FROM LITERATURE TO REASONING: GENERALIZABLE REWARD MODELS FOR AI OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

AI4AI systems rely on reward models (RMs) to efficiently evaluate candidate system modifications. However, scenario-specific RM training often improves performance on seen tasks while degrading out-of-distribution (OOD) generalization. To address this issue, we propose LitRM, a literature-grounded reward modeling framework that transforms scientific knowledge into context-specific reasoning supervision. LitRM incorporates structured optimization knowledge through continual pretraining and retrieves mechanism-level evidence to guide a teacher model in reasoning over the current system state (BASE) and candidate modification (DIFF). The resulting reasoning guides evidence extraction, while joint training with teacher- and self-generated reasoning enables the student model to evaluate candidates without external retrieval at inference time. Experiments on multiple benchmarks show that LitRM consistently improves OOD generalization over current methods, yielding a 15.40 percentage-point gain in average OOD AUC while maintaining strong performance on seen tasks.

1 INTRODUCTION

AI has increasingly become an efficient tool for improving AI itself, giving rise to the emerging paradigm of AI for AI (AI4AI). Recent systems use large language models and autonomous agents to propose and refine model architectures, training strategies, data pipelines, and system configurations (Lu et al., 2026; Nathani et al., 2025). Since executing every candidate modification can be prohibitively expensive, reward models (RMs) are increasingly used to evaluate a proposed modification (DIFF) with respect to the current system state (BASE) before execution, thereby guiding search, ranking, and optimization (Chen et al., 2026b; Wang et al., 2025).

A common AI4AI approach is to train scenario-specific RMs on historical optimization trajectories from a target repository or task (Wang et al., 2025; Jia, 2024). While such training substantially improves evaluation on familiar scenarios, we observe a consistent generalization gap: *Seen performance improves, whereas out-of-distribution (OOD) performance often stagnates or degrades*. Figure 1 illustrates this phenomenon across different reward-modeling approaches and model scales, with details provided in Appendix B.1. These results suggest that existing RMs may learn local correlations between repositories, modifications, and outcomes rather than transferable optimization mechanisms.

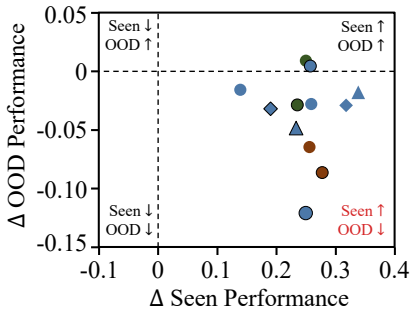

Figure 1: Seen–OOD generalization gap in scenario-specific reward models. Most models improve Seen AUC after training but degrade on OOD data.

Such a limitation is also difficult to address with existing reward-modeling approaches. First, increasing model scale does not consistently close the Seen–OOD gap (Razin et al., 2026; Shu et al., 2023). In addition, strong general-purpose LLMs remain unreliable when evaluating specific system states and candidate modifications (Szymanski et al., 2025; Son et al., 2024). Generative reward models and chain-of-thought reasoning provide richer intermediate reasoning, but the generated reasoning is not necessarily grounded in optimization mechanisms relevant to the current scenario (Wang et al.,

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.

2026). The key challenge is therefore not simply acquiring more knowledge, but learning to apply transferable optimization knowledge to the current system state and candidate modification.

To address this challenge, we propose **LitRM**, a literature-grounded reward modeling framework that converts scientific knowledge into context-specific evaluation reasoning. LitRM internalizes structured optimization knowledge through continual pretraining and retrieves mechanism-level evidence to construct scenario-specific teacher reasoning for each BASE–DIFF pair. The student then learns to use generated reasoning to guide evidence extraction from BASE and DIFF for reward prediction. During training, teacher-provided reasoning supervises reasoning generation, while self-generated reasoning is used for reward learning, enabling evaluation without external retrieval or teacher assistance at inference time. Our contributions are summarized as follows:

- We identify and systematically characterize a generalization issue in scenario-specific reward modeling for AI4AI, where improved performance on seen optimization scenarios can coincide with degraded OOD performance across repositories and optimization types.
- We propose LitRM, which incorporates literature-derived optimization knowledge through continual pretraining, context-specific teacher supervision, student-generated reasoning, and reasoning-guided evidence reading over the original BASE–DIFF input.
- Extensive experiments across multiple AI4AI benchmarks, model scales, and OOD settings show that LitRM improves average OOD AUC by 15.40 percentage points over the strongest baseline while maintaining strong Seen performance. Ablation and sensitivity studies further validate the contributions of reasoning supervision, evidence reading, and literature grounding.

2 Related Works and Background

In this section, we review the background of AI4AI, reward modeling, and reasoning-based evaluation, followed by related work on reward generalization and external knowledge augmentation.

AI for AI and Automated System Optimization. Recent advances in large language models and autonomous agents have enabled increasingly automated AI development, including algorithm design, code generation, experiment execution, and iterative system optimization (Chan et al., 2025; Lu et al., 2024). This emerging AI4AI paradigm aims to reduce human effort and accelerate exploration over large optimization spaces (Jiang et al., 2025). Since directly executing every candidate modification is expensive, learned evaluators are increasingly used to prioritize promising candidates before costly validation (Chen et al., 2026a).

Reward Modeling and Generalization. Reward models (RMs) provide scalable evaluation signals for ranking, preference learning, and optimization (Li et al., 2024; Ivison et al., 2024). While conventional RMs are largely discriminative, recent work explores generative and foundation reward models with stronger cross-task transfer (Zhang et al., 2025; Wang et al., 2025). Nevertheless, OOD generalization remains difficult when training data are limited to specific repositories or tasks, where gains on seen scenarios may not transfer to unseen ones (Razin et al., 2026; Yang et al., 2024).

Generative and Reasoning Reward Models. Recent work augments reward modeling with critiques, rationales, rubrics, and chain-of-thought reasoning (Zhang et al., 2025). Generative Verifiers cast reward modeling as next-token prediction with chain-of-thought verification, while RM-R1, Think-RM, CLoud, and related approaches further combine reasoning supervision with reward learning (Chen et al., 2026b; Hong et al., 2026; Ankner et al., 2024). In particular, CLoud distills teacher critiques, generates student critiques, and jointly optimizes critique generation and reward prediction. However, existing approaches largely rely on parametric knowledge or task-specific supervision, without explicitly grounding reasoning in transferable external knowledge or using reasoning to selectively extract evidence from the underlying system state and candidate modification.

External Knowledge and Literature-Grounded Reasoning. Retrieval-augmented models incorporate external knowledge to improve factuality and reasoning (Lewis et al., 2020; Asai et al., 2024), while scientific literature provides rich evidence about optimization mechanisms, conditions, and empirical outcomes (Lála et al., 2023). However, retrieving a successful method does not establish that the same mechanism applies to a new system state or candidate modification. Existing retrieval-based approaches therefore leave open how external scientific knowledge can be adapted into context-specific reasoning for reward modeling.

3 REASONING-AUGMENTED REWARD MODELING

To incorporate explicit reasoning into AI4AI reward modeling, we first formulate the evaluation problem and then describe a teacher–student training procedure that separates teacher-supervised reasoning generation from reward learning with student-generated reasoning.

3.1 TASK FORMULATION AND MODEL OVERVIEW

We study candidate optimization evaluation in AI4AI systems. Given the current system state $\boldsymbol{b}$ (BASE) and a proposed modification $\boldsymbol{d}$ (DIFF), the goal is to estimate the quality of the modification before incurring the cost of actual execution. For each training instance, the modification is executed offline to obtain a binary outcome $y \in \{0, 1\}$ indicating whether it satisfies a predefined improvement criterion. The training data are therefore written as,

$$\mathcal{D}_{\text{train}} = \left\{ (\boldsymbol{x}_i, y_i) \right\}_{i=1}^{N}, \quad \text{where } \boldsymbol{x}_i = (\boldsymbol{b}_i, \boldsymbol{d}_i).$$

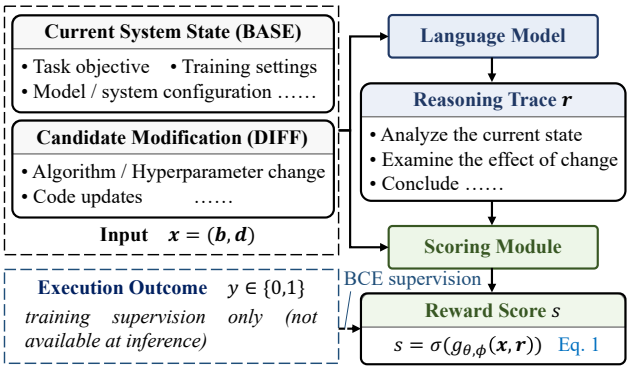

Figure 2: Overview of the reasoning-augmented reward model for evaluating candidates from BASE and DIFF.

