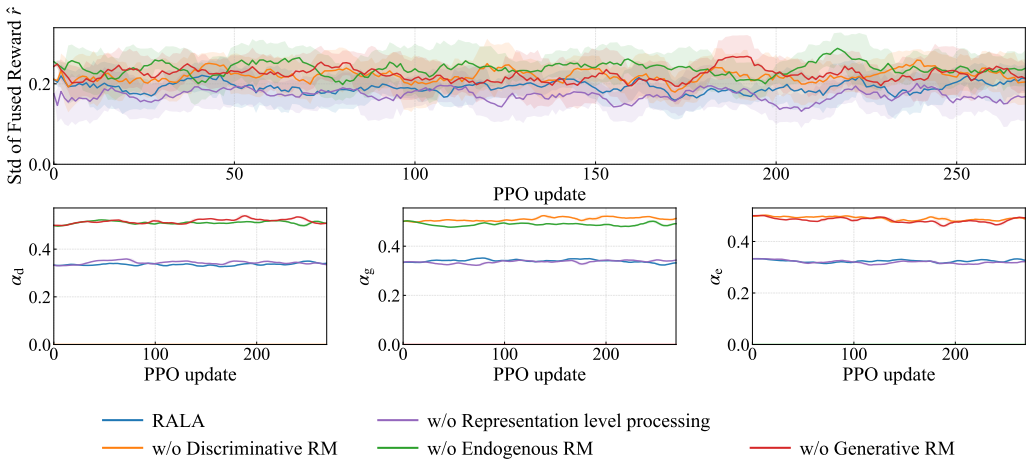


Figure 3: Training dynamics on the Code task. The top panel shows the std of the fused reward $\hat{r}$ during PPO updates. The bottom panels illustrate the adaptive results of weights α_d, α_g, and α_e.

On Math, Figure 4 shows a different pattern of component contributions to reward dispersion. The reward-removal variants generally exhibit higher fused-reward dispersion than complete RALA, highlighting the combined signals' contribution to reward-scale stability in these runs. The discriminative component also contributes to downstream reasoning: removing it reduces the mean AIME 2024 score from 22.00% to 14.67% in Table 3. The complete model combines stable fused-reward trajecto-

ries in these runs with complementary task-level gains. The trajectory plots and component ablations thus describe numerical behavior and downstream utility, respectively.

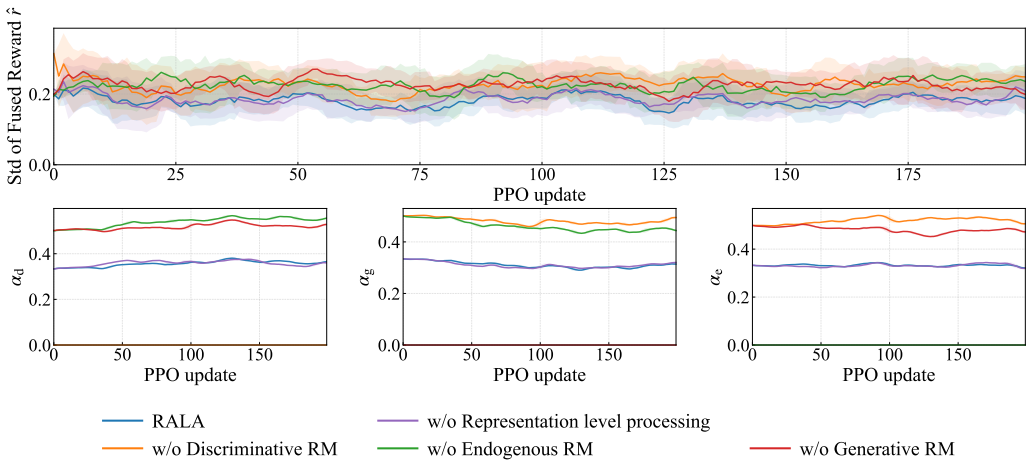


Figure 4: Training dynamics on the Math task. The top panel shows the std of the fused reward $\hat{r}$ during PPO updates. The bottom panels illustrate the adaptive results of weights $\alpha_{\mathtt{d}}$, $\alpha_{\mathtt{g}}$, and $\alpha_{\mathtt{e}}$.

D Prompt Templates

This appendix reports four prompt templates: (1) the generative RM in RALA, (2) the AI judge used to construct its training labels, (3) the Generative Verifier baseline, and (4) the LLM-as-a-Judge baseline.