Here, the execution outcome is used only as supervision and is not available when evaluating a new candidate.

Figure 2 illustrates the model interface. Our model contains a shared language-model backbone together with a scoring head. Given an input instance $\boldsymbol{x}$, the language model first generates a reasoning trace $\boldsymbol{r}$ autoregressively,

$$\boldsymbol{r} \sim p_\theta(\cdot \mid \boldsymbol{x}),$$

where θ denotes the language-model parameters. Conditioned on both the input $\boldsymbol{x}$ and the generated reasoning $\boldsymbol{r}$, the scoring head then produces a continuous reward score

$$g_{\theta,\phi}(\boldsymbol{x}, \boldsymbol{r}) \in \mathbb{R}, \qquad s_{\theta,\phi}(\boldsymbol{x}, \boldsymbol{r}) = \sigma(g_{\theta,\phi}(\boldsymbol{x}, \boldsymbol{r})), \tag{1}$$

where ϕ denotes the parameters of the scoring head and $\sigma(\cdot)$ is the sigmoid function.

The reasoning trace provides an explicit analysis of why a modification may or may not work under the current system state, while the continuous score serves as the final signal for ranking and evaluation. Predictions are based on this score rather than any discrete judgment contained in the generated reasoning. We next describe how teacher and student reasoning provide complementary supervision during training.

3.2 TEACHER–STUDENT TRAINING WITH COMPLEMENTARY SUPERVISION

With the reasoning-conditioned reward model defined above, we use a three-step teacher–student training procedure, as illustrated in Figure 3. The key design is the complementary use of teacher and student reasoning: qualified teacher traces supervise reasoning generation, whereas student-generated traces are paired with execution outcomes for reward learning. The teacher model is used only to construct offline supervision and is not required at inference time.

Stage 1: Teacher Reasoning Warm-up. For each training instance, a strong teacher generates a reasoning trace and judgment from the current system state and candidate modification. Following the rationale-reconstruction idea of STaR (Zelikman et al., 2022), if the initial prediction is incorrect or fails predefined quality checks, we provide the ground-truth outcome and allow one refinement conditioned on the correct answer. This refinement is used only for offline teacher-data construction. Qualified traces are retained for supervised fine-tuning to warm-start structured scenario reasoning.

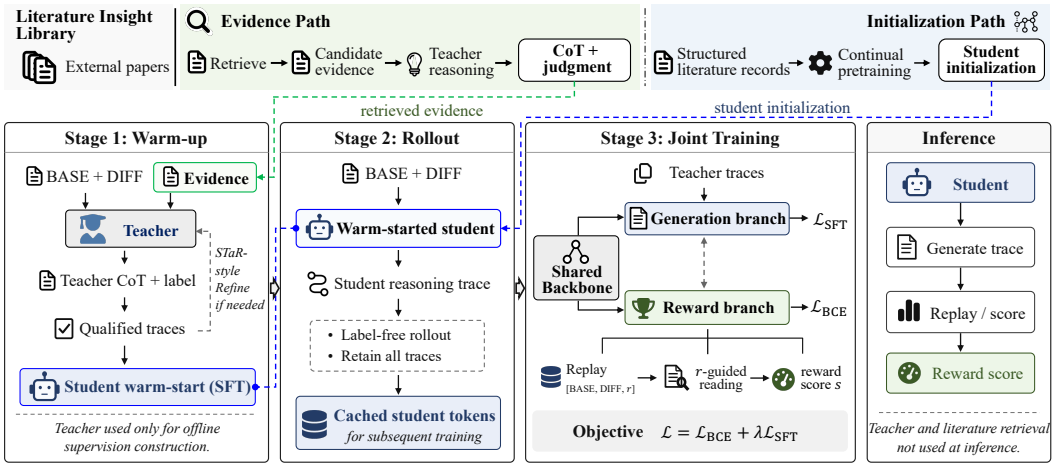

Figure 3: Training procedure of LitRM with complementary teacher and student supervision.

Stage 2: Student Rollout. After warm-up, we freeze a student snapshot and perform one offline blind rollout over the training data. Each instance produces one reasoning trace without access to its ground-truth label, and all traces are retained, including imperfect ones. The generated tokens and segment boundaries are cached and fixed throughout joint training, exposing reward learning to the reasoning distribution actually produced by the student. During optimization, the cached tokens are replayed while their hidden representations are recomputed by the current model, without new traces being regenerated.

Stage 3: Joint Training. The student jointly optimizes reasoning generation and reward prediction. Qualified teacher traces supervise the generation branch, preserving high-quality structured reasoning, while student-generated traces supervise the reward branch together with execution outcomes, helping to mitigate the mismatch between training-time and inference-time reasoning. Both objectives update the shared backbone. We formalize the two branches in the next subsection.

3.3 Joint Reasoning and Reward Optimization

We now formalize the joint optimization. The two branches share the same language-model backbone but use complementary reasoning supervision: teacher traces train reasoning generation, while student-generated traces train reward prediction against execution outcomes.

Branch 1: Teacher-Supervised Reasoning Generation. For each sample with a qualified teacher trace, we construct a target sequence $\boldsymbol{a}_i^T$ containing the teacher reasoning, required control tokens, and the ground-truth label. Given input $\boldsymbol{x}_i$, the student is trained with teacher forcing:

$$\ell_{\text{SFT},i}(\theta) = -\frac{1}{T_i} \sum_{t=1}^{T_i} \log p_\theta \left(a_{i,t}^T \mid \boldsymbol{x}_i, \boldsymbol{a}_{i<t}^T \right), \tag{2}$$

where T_i is the target-sequence length. The loss is averaged over target tokens only, excluding the input prompt. This branch preserves structured reasoning during reward optimization.

Branch 2: Reward Learning with Student-Generated Reasoning. For each training sample, the student-generated reasoning $\boldsymbol{r}_i^S$ from Stage 2 is paired with the original input $\boldsymbol{x}_i$. The scoring head produces

$$g_i = g_{\theta,\phi} \left(\boldsymbol{x}_i, \boldsymbol{r}_i^S \right), \qquad s_i = \sigma(g_i),$$

and is supervised by the execution outcome y_i using weighted binary cross-entropy:

$$\ell_{\text{BCE},i}(\theta, \phi) = -w^+ y_i \log s_i - (1 - y_i) \log(1 - s_i), \tag{3}$$

where $w^+ = N_-/N_+$ compensates for class imbalance. The execution outcome is used only as a training target and is never provided as an input to the student scoring function. Because all

rollout traces are retained, this branch learns from the reasoning distribution actually produced by the student, including imperfect traces.

Joint Objective. Let $\mathcal{B}$ denote all valid training samples in a batch and $B_T \subseteq \mathcal{B}$ the subset with qualified teacher traces. We average the two objectives over their respective sample sets:

$$\mathcal{L}_{\text{joint}}(\theta, \phi) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \ell_{\text{BCE},i}(\theta, \phi) + \lambda \frac{1}{|B_T|} \sum_{i \in B_T} \ell_{\text{SFT},i}(\theta), \tag{4}$$

where λ controls the reasoning-supervision weight and the SFT term is zero when $B_T = \emptyset$. Thus, all valid samples contribute reward supervision, whereas only qualified teacher samples contribute reasoning supervision.

Inference. At inference time, the teacher is removed. Given a test input $\boldsymbol{x}$, the student first generates a reasoning trace $\boldsymbol{r}^S$ and then produces a continuous reward score using the same backbone and scoring head. Candidate ranking and AUC are based on this continuous score rather than the discrete answer generated by the language model.

4 Literature-Grounded Reward Modeling

The reasoning-augmented reward model in Sec. 3 can learn from explicit reasoning, but its effectiveness still depends on the knowledge encoded in that reasoning. We therefore ground reward modeling in scientific literature through two complementary paths. First, structured optimization knowledge is injected into the student via continual pretraining. Second, retrieved mechanism-level evidence is adapted into context-specific teacher reasoning for the current BASE–DIFF pair. The resulting reasoning further guides evidence extraction from BASE and DIFF during prediction.

4.1 Literature-Grounded Knowledge Construction

Scientific papers often report controlled comparisons between a baseline system and a proposed modification, together with mechanism analysis and experimental outcomes. We extract such comparisons into a structured knowledge base. Each literature record is represented as,

$$k_j = \left(\boldsymbol{b}_j^P, \boldsymbol{d}_j^P, \boldsymbol{r}_j^P, y_j^P, \boldsymbol{m}_j^P\right), \qquad \mathcal{K} = \{k_j\}_{j=1}^M, \tag{5}$$

where $\boldsymbol{b}_j^P$ and $\boldsymbol{d}_j^P$ describe the baseline configuration and its modification, $\boldsymbol{r}_j^P$ contains the mechanism-oriented reasoning derived from the paper, y_j^P denotes the outcome inferred from the controlled comparison, and $\boldsymbol{m}_j^P$ stores provenance and experimental metadata.

We initialize the student with continual pretraining (CPT) on literature records. Each record is serialized as BASE, DIFF, mechanism, and conclusion and trained with next-token prediction. Let $\mathcal{C}$ denote the resulting text segments and w a token sequence:

$$\mathcal{L}_{\text{CPT}}(\theta) = -\frac{1}{|\mathcal{C}|} \sum_{w \in \mathcal{C}} \frac{1}{|w| - 1} \sum_{t=2}^{|w|} \log p_\theta \left(w_t \mid w_{<t}\right).$$

This stage injects general relationships among system states, modifications, mechanisms, and outcomes before the teacher–student training in Sec. 3.

4.2 Context-Specific Reasoning Adaptation

Literature evidence cannot be directly transferred to a new optimization instance because the same modification may behave differently under different conditions. We therefore retrieve relevant paper evidence and let the teacher adapt its underlying mechanism to the current BASE–DIFF pair.

For each training instance $\boldsymbol{x}$, we construct two retrieval views: a joint view describing the overall system context and modification, and an action view emphasizing the changed component and its

role. Each view independently retrieves candidate records from $\mathcal{K}$ using embedding similarity, and the two rankings are fused with reciprocal rank fusion (RRF):

$$\mathrm{RRF}(k \mid \boldsymbol{x}) = \sum_{v \in \{\mathrm{joint},\mathrm{action}\}} \frac{\mathbb{I}[k \in \mathcal{N}_v(\boldsymbol{x})]}{c + \mathrm{rank}_v(k \mid \boldsymbol{x})}. \tag{6}$$

The top-ranked evidence is provided to the teacher together with the current BASE and DIFF. Rather than copying the conclusion of a retrieved paper, the teacher determines whether its mechanism is applicable under the current system configuration and constructs a scenario-specific reasoning trace. The reasoning is required to connect the relevant condition in BASE, the role of DIFF, and the mechanism through which the modification may affect the target objective. Retrieved results and literature outcomes are therefore treated as supporting evidence rather than direct prediction labels.

The resulting teacher traces follow the same refinement and quality-control procedure described in Sec. 3.2 and are subsequently used by the reasoning-supervision branch in Sec. 3.3. Literature retrieval is used only during training-time teacher construction and is not required at inference time.

4.3 Reasoning-Guided Reward Modeling

Finally, we use the generated reasoning not as the reward itself, but as a query for identifying evidence relevant to the current judgment. Given the final-layer token representations of BASE, DIFF, and reasoning, denoted by H_b, H_d, and H_r, respectively, we first aggregate the reasoning representation and construct a query:

$$\bar{h}_r = \frac{1}{L_r} \sum_{t=1}^{L_r} H_{r,t}, \qquad q = \mathrm{LN}_q(\bar{h}_r) W_Q.$$

For each input segment $s \in \{b, d\}$, we project its token representations into keys and values:

$$K_s = \mathrm{LN}_e(H_s) W_K, \qquad V_s = \mathrm{LN}_e(H_s) W_V.$$

The reasoning query then attends separately to BASE and DIFF:

$$\alpha_s = \mathrm{softmax}\left(\frac{q K_s^\top}{\sqrt{d_a}}\right), \qquad e_s = \alpha_s V_s, \qquad s \in \{b, d\}.$$

The two evidence vectors are concatenated and mapped to the final reward logit and score:

$$g_{\theta,\phi}(\boldsymbol{x}, \boldsymbol{r}) = \mathrm{MLP}_\phi\left([e_b; e_d]\right), \qquad s_{\theta,\phi}(\boldsymbol{x}, \boldsymbol{r}) = \sigma\left(g_{\theta,\phi}(\boldsymbol{x}, \boldsymbol{r})\right). \tag{7}$$

This design lets reasoning determine where to read from the current BASE and DIFF, while the final reward remains grounded in the original instance rather than the generated conclusion itself. The resulting score is optimized by the reward-learning objective in Sec. 3.3.

5 Experimental Studies

To evaluate LitRM, we compare against representative discriminative, generative, and reasoning-based reward models on multiple AI4AI benchmarks, followed by generalization, ablation, and sensitivity studies. Detailed settings are provided in Appendix A.

Benchmarks and Datasets. We use three AI4AI benchmarks: *TraceML* (Yan et al., 2026), *ForeAgent* (Zheng et al., 2026), and *RecOptBench*. *TraceML* contains historical machine-learning development trajectories, *ForeAgent* uses executed solution pairs from competition tasks, and *RecOptBench* is an internal benchmark of real recommendation-system optimization trajectories. Each includes Seen and OOD splits defined by repository, task, or optimization type.

Baselines. We compare with Discriminative RM (Ouyang et al., 2022), GenRM-Direct and GenRM-CoT (Zhang et al., 2025), GRM-SFT and GRM-CoT (Yang et al., 2024), and CLoud (Ankner et al., 2024). When necessary, methods are adapted to the same pointwise candidate-evaluation setting.

Setup. Unless otherwise specified, all methods share the same backbone, training data, and evaluation protocol. Main experiments use Qwen3-1.7B-Base, with additional scales and model families studied separately. All experiments are conducted on NVIDIA A100 GPUs. LitRM uses a literature corpus of 30,361 structured insights extracted from 19,050 ICLR, ICML, and NeurIPS papers from 2018–2025. We report ROC-AUC on both Seen and OOD settings.

5.1 Overall Performance

We first evaluate whether LitRM can improve reward modeling on both familiar and unseen optimization scenarios. For each benchmark, the Seen setting evaluates tasks or repositories covered by the training distribution, while OOD performance is averaged across multiple held-out subsets representing different distribution shifts, such as unseen repositories or optimization types. Table 1 compares LitRM with six representative baselines under the same evaluation protocol. LitRM achieves the best performance across all three benchmarks in both Seen and OOD settings. The improvement is particularly pronounced under distribution shift, yielding a 15.40 pp gain in average OOD AUC over the strongest baseline ($54.59\% \rightarrow 69.99\%$). Meanwhile, LitRM maintains strong Seen performance, suggesting that improved OOD generalization does not require sacrificing performance on familiar scenarios.

Table 1: Overall performance (AUC, %) on three AI4AI benchmarks. The best results are shown in **bold** and the second-best results are underlined.

Method	Seen				OOD			
	TraceML	ForeAgent	RecO.	Avg.	TraceML	ForeAgent	RecO.	Avg.
Disc. RM	69.08	74.70	79.07	74.28	52.79	41.73	49.59	48.04
GenRM-Direct	81.05	75.52	82.43	79.67	58.10	54.26	51.40	54.59
GenRM-CoT	55.66	62.89	57.82	58.79	50.87	49.38	48.02	49.42
GRM-SFT	80.74	77.55	79.74	79.34	47.49	45.15	51.84	48.16
GRM-CoT	79.02	76.85	77.81	77.89	52.54	48.47	49.16	50.06
CLoud	80.14	73.32	81.04	78.17	54.51	50.94	49.76	51.74
LitRM	**82.79**	**80.02**	**86.17**	**82.99**	**66.86**	**72.54**	**70.56**	**69.99**

To further examine the training dynamics, Fig. 4 plots performance over training progress on the Seen and averaged OOD settings of TraceML and ForeAgent. While most baselines improve substantially on Seen data, their OOD performance shows limited improvement or even deteriorates as training progresses. LitRM instead achieves consistent gains across both settings, substantially widening the OOD margin while retaining strong Seen performance. These results suggest that literature-grounded reasoning helps reduce reliance on scenario-specific patterns and supports more transferable candidate evaluation.

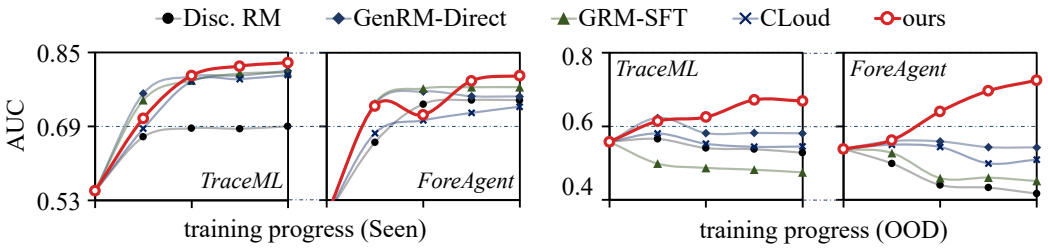

Figure 4: Performance trajectories during training on the Seen and OOD of *TraceML* and *ForeAgent*.

5.2 Generalization Study

Beyond overall performance, we further examine whether LitRM generalizes across model backbones and different forms of distribution shift. Additional results are provided in Appendix B.