D.1 Generative RM Used in RALA

Generative RM template

```
Question: {prompt}

Response: {response}

Based on the question and the response provided, is the response correct
and complete?
Answer with only “Yes” or “No”.

Answer:
```

D.2 AI Preference Labeling Judge for Generative RM Data Construction

AI preference labeling judge template

```
You are a careful evaluator. Given a TASK and two CANDIDATES, decide which candidate
better satisfies the task. Consider correctness, completeness, faithfulness to the
task requirements, clarity, and overall quality.
Reply with a single character: 1 if CANDIDATE 1 is better, otherwise 2.

TASK:
{prompt}

CANDIDATE 1:
{response_1}

CANDIDATE 2:
{response_2}

Answer:
```

D.3 Generative Verifier

Generative Verifier Prompt

```
You need to determine if the "Response" is a good response to the "Query". A "good"
response MUST satisfy the following conditions:
1. Relevance to Query: The answer must directly and comprehensively address the
user's query without any irrelevant information.
2. Factuality and Helpfulness: The information in the answer should be accurate
and useful.
Please carefully review the following materials.
Query:
{query}
Response:
{response}
Based on your evaluation, is the answer a good response? Answer with only "YES" or
"NO".
```

D.4 LLM-as-a-Judge

Prompt Template for the LLM-as-a-Judge

System message
Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below. You should choose the assistant that follows the user's instructions and answers the user's question better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses. Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

Question:
{question}
Assistant A:
{answer_a}
Assistant B:
{answer_b}

E GENERATED RESPONSES FOR TRAINING DISCRIMINATIVE RM

E.1 CODE TASK

Sample 1

Prompt

You are tasked with creating a program that determines whether a person is of legal drinking age based on their birth year. The program should take the birth year as input and output a message indicating whether the person is of legal drinking age or not.
Write a function ‘checkLegalDrinkingAge‘ that takes an integer ‘birthYear‘ as input and returns a string message. If the person is of legal drinking age (i.e., 21 years or older), the message should be "You are of legal drinking age". If the person is not of legal drinking age, the message should be "You are not of legal drinking age".
For example, if the input ‘birthYear‘ is 2000, the function should return "You are not of legal drinking age", and if the input ‘birthYear‘ is 1990, the function should return "You are of legal drinking age".

Positive sample

```
def checkLegalDrinkingAge(birthYear):
    legal_drinking_age = 21
    current_year = 2022  # Assuming the current year is 2022

    age = current_year - birthYear
    if age >= legal_drinking_age:
        return "You are of legal drinking age"
    else:
        return "You are not of legal drinking age"

# Test cases
print(checkLegalDrinkingAge(2000))  # Output: "You are not of legal drinking age"
print(checkLegalDrinkingAge(1990))  # Output: "You are of legal drinking age"
```

Negative sample

JavaScript snippet.

```
function checkLegalDrinkingAge(birthYear) {
    const currentYear = new Date().getFullYear();
    const legalAge = 21;
    const age = currentYear - birthYear;
    return age >= legalAge ? "You are of legal drinking age" : "You are not of
     legal drinking age";
}
```

This function first calculates the age of the person by subtracting the birth year from the current year. Then, it checks if this age is greater than or equal to the legal drinking age (21). If it is, it returns the message “You are of legal drinking age”. Otherwise, it returns the message “You are not of legal drinking age”.

Example usage.

```
console.log(checkLegalDrinkingAge(1990)); // Outputs: "You are of legal drinking
     age"
console.log(checkLegalDrinkingAge(2000)); // Outputs: "You are not of legal
     drinking age"
```

Sample 2

Prompt

Formulate a programmatic representation to execute the addition operation on two distinct linked list structures of varied lengths. list1 = [2->4->3] list2 = [5->6->4->1]

Positive sample

For simplicity, the linked list nodes contain a single digit.

Here is a Python solution using a LinkedList class and a Node class. This solution works by iterating through both linked lists at once and adding the corresponding values together. If one list is longer than the other, the remaining values are simply added to the result:

```
class Node:
    def __init__(self, value=0, next=None):
        self.value = value
        self.next = next

class LinkedList:
    def __init__(self, node=None):
        self.head = node

    def add(self, node):
        if not self.head:
            self.head = node
        else:
            curr = self.head
            while curr.next:
                curr = curr.next
            curr.next = node

def add_two_lists(l1, l2):
    dummy_head = Node(0)
    p = l1
    q = l2
    curr = dummy_head
    carry = 0

    while p is not None or q is not None:
        x = p.value if p is not None else 0
        y = q.value if q is not None else 0

        sum = carry + x + y
        carry = sum // 10  # For getting carry

        curr.next = Node(sum % 10)
        curr = curr.next

        if p is not None: p = p.next
        if q is not None: q = q.next

    if carry > 0:
        curr.next = Node(carry)

    return dummy_head.next

# Link List 1 : 2->4->3
l1 = LinkedList(Node(2))
l1.add(Node(4))
l1.add(Node(3))

# Link List 2 : 5->6->4->1
l2 = LinkedList(Node(5))
l2.add(Node(6))
l2.add(Node(4))
l2.add(Node(1))

result = add_two_lists(l1.head, l2.head)

while result:
    print(result.value, end='')
    result = result.next
    if result: print('->', end='')
```

This program creates two linked lists, adds corresponding elements together, handles carryover, and prints the result.