- **Generalization across Model Scales and Families.** We first evaluate whether LitRM remains effective across different backbone configurations. Table 2 compares multiple Qwen3 scales and an additional Gemma backbone on *ForeAgent*. Increasing model size alone does not consistently improve OOD performance and can even amplify OOD degradation after scenario-specific tuning. In contrast, LitRM consistently improves OOD AUC across the evaluated backbones while maintaining strong Seen performance. These results suggest that the gains of LitRM are not tied to a particular model scale or family.

- **Generalization across OOD Types.** We next evaluate different forms of distribution shift on *RecOptBench*, including cross-repository, unseen-category, temporal, and compositional OOD. Detailed construction protocols for the additional RecOptBench OOD splits are provided in Appendix A.2. As shown in Table 3, LitRM achieves the best performance across all four settings, with the largest gains on cross-repository and unseen-category shifts. This indicates that its generalization benefit extends beyond a single OOD construction and remains effective under qualitatively different distribution shifts.

Table 2: Generalization across model scales and families on *ForeAgent* (AUC, %).

Model	Seen			OOD		
	Base	Tuned	ours	Base	Tuned	ours
Q3-1.7B	49.72	75.52	**80.02**	53.83	54.26	**72.54**
Q3-4B	58.03	77.05	**81.53**	47.13	43.94	**66.52**
Q3-8B	56.17	79.53	**81.39**	52.58	47.74	**67.79**
Gem-4B	61.15	77.43	**85.51**	54.23	49.04	**65.71**

Table 3: Generalization across OOD types on *RecOptBench* (AUC, %).

Method	Seen	X-Repo	Unseen	Temp.	Comp.
Disc. RM	79.07	49.59	66.91	70.12	74.46
GenRM-D	82.43	51.40	67.85	74.37	78.20
GRM-SFT	79.74	51.84	67.51	72.43	76.39
CLoud	81.04	49.76	63.90	72.01	73.58
ours	**86.17**	**70.56**	**77.15**	**80.89**	**83.68**

5.3 Ablation Study

We conduct a progressive ablation on TraceML to isolate the contribution of each component in LitRM. As shown in Fig. 5, teacher-provided reasoning improves average OOD performance but remains insufficient for reliable Seen evaluation. Introducing student-generated reasoning for reward learning, the reasoning-guided evidence reader, and literature-derived knowledge yields consistent additional gains.

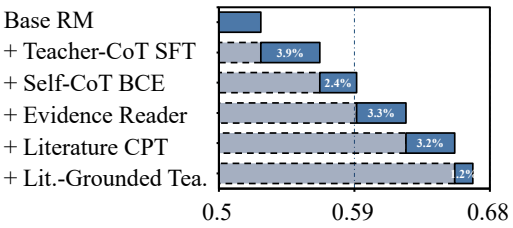

Figure 5: Incremental ablation of LitRM on *TraceML* (OOD Avg. AUC).

In particular, the average OOD AUC increases from 52.79% for the base RM to 66.86% for the full LitRM, while Seen AUC reaches 82.79%. The progressive gains from student-side reward learning, evidence reading, and literature grounding indicate that LitRM benefits from both transferable external knowledge and its integration into the student scoring process. Detailed ablation results are provided in Appendix B.

5.4 Sensitivity Analysis

We further study the sensitivity of LitRM to three key hyperparameters on TraceML. Figure 6 summarizes both performance and computational cost, with additional results provided in Appendix B.

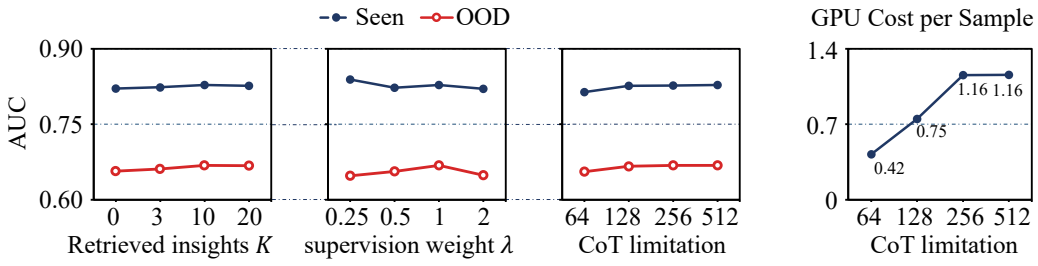

Figure 6: Sensitivity and efficiency analysis of LitRM on *TraceML*. We vary the number of retrieved insights K, the reasoning-supervision weight λ, and the CoT length limit, and report Seen/OOD AUC together with the corresponding GPU cost per sample.

- **Retrieved evidence budget.** Increasing the number of retrieved literature insights from $K = 0$ to $K = 10$ improves the average OOD AUC from 65.67% to 66.86%, demonstrating the benefit of providing sufficient external evidence. Increasing the budget further to $K = 20$ brings little additional improvement. We therefore use $K = 10$ as the default setting.
- **Reasoning-supervision weight.** LitRM remains effective across different values of the SFT weight λ, while $\lambda = 1.0$ achieves the best average OOD performance of 66.86%. Smaller weights provide weaker reasoning supervision, whereas an excessively large weight also degrades performance. We therefore set $\lambda = 1.0$ by default.
- **Reasoning budget.** Increasing the CoT limit from 64 to 256 tokens improves average OOD AUC from 65.60% to 66.86% as truncation decreases. Extending the limit to 512 tokens yields negligible additional accuracy or GPU cost, since most generated traces already fit within 256 tokens. We therefore use 512 tokens by default to accommodate longer traces without materially increasing computation.

6 Discussion

We briefly discuss why LitRM generalizes better, provide qualitative evidence from case studies, and summarize its main limitations. Additional analyses and complete examples are provided in Appendix C.

- **Why Does LitRM Generalize Better?** Scenario-specific RMs may rely on local patterns in $\boldsymbol{x} = (\boldsymbol{b}, \boldsymbol{d})$ that fail to transfer across repositories, tasks, or optimization types. Similar shortcut reliance and degraded OOD transfer have been observed in reward modeling and language models more broadly (Jia, 2024; Li et al., 2025; Sun et al., 2024). LitRM instead introduces reasoning $\boldsymbol{r}$ to connect the current limitation, proposed modification, and underlying mechanism, and then uses $\boldsymbol{r}$ to selectively read evidence from BASE and DIFF. In this way, reasoning determines where to read, while the original BASE–DIFF evidence determines what to predict. This factorization encourages prediction based on transferable mechanisms and context-specific evidence rather than scenario-specific correlations.
- **Case Studies.** Qualitative examples in Appendix C further illustrate this effect. Literature grounding helps the teacher identify concrete bottlenecks, such as missing temporal information or retrieval truncation, and construct more mechanism-specific supervision. On OOD examples, the literature-enhanced student also shows better agreement among its reasoning, continuous reward score, and observed outcome, without using retrieval or a teacher at inference time.
- **Limitations and Future Works.** LitRM depends on the coverage and quality of the literature corpus, and distribution or temporal gaps in source data can limit the knowledge available to the model. Retrieval may also surface topically related but mechanism-inapplicable evidence, a known challenge in retrieval-based systems, making teacher-side filtering and adaptation important. Imperfect retrieval or teacher reasoning can therefore introduce noisy supervision. In addition, our study focuses on AI4AI tasks with explicit BASE–DIFF comparisons and execution feedback. Extending LitRM to settings with sparse evidence or weaker feedback remains an important direction.

7 Conclusion

We study the generalization problem of scenario-specific reward models in AI4AI, where improved performance on familiar optimization scenarios can coincide with degraded OOD performance. To address this issue, we propose LitRM, which grounds reward modeling in transferable optimization knowledge from scientific literature. LitRM combines literature-based knowledge construction, context-specific teacher reasoning, student-generated reward learning, and reasoning-guided evidence reading over BASE and DIFF. Experiments across multiple AI4AI benchmarks, model scales, and OOD settings show that LitRM improves average OOD AUC from 54.59% to 69.99% ($+15.40$ pp) while maintaining strong Seen performance. These results suggest that grounding reward models in mechanism-level knowledge can reduce reliance on local optimization patterns and provide a promising direction for more generalizable and reliable AI4AI evaluation.

AI USE STATEMENT

We used generative AI tools to assist with literature search, scientific writing, and language editing. Generative AI was not used to generate experimental results or replace empirical evaluation.

ETHICS STATEMENT

This work uses publicly accessible research literature and an internally constructed industrial benchmark. The literature corpus is derived from public papers, and we plan to release the resulting structured records and processing artifacts. RecOptBench is constructed from proprietary recommendation-system optimization records and cannot be released at submission time. We provide details of its construction, labeling, and data splits in the appendix, and may release a sanitized or derived version subject to confidentiality requirements.

REPRODUCIBILITY STATEMENT

We provide detailed information to support reproducibility throughout the paper and appendix. Section 3 and Section 4 describe the training procedure and objectives, while Appendix A documents dataset construction, evaluation protocols, baseline adaptations, model and optimization settings, literature-corpus construction, retrieval, teacher supervision, and student rollout. Appendix B provides additional experimental results and complete ablation and sensitivity analyses. We also report the data splits, training budgets, hyperparameters, and evaluation metrics used in our experiments. An anonymized implementation and data-processing scripts are provided in the supplementary material.

REFERENCES

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D Chang, and Prithviraj Ammanabrolu. Critique-out-loud reward models. *arXiv preprint arXiv:2408.11791*, 2024.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, volume 2024, pp. 9112–9141, 2024.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 50466–50494, 2025.

Jiefeng Chen, Bhavana Dalvi Mishra, Jaehyun Nam, Rui Meng, Tomas Pfister, and Jinsung Yoon. Mars: Modular agent with reflective search for automated ai research. *arXiv preprint arXiv:2602.02660*, 2026a.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, et al. Rm-r1: Reward modeling as reasoning. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 88313–88342, 2026b.

Ilgee Hong, Changlong Yu, Liang Qiu, Weixiang Yan, Zhenghao Xu, Haoming Jiang, Qingru Zhang, Qin Lu, Xin Liu, Chao Zhang, et al. Think-rm: Enabling long-horizon reasoning in generative reward models. *Advances in Neural Information Processing Systems (NeurIPS)*, 38:174227–174251, 2026.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. Unpacking DPO and PPO: Disentangling best practices for learning from preference feedback. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Chen Jia. Generalizing reward modeling for out-of-distribution preference learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pp. 107–124. Springer, 2024.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodriques, and Andrew D White. Paperqa: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:9459–9474, 2020.

Jiaxiang Li, Siliang Zeng, Hoi-To Wai, Chenliang Li, Alfredo Garcia, and Mingyi Hong. Getting more juice out of the sft data: Reward learning from human demonstration improves sft for llm alignment. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:124292–124318, 2024.

Zichao Li, Xueru Wen, Jie Lou, Yuqiu Ji, Yaojie Lu, Xianpei Han, Debing Zhang, and Le Sun. The devil is in the details: Tackling unimodal spurious correlations for generalizable multimodal reward models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 36281–36297. PMLR, 13–19 Jul 2025.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, et al. Mlgym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*, 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 (NeurIPS)*, 35:27730–27744, 2022.

Noam Razin, Yong Lin, Jiarui Yao, and Sanjeev Arora. Why is your language model a poor implicit reward model? In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 24586–24622, 2026.

Yang Shu, Xingzhuo Guo, Jialong Wu, Ximei Wang, Jianmin Wang, and Mingsheng Long. Clipood: Generalizing clip to out-of-distributions. In *InternationalConferencee on Machine Learning (ICML)*, pp. 31716–31731. PMLR, 2023.

Guijin Son, Hyunwoo Ko, Hoyoung Lee, Yewon Kim, and Seunghyeok Hong. Llm-as-a-judge & reward model: What they can and cannot do. *arXiv preprint arXiv:2409.11239*, 2024.

Zechen Sun, Yisheng Xiao, Juntao Li, Yixin Ji, Wenliang Chen, and Min Zhang. Exploring and mitigating shortcut learning for generative large language models. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 6883–6893, Torino, Italia, May 2024. ELRA and ICCL.

Annalisa Szymanski, Noah Ziems, Heather A Eicher-Miller, Toby Jia-Jun Li, Meng Jiang, and Ronald A Metoyer. Limitations of the llm-as-a-judge approach for evaluating llm outputs in expert knowledge tasks. In *Proceedings of the 30th International Conference on Intelligent User Interfaces (IUI)*, pp. 952–966, 2025.

Chenglong Wang, Yang Gan, Yifu Huo, Yongyu Mu, Qiaozhi He, Murun Yang, Bei Li, Tong Xiao, Chunliang Zhang, Tongran Liu, et al. Gram: A generative foundation reward model for reward generalization. *arXiv preprint arXiv:2506.14175*, 2025.

Chenglong Wang, Yongyu Mu, Hang Zhou, Yifu Huo, Ziming Zhu, Jiali Zeng, Murun Yang, Bei Li, Xiaoyang Hao, Chunliang Zhang, et al. Gram-r^2: Self-training generative foundation reward models for reward reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 40, pp. 33395–33403, 2026.

Jiarui Yan, Weiwei Sun, Sijie Li, Wenhan Li, and Yiming Yang. TraceML: An empirical analysis of human-agent planning in machine learning development. In *COLM 2026 WAB*, 2026.

Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for llms. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:62279–62309, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Seyed Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 12476–12505, 2025.

Jingsheng Zheng, Jintian Zhang, Yujie Luo, Yuren Mao, Yunjun Gao, Lun Du, Huajun Chen, and Ningyu Zhang. Can we predict before executing machine learning agents? In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3941–3974, 2026.

A Detailed Experimental Settings

A.1 Task Definition and Evaluation Protocol

Each evaluation instance consists of an input

$$\boldsymbol{x} = (\mathrm{BASE}, \mathrm{DIFF}),$$

where **BASE** describes the current or reference system and **DIFF** describes a candidate modification. The reward model outputs a continuous score used to rank candidate modifications. Ground-truth labels are derived from actual execution outcomes and are used only for supervision and evaluation.

For all three benchmarks, we normalize the direction of the evaluation metric before computing relative improvement. For metrics where larger values are better,

$$\Delta_{\mathrm{rel}} = \frac{s_{\mathrm{new}} - s_{\mathrm{old}}}{|s_{\mathrm{old}}|},$$

while for metrics where smaller values are better,

$$\Delta_{\mathrm{rel}} = \frac{s_{\mathrm{old}} - s_{\mathrm{new}}}{|s_{\mathrm{old}}|}.$$

A candidate is labeled positive if $\Delta_{\mathrm{rel}} \geq 0.10$ and negative otherwise. This task evaluates whether a candidate reaches the specified 10% relative-improvement threshold, rather than estimating improvement magnitude. Negative labels include unchanged, mildly improved, and degraded candidates. Samples with $|s_{\mathrm{old}}| \leq 10^{-12}$ are excluded because the relative improvement is undefined. We report ROC-AUC as the primary metric and preserve the natural class distribution of each test split. The default hyperparameters and final model checkpoints were fixed before examining the final OOD test results. The OOD sensitivity results in Section 5.4 characterize these prespecified settings and were not used for selection.

A.2 Benchmarks and Data Construction

TraceML. TraceML (Yan et al., 2026) contains version-level machine-learning development trajectories collected from Kaggle competitions, including states, actions, execution scores, and human/agent development histories. We use the official dataset snapshot and construct reward-modeling examples from human trajectories only.

We associate each action with its pre-modification state using the original trajectory identifiers and retain the raw scores and provenance fields only for labeling and auditing. Samples with missing scores, missing metric directions, or identifiable no-op modifications are removed. We do not expose execution scores or ground-truth labels to the model input.

Competition instances from different years are grouped into fixed competition families. Entire competition families are held out for OOD evaluation, while the remaining trajectories are split by repository family and trajectory identifier to prevent actions from the same trajectory from appearing in multiple splits. The training set contains 13,353 examples covering 102 competition families. The Seen test set contains 2,937 examples, and the two OOD splits contain 878 and 857 examples, respectively.

ForeAgent. ForeAgent is constructed from the dataset released with *Can We Predict Before Executing Machine Learning Agents?* (Zheng et al., 2026). Each original example contains two independently executed solutions for the same machine-learning competition task. Unlike TraceML, the two solutions do not necessarily form a true parent–child modification sequence. We treat solution A as BASE and solution B as the candidate DIFF and predict whether B improves over A by at least 10%.

We use the `solutions_subset_50` subset and remove invalid scores, self-comparisons, conflicting code pairs, and inconsistent duplicates. Each solution is converted into a factual English summary covering data processing, features, model, training, validation, and inference. The summarizer processes each solution independently without access to execution scores, pairwise outcomes, or the other solution.

To reduce implementation leakage, we canonicalize code with AST-based identifiers and split by implementation group before retaining valid within-split comparisons. Entire competition tasks are held out for OOD evaluation, while the remaining tasks are split into training, validation, and Seen test sets. The resulting benchmark contains 5,541 training examples, 236 Seen validation examples, 242 Seen test examples, and 983 OOD examples.

RecOptBench. RecOpt is an internal industrial benchmark collected from real recommendation-system optimization trajectories. Each example contains the current system or experiment state as BASE, the subsequent modification as DIFF, and the execution outcome produced after deployment or offline evaluation.

We retain the cleaned binary labels provided by the upstream system. The upstream system assigns a positive label when the direction-normalized relative improvement is at least 10%, and a negative label otherwise, consistent with the criterion in Appendix A.1. We do not recompute these labels from the raw result fields. After removing samples with missing identifiers, missing BASE/DIFF content, or invalid labels, the benchmark contains 35,796 examples from 50 repositories. Two repository groups are held out entirely for OOD evaluation. The remaining repositories are split by BASE groups such that identical BASE identifiers, normalized BASE texts, and associated groups do not cross splits. This yields 20,383 training examples, 2,939 Seen validation examples, 2,709 Seen test examples, and two OOD test sets containing 3,189 and 6,576 examples.