Negative sample

The result should be [7->0->7]

```
'''

# Definition for singly-linked list.
class ListNode:
    def __init__(self, x):
        self.val = x
        self.next = None

class Solution:
    def addTwoNumbers(self, l1: ListNode, l2: ListNode) -> ListNode:
        dummy = ListNode(0)
        current, carry = dummy, 0
        while l1 or l2 or carry:
            val = carry
            if l1:
                val += l1.val
                l1 = l1.next
            if l2:
                val += l2.val
                l2 = l2.next
            carry, val = divmod(val, 10)
            current.next = ListNode(val)
            current = current.next
        return dummy.next

# Helper function to create linked list from list
def create_linked_list(lst):
    head = ListNode(lst[0])
    current = head
    for i in lst[1:]:
        current.next = ListNode(i)
        current = current.next
    return head

# Helper function to print linked list
def print_linked_list(lst):
    while lst:
        print(lst.val, end=" ")
        lst = lst.next

# Test
l1 = create_linked_list([2, 4, 3])
l2 = create_linked_list([5, 6, 4, 1])
s = Solution()
result = s.addTwoNumbers(l1, l2)
print_linked_list(result)  # Output: 7 0 7

# This program creates two linked lists from input lists, adds them together using
    the addTwoNumbers function, and then prints the resultant linked list.

'''
```

The addTwoNumbers function works by creating a dummy node and a current pointer. It then iteratively adds the values from l1 and l2, along with any carry from the previous sum, and creates a new node for the result.

E.2 MATH TASK

Sample 1

Prompt

Bryan has some 3 cent stamps and some 4 cent stamps. What is the least number of stamps he can combine so the value of the stamps is 33 cents?

Positive sample

We can set up the equation $3x + 4y = 33$ where x represents the number of 3 cent stamps and y represents the number of 4 cent stamps. To find the minimum number of stamps, we want to minimize the total number of stamps $x + y$. We can start by subtracting $3x$ from both sides to get $4y = 33 - 3x$. Then, we can divide both sides by 4 to get $y = \frac{33-3x}{4}$. Since x and y must be whole numbers, we need to find a value of x that makes $\frac{33-3x}{4}$ a whole number. From here, we can test different values of x to see which ones give a whole number for y:

- For $x = 1$, we have $y = \frac{33-3(1)}{4} = \frac{30}{4} = 7.5$, which is not a whole number.

- For $x = 2$, we have $y = \frac{33-3(2)}{4} = \frac{27}{4} = 6.75$, which is not a whole number.

- For $x = 3$, we have $y = \frac{33-3(3)}{4} = \frac{24}{4} = 6$, which is a whole number.

Therefore, the minimum number of stamps Bryan can combine is $x + y = 3 + 6 = \boxed{9}$. 9

Negative sample

Let's use Python to solve this problem. We need to find non-negative integers x and y such that $3x+4y = 33$ and $x + y$ is minimized.

We can iterate over possible values of x and solve for y to find the solution. Here's the Python code to do that:

```
def find_min_stamps(target_value):
    min_stamps = float('inf')  # Initialize with a large number
    for x in range(target_value // 3 + 1):  # Iterate over possible values of x
        y = (target_value - 3 * x) / 4
        if y.is_integer() and y >= 0:  # Check if y is a non-negative integer
            min_stamps = min(min_stamps, x + int(y))
    return min_stamps if min_stamps != float('inf') else None

# Find the minimum number of stamps for a target value of 33 cents
min_stamps = find_min_stamps(33)
print(min_stamps)
```

```
Output: 9
```

The least number of stamps Bryan can combine to get a value of 33 cents is $\boxed{9}$. This can be achieved by using 3 stamps of 3 cents each and 6 stamps of 4 cents each.