Additional RecOptBench OOD splits. We additionally evaluate generalization under unseen-category, temporal, and compositional distribution shifts on RecOptBench.

For unseen-category evaluation, categories are defined by the second-level action label (L2) of DIFF. We conduct independent leave-one-category-out experiments for *Other Loss Design*, *Optimizer Replacement*, and *Decoder*. In each experiment, all examples belonging to the held-out category form the test set. We exclude candidate training examples whose BASE identifiers appear in the test set, regardless of their action categories. Training examples are then selected from the remaining pool

Table 4: Dataset statistics used in the experiments.

Benchmark	Train	Seen Dev	Seen Test	OOD Test	OOD Unit
TraceML	13,353	3,009	2,937	878 / 857	Competition family
ForeAgent	5,541	236	242	983	Competition task
RecOpt	20,383	2,939	2,709	3,189 / 6,576	Repository

using a fixed, split-specific MD5 ordering of sample identifiers. The Unseen column in Table 3 reports the arithmetic mean of the ROC-AUC values from these category-holdout experiments.

For temporal evaluation, we preserve the original chronological training and test partitions. Candidate-experiment timestamps span March 29–July 23, 2026 for the retained training examples and July 24–August 9, 2026 for the test examples. Both the BASE and candidate experiments of each retained pair fall within the corresponding partition's time period. The training and test sets share no BASE identifiers; no additional trajectory-level regrouping is performed. Training examples are selected using a fixed MD5 ordering of sample identifiers, while all eligible test examples are retained.

For compositional evaluation, we define a composition as a repository–action-category pair. The test set contains combinations absent from training, while each constituent repository and action category occurs in other training combinations. This evaluates generalization to unseen combinations of individually observed components.

A.3 Baseline Implementations

We compare LitRM with representative discriminative, generative, and reasoning-based reward models. All methods use the same dataset splits and evaluation protocol. Methods originally designed for pairwise preference modeling are adapted to pointwise candidate evaluation without constructing artificial preference pairs across unrelated BASE instances. LitRM shares the use of critique-based training with CLoud, while incorporating structured literature knowledge and reasoning-guided evidence reading. The comparisons evaluate complete training recipes and do not isolate architecture from external knowledge or training resources.

- **Discriminative RM.** Following the standard scalar reward-modeling paradigm (Ouyang et al., 2022), we obtain a representation from the language-model backbone using mean pooling or the final valid token and apply an MLP scoring head. Training uses focal binary cross-entropy, and inference requires only a single forward pass without generating reasoning.
- **GRM-SFT and GRM-CoT.** We adapt the generalizable reward-modeling framework of Yang et al. (2024). GRM-SFT augments discriminative reward learning with an auxiliary generation objective over positive BASE–DIFF transformations. GRM-CoT is our CoT-adapted variant in which the auxiliary generation target is replaced by teacher reasoning followed by the correct label. Both variants perform discriminative training on all valid samples, and their scoring heads read representations from the original BASE–DIFF input rather than generated reasoning.
- **GenRM-Direct and GenRM-CoT.** Following Generative Verifiers (Zhang et al., 2025), GenRM-Direct predicts the binary outcome directly as a generated token. GenRM-CoT first generates a reasoning trace and then predicts the binary answer token. Both are trained with language-model cross-entropy and use the logit difference between the positive and negative answer tokens as the continuous reward score.
- **LitRM without Literature.** To isolate the contribution of literature grounding, we also evaluate a variant that uses the same teacher–student supervision split and reasoning-guided evidence reader as LitRM but removes literature continual pretraining and literature-assisted teacher reasoning. Teacher reasoning is constructed only from BASE and DIFF.
- **LitRM.** LitRM first injects structured literature knowledge through continual pretraining. During teacher-data construction, retrieved literature evidence is provided to the teacher together with BASE and DIFF. The teacher may reject irrelevant evidence and is instructed to justify predictions through applicable mechanisms rather than paper authority. Qualified teacher traces supervise

reasoning generation, while student-generated traces supervise reward learning. At inference time, no teacher or literature retriever is used.

A.4 Model and Optimization Settings

Main experiments use Qwen3-1.7B-Base. We additionally evaluate Qwen3-4B-Base and Qwen3-8B-Base for scaling experiments and Gemma-3-4B-PT for cross-family evaluation. All experiments start from pretrained base models rather than instruction-tuned variants, and reasoning is generated autoregressively without relying on built-in chat templates or proprietary thinking modes.

Training is implemented with PyTorch 2.6 and Transformers 4.51.1. Experiments are conducted on NVIDIA A100 GPUs. For the 1.7B setting, generation and joint training use four GPUs with micro-batch size 1 per GPU and gradient accumulation of 8, giving an effective global batch size of 32. Larger models adjust micro-batch size and parameter sharding while preserving the effective global batch size.

Unless otherwise specified, LitRM uses AdamW with a backbone learning rate of 5×10^{-6} and a scoring-module learning rate of 1×10^{-4}. We use cosine learning-rate decay with a 10% warm-up ratio, weight decay of 10^{-4}, gradient clipping at 1.0, and BF16 forward computation. The maximum input length is 4,096 tokens, the maximum generated reasoning length is 512 tokens, and the maximum combined sequence length is 6,144 tokens. The reasoning-guided attention and scoring hidden dimensions are both set to 256, with GELU activation and dropout of 0.1.

The joint objective combines teacher-supervised language modeling with weighted binary cross-entropy on student-generated reasoning. The two losses are normalized over their own valid sample sets. The default reasoning-supervision weight is $\lambda = 1$. The positive-class weight in the BCE objective is computed only from the training split as N_-/N_+.

A.5 Literature Corpus Construction

The literature corpus covers ICLR, ICML, and NeurIPS papers published from 2018 to 2025. We process 24,828 papers and retain 30,361 structured optimization insights extracted from 19,050 papers.

Each insight contains the baseline configuration, candidate modification, mechanism analysis, outcome information, and provenance metadata. Literature continual pretraining uses only the serialized BASE, DIFF, mechanism, and conclusion fields. Explicit binary answers, confidence estimates, and auxiliary verification fields are excluded from the continual-pretraining text.

For each paper, the extraction process identifies controlled experimental comparisons with traceable quantitative evidence. Records are rejected if required fields are missing, numerical evidence cannot be matched to the source, the inferred conclusion conflicts with the computed outcome, or the comparison is duplicated. Automated validation checks formatting, numerical consistency, and label consistency, but does not constitute complete factual or causal verification.

A.6 Literature Continual Pretraining

Continual pretraining uses the 30,361 structured literature records with a standard next-token prediction objective. Each serialized record contains the baseline system, proposed modification, mechanism explanation, and conclusion. The maximum sequence length is 4,096 tokens. Longer records are split into token segments with one-token overlap. Continual pretraining runs for four epochs and does not directly optimize a binary outcome-prediction objective.

A.7 Literature Retrieval and Teacher Construction

We encode literature records using Qwen3-Embedding-4B. For each training example, LitRM constructs two retrieval queries: one from the joint BASE–DIFF context and another emphasizing the candidate modification. Each query retrieves the top 100 literature records. The two rankings are fused using reciprocal rank fusion with constant $c = 60$, and the top 10 fused candidates are provided to the teacher.

We use GPT-5.6 sol as the teacher model. Each request combines a literature-free or literature-assisted prompt template with the BASE–DIFF input and, when applicable, retrieved paper evidence. The teacher may use at most three retrieved records and may reject all records if none is applicable.

The teacher first generates reasoning and a judgment without access to the ground-truth training label. If the initial judgment is incorrect or the output fails the qualification checks, we reveal the ground-truth training label for one STaR-style reconstruction attempt. The teacher is instructed not to repeat the label hint or use the supplied label as causal evidence. Reconstruction revises the teacher output; the execution-derived training label remains unchanged.

Qualification requires complete formatting, a parseable answer matching the ground-truth training label, verbatim-verifiable quotations from the input and any cited paper evidence, no explicit outcome leakage or repetition of the label hint, and compliance with the language and length requirements. Only the mechanism and conclusion are retained as the reasoning text for student supervision; counterchecks, evidence-auditing fields, literature identifiers, and confidence information are excluded.

A.8 Student Rollout and Joint Training

After teacher-supervised warm-up, we freeze a snapshot of the student and generate exactly one reasoning trace for each training example without revealing the ground-truth label. All traces are retained, including incomplete or incorrect ones. The resulting token sequences and segment boundaries are cached.

These traces are not refreshed during joint training. Instead, the cached tokens are replayed while their hidden representations are recomputed by the current student model at every forward pass. This exposes the reward branch to reasoning generated by the student while allowing gradients from the reward loss to update the shared backbone and scoring module.

Student generation uses temperature 0.6, top-p 0.95, top-k 20, and a maximum of 512 newly generated tokens. One reasoning trace is sampled per example.

A.9 Inference

At inference time, LitRM does not use the teacher, literature retriever, execution outcome, or ground-truth label. Given BASE and DIFF, the student first generates one reasoning trace and then replays the resulting sequence through the shared model. The reasoning representation acts as a query over the original BASE and DIFF representations, and the resulting evidence representations are passed to the scoring head to produce a continuous reward score. All candidate ranking and ROC-AUC results are computed from this continuous score rather than the discrete answer generated in the reasoning sequence.

B Additional Experimental Results

This section provides additional experimental results complementing the analyses in Section 1 and Section 5. We report detailed OOD results, additional training dynamics and backbone comparisons, and complete ablation and sensitivity results.

B.1 Seen–OOD Generalization Details

Figure 1 summarizes the final changes in Seen and OOD performance after scenario-specific reward-model training. Colors distinguish reward-modeling methods, while marker shapes indicate model scale: circles denote 1.7B models, diamonds denote 4B models, and triangles denote 8B models. The corresponding results are reported in Table 5. For TraceML, the OOD changes in Figure 1 and Table 5 are evaluated on the HMS split. For ForeAgent, they are evaluated on its held-out OOD test set.

Table 5: Detailed results underlying Figure 1. Δ denotes the performance change after scenario-specific training relative to the corresponding base model.

Benchmark	Method	Scale	Δ Seen (pp)	Δ OOD (pp)
TraceML	Disc. RM	1.7B	+13.97	-1.59
TraceML	GenRM-Direct	1.7B	+25.94	-2.81
TraceML	GRM-SFT	1.7B	+25.63	-6.48
TraceML	CLoud	1.7B	+25.03	+0.89
ForeAgent	Disc. RM	1.7B	+24.98	-12.10
ForeAgent	GenRM-Direct	1.7B	+25.80	+0.43
ForeAgent	GRM-SFT	1.7B	+27.83	-8.68
ForeAgent	CLoud	1.7B	+23.60	-2.89
TraceML	GenRM-Direct	4B	+31.78	-2.92
TraceML	GenRM-Direct	8B	+33.83	-1.80
ForeAgent	GenRM-Direct	4B	+19.02	-3.19
ForeAgent	GenRM-Direct	8B	+23.36	-4.84

B.2 Detailed OOD Results

In addition to the OOD results reported in the main text, TraceML and RecOpt contain multiple held-out OOD subsets. Table 6 reports their performance separately. LitRM consistently improves across all evaluated OOD subsets rather than benefiting from a single held-out domain. In particular, it reaches an average OOD AUC of 66.86% on TraceML and 70.56% on RecOpt, substantially outperforming the representative baselines.

Table 6: Detailed OOD performance (AUC, %) on TraceML and RecOpt.

Method	TraceML			RecOpt		
	ICR	HMS	Avg.	LZD	Falcon	Avg.
Disc. RM	51.36	54.22	52.79	52.83	46.35	49.59
GenRM-Direct	63.20	53.00	58.10	54.50	48.29	51.40
GRM-SFT	45.65	49.33	47.49	54.21	49.47	51.84
CLoud	52.31	56.70	54.51	49.34	50.18	49.76
LitRM	**65.39**	**68.33**	**66.86**	**71.83**	**69.28**	**70.56**

B.3 Additional Training Dynamics

Figure 4 in the main text presents representative training trajectories on TraceML and ForeAgent. Table 7 additionally reports the corresponding dynamics on RecOpt. For the baselines, Seen AUC generally increases with training while OOD performance gradually deteriorates. LitRM exhibits a different pattern: its Seen AUC increases from 72.26% to 86.17%, while average OOD AUC improves from 54.34% to 70.56%. This provides additional evidence that the OOD improvement is maintained throughout optimization rather than appearing only at the final checkpoint.

Table 7: Training dynamics on RecOpt (AUC, %). OOD denotes the average over the two held-out OOD subsets.

Method	Seen				OOD Avg.			
	25%	50%	75%	100%	25%	50%	75%	100%
Disc. RM	74.03	77.86	78.71	79.07	54.05	51.11	50.91	49.59
GenRM-Direct	77.23	81.34	80.94	82.43	56.05	54.22	54.15	51.40
GRM-SFT	72.67	78.23	79.17	79.74	56.23	54.49	53.33	51.84
CLoud	69.68	78.93	79.39	81.04	54.64	52.11	51.18	49.76
LitRM	72.26	83.02	84.69	**86.17**	54.34	65.88	66.40	**70.56**

B.4 Additional Backbone Results

Section 5.2 shows that LitRM generalizes across model scales and families. We further examine whether simply increasing backbone capacity resolves the OOD generalization problem. Table 8 reports the BASE and scenario-specific tuned models on TraceML. Across all tested Qwen3 scales, tuning substantially improves Seen AUC but does not improve OOD performance. For example, scaling Qwen3 from 1.7B to 8B increases tuned Seen AUC from 81.05% to 86.46%, whereas OOD AUC remains around 50%. The Gemma backbone exhibits a similar Seen–OOD gap. These results indicate that model scaling alone is insufficient to address the observed generalization problem.

Table 8: Effect of backbone scale and family under standard scenario-specific tuning on TraceML (AUC, %). Both Base OOD and Tuned OOD report results on the HMS split.

Model	Base Seen	Tuned Seen	Base OOD	Tuned OOD
Qwen3-1.7B	55.11	81.05	55.81	53.00
Qwen3-4B	52.20	83.98	47.76	44.84
Qwen3-8B	52.63	86.46	52.43	50.63
Gemma-3-4B-PT	53.15	78.59	48.89	49.83

B.5 Complete Ablation Results

Figure 5 summarizes the progressive ablation study. Table 9 provides the complete Seen and OOD results. Teacher-CoT supervision alone increases average OOD AUC from 52.79% to 56.71%, but substantially reduces Seen performance. Using student-generated reasoning for BCE reward learning restores Seen AUC to 80.53% and further improves OOD AUC to 59.14%.

The reasoning-guided evidence reader further improves average OOD AUC by 3.29 percentage points. Literature continual pretraining further increases average OOD AUC from 62.43% to 65.67%, and grounding the teacher in retrieved literature evidence raises it to 66.86%. Overall, the complete LitRM improves average OOD AUC by 14.07 percentage points over the base discriminative RM while also improving Seen AUC from 69.08% to 82.79%.

Table 9: Complete progressive ablation on TraceML (AUC, %).

Configuration	Seen	ICR	HMS	OOD Avg.
Disc. RM	69.08	51.36	54.22	52.79
* Teacher-CoT SFT	52.08	55.42	57.99	56.71
* Self-CoT BCE	80.53	58.13	60.15	59.14
* Evidence Reader	81.91	62.73	62.13	62.43
* Literature CPT	82.08	64.70	66.64	65.67
* Literature-Grounded Teacher	**82.79**	**65.39**	**68.33**	**66.86**

B.6 Complete Sensitivity Results

We provide the complete sensitivity results corresponding to Figure 6. Overall, LitRM is relatively stable around the default hyperparameters, while overly limited evidence or reasoning budgets reduce OOD performance.

Retrieved Evidence Budget. Table 10 reports results for different numbers of retrieved literature insights. Removing retrieval ($K = 0$) reduces average OOD AUC to 65.67. Increasing K to 3 already yields a substantial improvement, and performance peaks at $K = 10$. Increasing the retrieval budget to 20 provides no additional benefit.

Reasoning-Supervision Weight. Table 11 varies the SFT weight λ. The best average OOD result is obtained at $\lambda = 1$. A smaller weight weakens reasoning supervision, while increasing it to $\lambda = 2$ also reduces OOD performance, indicating that balancing reasoning generation and execution-based reward learning is important.

Table 10: Sensitivity to the retrieved evidence budget on TraceML (AUC, %).

K	Seen	ICR	HMS	OOD Avg.
0	82.08	64.7	66.64	65.67
3	82.35	64.24	67.98	66.11
10	**82.79**	**65.39**	68.33	**66.86**
20	82.67	65.01	**68.56**	66.79

Table 11: Sensitivity to the reasoning-supervision weight on TraceML (AUC, %).

λ	Seen	ICR	HMS	OOD Avg.
0.25	**83.91**	62.26	67.29	64.78
0.5	82.27	63.48	67.82	65.65
1.0	82.79	**65.39**	**68.33**	**66.86**
2.0	82.06	64.05	65.64	64.85

Reasoning Budget and Computational Cost. Table 12 reports the performance–cost trade-off as the maximum CoT length varies. With a 64-token limit, all generated traces are truncated. A 128-token limit still truncates 99.17% of traces. Increasing the limit to 256 tokens reduces truncation to 0.30% and improves average OOD AUC to 66.86%. At 512 tokens, the average generated length remains approximately 173 tokens and GPU time remains essentially unchanged relative to the 256-token setting. We therefore use 512 tokens by default to avoid truncating unusually long traces without materially increasing average computation.