Sample 2

Prompt

What integer n satisfies $0 \leq n < 101$ and

$$100n \equiv 72 \pmod{101} \; ?$$

Positive sample

We can rewrite the congruence as $100n - 72 \equiv 0 \pmod{101}$. This implies that $100n \equiv 72 \pmod{101}$. Since 100 is relatively prime to 101, we can divide both sides by 100 to get $n \equiv 72 \cdot 100^{-1} \pmod{101}$. To find the inverse of 100 modulo 101, we can use the Euclidean Algorithm. We have

$$\begin{aligned} 101 &= 1 \cdot 100 + 1 \\ 100 &= 1 \cdot 101 - 1 \end{aligned}$$

Therefore, $100^{-1} \equiv -1 \pmod{101}$. Substituting this back into our congruence, we have $n \equiv 72 \cdot (-1) \pmod{101}$. This simplifies to $n \equiv -72 \pmod{101}$. Since we want the smallest nonnegative solution, we

find that $n \equiv 101 - 72 \equiv 29 \pmod{101}$. Thus, the integer n that satisfies the given congruence is $\boxed{29}$.
29

Negative sample

To solve for the integer n that satisfies $0 \leq n < 101$ and $100n \equiv 72 \pmod{101}$, we need to find the multiplicative inverse of 100 modulo 101. The multiplicative inverse of 100 modulo 101 is an integer x such that:

$$100x \equiv 1 \pmod{101}$$

We can use the Extended Euclidean Algorithm to find this inverse. First, we apply the Euclidean Algorithm to find the greatest common divisor (gcd) of 100 and 101.

1. **Euclidean Algorithm:**

$$101 = 1 \cdot 100 + 1$$
$$100 = 100 \cdot 1 + 0$$

The gcd is 1, which means 100 and 101 are coprime, and an inverse exists.

2. **Extended Euclidean Algorithm:** We work backwards from the Euclidean Algorithm to express 1 as a combination of 100 and 101.

$$1 = 101 - 1 \cdot 100$$

From this equation, we can see that:

$$1 \equiv -1 \cdot 100 \pmod{101}$$

Therefore, the multiplicative inverse of 100 modulo 101 is -1. Since we need a positive integer x such that $0 \leq x < 101$, we convert -1 to a positive equivalent by adding 101:

$$-1 + 101 = 100$$

Thus, the multiplicative inverse of 100 modulo 101 is 100.

3. **Solving the original congruence:** We multiply both sides of the congruence $100n \equiv 72 \pmod{101}$ by the inverse of 100:

$$100 \cdot 100n \equiv 100 \cdot 72 \pmod{101}$$
$$n \equiv 72 \pmod{101}$$

Since $0 \leq 72 < 101$, the solution is:

$$n = 72$$

Therefore, the integer n that satisfies the given congruence is $\boxed{72}$.

F Generated Responses of Benchmarks

F.1 HumanEval

Sample 1 (Python/2)

Prompt

```
def truncate_number(number: float) -> float:
    """ Given a positive floating point number, it can be decomposed into
    and integer part (largest integer smaller than given number) and decimals
    (leftover part always smaller than 1).

    Return the decimal part of the number.
    >>> truncate_number(3.5)
    0.5
    """
```

Response

```
def truncate_number(number: float) -> float:
    return number - int(number)
```

Sample 2 (Python/4)

Prompt

```
from typing import List


def mean_absolute_deviation(numbers: List[float]) -> float:
    """ For a given list of input numbers, calculate Mean Absolute Deviation
    around the mean of this dataset.
    Mean Absolute Deviation is the average absolute difference between each
    element and a centerpoint (mean in this case):
    MAD = average | x - x_mean |
    >>> mean_absolute_deviation([1.0, 2.0, 3.0, 4.0])
    1.0
    """
```

Response

```
from typing import List

def mean_absolute_deviation(numbers: List[float]) -> float:
    mean = sum(numbers) / len(numbers)
    return sum(abs(x - mean) for x in numbers) / len(numbers)
```

F.2 MBPP

Sample 1 (Task 17)

Prompt

Write a function to find the perimeter of a square.

Response

```
def square_perimeter(side_length):
    return 4 * side_length
```

Sample 2 (Task 21)

Prompt

Write a function to find m number of multiples of n.

Response

```
def multiples_of_num(m, n):
    return [i * n for i in range(1, m+1)]
```

F.3 MATH-500

Sample 1 (Problem 79)

Prompt

For what value of x is $2^3 \cdot 3^x = 72$?

Response

To solve the equation $2^3 \cdot 3^x = 72$, we start by expressing 72 as a product of its prime factors. We know that:

$$72 = 2^3 \cdot 3^2$$

So, we can rewrite the original equation as:

$$2^3 \cdot 3^x = 2^3 \cdot 3^2$$

Since the bases are the same, we can equate the exponents of the corresponding bases:

$$3^x = 3^2$$

This implies that:

$$x = 2$$

Therefore, the value of x is $\boxed{2}$.