Table 12: Performance–reasoning-budget trade-off on TraceML.

CoT Limit	Avg. Len.	Trunc. (%)	Seen	ICR	HMS	OOD Avg.	GPU Time (s/sample)
			AUC (%)				
64	64.00	100.00	81.38	64.72	66.48	65.60	0.42
128	127.92	99.17	82.63	65.27	68.01	66.64	0.75
256	173.18	0.30	82.71	65.39	68.33	66.86	1.16
512	173.21	0.00	**82.79**	65.39	68.33	66.86	1.16

B.7 WITHIN-BASE RANKING ON RECOPTBENCH

We additionally evaluate candidate ranking within the same BASE on the LZD and Falcon test splits of RecOptBench. Each group contains candidates sharing the same BASE, target metric, and evaluation protocol. We use the same model checkpoints and continuous prediction scores as in the main evaluation.

Metric and eligible groups. We compute ROC-AUC separately for each group containing at least one positive and one negative example. We then aggregate the group-level AUC values using candidate-count weights:

$$\mathrm{GAUC}_{\mathrm{BASE}} = \frac{\sum_{g \in \mathcal{G}_{\pm}} n_g \, \mathrm{AUC}_g}{\sum_{g \in \mathcal{G}_{\pm}} n_g}. \tag{8}$$

The sums range over eligible BASE groups, and each group's weight is its number of candidates. Groups containing only one class are excluded because their within-group AUC is undefined. All compared models are evaluated on identical groups, candidates, and execution labels.

Evaluation coverage and results. Table 13 reports the evaluation coverage and BASE-grouped AUC separately for LZD and Falcon. Candidate coverage is the percentage of all test candidates that belong to eligible groups. We report both the number of eligible groups and their candidate coverage to make the evaluated population explicit.

Table 13: Evaluation coverage and within-BASE ranking on RecOptBench. GAUC and candidate coverage are reported as percentages. All methods use the same eligible groups within each split.

	LZD	Falcon
Evaluation coverage		
Total test candidates	3,189	6,576
Total BASE groups	1,200	2,726
Eligible BASE groups	216	474
Candidates in eligible groups	1236	2352
Candidate coverage (%)	38.76	35.77
BASE-grouped AUC (%)		
Discriminative RM	50.15	46.08
GenRM-Direct	53.54	49.76
GRM-SFT	51.31	50.35
CLoud	49.16	51.48
LitRM	61.59	60.07

Interpretation. Pooled ROC-AUC includes comparisons across different BASE instances, whereas BASE-grouped AUC evaluates the ordering of positive and negative candidates within the same BASE. The two metrics therefore characterize different aspects of reward prediction.

C More Details about Discussion

This section provides additional qualitative analyses supporting the discussion in the main paper. We examine two complementary aspects of LitRM: how literature grounding affects teacher reasoning construction, and how the resulting student behaves on unseen OOD examples. For teacher cases, we compare the teacher's first prediction with and without retrieved literature evidence under the same BASE–DIFF input. No ground-truth label or STaR-style refinement is provided before these predictions. For student cases, we compare an otherwise similar model without literature grounding against LitRM on OOD test examples. Both students generate reasoning independently at inference time without teacher or literature retrieval.

C.1 Literature-Grounded Teacher Reasoning

Case 1: Recovering Missing Temporal Information. We first consider a sample from the Google Brain Ventilator Pressure Prediction task. The BASE uses a small feed-forward MLP with BatchNorm, ReLU, and Dropout, trained directly on per-row features without recurrent or attention-based temporal modeling. The DIFF introduces cumulative `u_in` and lagged `u_in` features within each sequence and increases training from 10 to 50 epochs.

Without literature evidence, the teacher recognizes that the additional historical features may help but remains uncertain whether they are sufficient to produce a substantial improvement. It therefore predicts a negative outcome. With literature grounding, however, the teacher explicitly connects the modification to a mechanism observed in related work: explicitly encoding historical statistics and temporal changes can provide sequence information to predictors that do not model temporal dependencies directly. This leads the teacher to identify the lack of temporal context as a concrete limitation of the BASE and to predict a positive outcome.

The executed modification reduces MAE from 4.371850 to 2.311690, corresponding to a 47.12% relative improvement and a positive label. This case illustrates that literature grounding can help the teacher move from a generic "historical features may help" argument to a more specific mechanism linking the system limitation, the modification, and the expected effect. Since the DIFF also increases the number of training epochs, however, the observed gain cannot be attributed solely to the temporal features.

Case 2: Identifying a Retrieval-Recall Bottleneck. The second teacher example comes from the Learning Equality Curriculum Recommendations task. The BASE embeds topics with a pretrained Transformer and retrieves candidate educational content through an Annoy index. It considers only the top 10 candidates and applies a cosine-similarity threshold of 0.98. The DIFF increases the

candidate pool from 10 to 200, lowers the threshold to 0.97, and introduces a fallback candidate when no item passes the filter.

Both teacher variants identify the trade-off between recall and false positives. The teacher without literature evidence nevertheless predicts that the gain is unlikely to exceed the required threshold. With retrieved literature evidence, the teacher instead identifies the combination of top-10 truncation and a high similarity threshold as a concrete recall bottleneck. It further reasons that increasing candidate coverage while retaining similarity filtering can recover relevant items without indiscriminately accepting all retrieved candidates.

Execution confirms the positive prediction: FScoreBetaMicro improves from 0.234676 to 0.258823, a relative gain of 10.29%. This example shows how literature grounding can sharpen reasoning about the mechanism of a candidate modification rather than simply transferring the empirical outcome of a retrieved paper.

C.2 OOD Student Reasoning and Reward Consistency

Beyond improving teacher supervision, we examine whether literature-grounded training changes how the student reasons on unseen examples. In the following cases, both models generate their own reasoning at inference time. No external retrieval or teacher assistance is used. For the qualitative examples only, we interpret scores of at least 0.5 as positive predictions and lower scores as negative predictions. This convention does not imply calibrated success probabilities. Ground-truth labels remain determined by the criterion of at least 10% relative improvement.

Case 3: Increasing AutoML Search Budget and Ensembling. This OOD example is from the ICR tabular-classification task. The BASE uses FLAML with feature processing and a search budget of 2,200 seconds. The DIFF increases the budget to 21,600 seconds and enables ensembling.

The student without literature grounding argues that the larger search space and ensemble could plausibly yield more than a 10% improvement, but assigns a continuous score of only 0.3601. Under this decision rule, the verbal judgment and the thresholded score disagree. LitRM instead notes that a longer search budget may help, but that the benefit of ensembling depends on error diversity, validation quality, and calibration, none of which are established by the input. It consequently predicts that the change is unlikely to reach the required improvement threshold and assigns a lower score of 0.2905.

The actual WeightedMulticlassLoss deteriorates from 0.354537 to 0.599689, corresponding to a 69.15% worsening and a negative label. In this case, LitRM produces reasoning and a continuous score that are both aligned with the observed outcome.

Case 4: Diminishing Returns from Larger Ensembles. The final example also comes from the ICR OOD split. The BASE already combines XGBoost and TabPFN through stacking, multiple random seeds, and fold averaging. The DIFF increases the number of TabPFN ensemble configurations from 12 to 25 and 50 without changing the underlying data or features.

Both students reason that simply adding more ensemble members may provide limited benefit. Under the same decision rule, the baseline score of 0.7073 gives a positive prediction despite its negative verbal judgment. LitRM instead emphasizes the likely correlation among additional ensemble members trained on the same data and features, as well as diminishing variance reduction once substantial ensembling is already present. Its negative reasoning is accompanied by a score of 0.2909.

The observed loss is unchanged at 1.082015, corresponding to 0% improvement and therefore a negative label. Here, the main difference is not the qualitative conclusion itself but the consistency between reasoning, continuous scoring, and the eventual outcome.

C.3 Summary of Qualitative Findings

These cases highlight two complementary effects of literature grounding. First, retrieved scientific knowledge can help the teacher identify mechanism-level bottlenecks, such as missing temporal information or retrieval truncation, and construct more targeted supervision. Second, after training, the student can apply similar mechanism-oriented reasoning to unseen examples without accessing

the literature at inference time. In the OOD cases above, LitRM also exhibits better agreement between generated reasoning and the continuous reward score, which is important because candidate ranking is ultimately based on the score rather than the generated textual judgment.

The examples should nevertheless be interpreted as qualitative evidence rather than causal proof. Retrieved papers differ from the target tasks, and multiple changes may occur within a single DIFF. Literature grounding therefore does not establish that a specific mechanism caused the observed improvement. Instead, these cases illustrate how external evidence can support more explicit and context-sensitive reasoning about why a candidate modification may or may not work.