Sample 2 (Problem 161)

Prompt

Evaluate $\log_2 64$.

Response

To evaluate $\log_2 64$, we need to determine the power to which the base 2 must be raised to obtain 64. In other words, we need to find x such that:

$$2^x = 64$$

We know that 64 can be expressed as a power of 2. Let's break it down:

$$64 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^6$$

So, we have:

$$2^x = 2^6$$

Since the bases are the same, we can equate the exponents:

$$x = 6$$

Therefore, $\log_2 64 = 6$.
The final answer is:

$$\boxed{6}$$

F.4 AIME 2024

Sample 1

Prompt

There exist real numbers x and y, both greater than 1, such that $\log_x\left(y^x\right) = \log_y\left(x^{4y}\right) = 10$. Find xy.

Response

To solve the given equations $\log_x(y^x) = 10$ and $\log_y(x^{4y}) = 10$, we start by simplifying each logarithmic expression.
First, consider the equation $\log_x(y^x) = 10$. Using the logarithm power rule $\log_b(a^c) = c\log_b(a)$, we can rewrite this as:

$$\log_x(y^x) = x\log_x(y) = 10$$

This implies:

$$\log_x(y) = \frac{10}{x}$$

Next, consider the equation $\log_y(x^{4y}) = 10$. Again, using the logarithm power rule, we can rewrite this as:

$$\log_y(x^{4y}) = 4y\log_y(x) = 10$$

This implies:

$$\log_y(x) = \frac{10}{4y} = \frac{5}{2y}$$

Now we have two expressions for $\log_x(y)$ and $\log_y(x)$:

$$\log_x(y) = \frac{10}{x} \quad \text{and} \quad \log_y(x) = \frac{5}{2y}$$

We know that $\log_x(y)$ and $\log_y(x)$ are reciprocals of each other, i.e., $\log_x(y) = \frac{1}{\log_y(x)}$. Therefore, we can set up the equation:

$$\frac{10}{x} = \frac{1}{\frac{5}{2y}} = \frac{2y}{5}$$

Solving for x in terms of y, we get:

$$\frac{10}{x} = \frac{2y}{5} \implies 10 \cdot 5 = 2y \cdot x \implies 50 = 2xy \implies xy = 25$$

Thus, the value of xy is $\boxed{25}$.

Sample 2

Prompt

Find the largest possible real part of

$$(75+117i)z+\frac{96+144i}{z}$$

where z is a complex number with $|z|=4$.

Response

Given the expression $(75+117i)z+\frac{96+144i}{z}$ where $|z|=4$, we start by letting $z=4e^{i\theta}$. Then, the expression becomes:

$$(75+117i)4e^{i\theta}+\frac{96+144i}{4e^{i\theta}}=4(75+117i)e^{i\theta}+\frac{96+144i}{4}e^{-i\theta}=4(75+117i)e^{i\theta}+(24+36i)e^{-i\theta}$$

Next, we express $e^{i\theta}$ and $e^{-i\theta}$ in terms of sine and cosine:

$$e^{i\theta}=\cos\theta+i\sin\theta \quad \text{and} \quad e^{-i\theta}=\cos\theta-i\sin\theta$$

Substituting these into the expression, we get:

$$4(75+117i)(\cos\theta+i\sin\theta)+(24+36i)(\cos\theta-i\sin\theta)$$

We expand both products:

$$4(75\cos\theta+75i\sin\theta+117i\cos\theta-117\sin\theta)+(24\cos\theta-24i\sin\theta+36i\cos\theta+36\sin\theta)$$

Combining like terms, we have:

$$4(75\cos\theta-117\sin\theta)+4i(75\sin\theta+117\cos\theta)+24\cos\theta+36\sin\theta+i(36\cos\theta-24\sin\theta)$$

$$=(300\cos\theta-468\sin\theta+24\cos\theta+36\sin\theta)+i(300\sin\theta+468\cos\theta+36\cos\theta-24\sin\theta)$$

$$=(324\cos\theta-432\sin\theta)+i(276\sin\theta+504\cos\theta)$$

The real part of this expression is $324\cos\theta-432\sin\theta$. To find the maximum value of $324\cos\theta-432\sin\theta$, we use the fact that the maximum value of $a\cos\theta+b\sin\theta$ is $\sqrt{a^2+b^2}$. Here, $a=324$ and $b=-432$, so we have:

$$\sqrt{324^2+(-432)^2}=\sqrt{104976+186624}=\sqrt{291600}=540$$

Therefore, the largest possible real part of $(75+117i)z+\frac{96+144i}{z}$ is $\boxed{540}$.