BEYOND CORRECTNESS: LEARNING ROBUST REASONING VIA TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) has recently strengthened LLM reasoning, but its focus on final-answer correctness leaves a critical gap: equally correct solutions can differ in the usefulness of their intermediate reasoning. We adopt a simple philosophical view—good reasoning should remain useful beyond the mind that produced it—and treat reasoning as a form of meaning transfer that can be assessed through truncation and continuation by another model. Building on this principle, we introduce **R**einforcement **L**earning with **T**ransferable **R**eward (**RLTR**), which rewards reasoning prefixes that support correct continuation by a separate model. This objective turns the usefulness of intermediate reasoning to another model into an auxiliary training signal for improving the generator itself. Our approach improves majority-vote accuracy while improving final-answer accuracy, and it reaches comparable performance in substantially fewer training steps. For example, RLTR gains a +5.8%p in Maj@64 on AMC23 compared to RLVR, and RLTR matches RLVR's average accuracy with roughly $2.5\times$ fewer training steps on MATH-500. Extensive analyses further support RLTR's effectiveness, showing improved prefix utility for held-out receivers.

1 INTRODUCTION

Large language models (LLMs) combined with reinforcement learning (RL) have recently demonstrated remarkable performance across various domains, such as solving complex mathematical problems (Shao et al., 2024), serving as coding assistants (Le et al., 2022), and contributing to scientific discoveries (Romera-Paredes et al., 2024). Compared with simple supervised fine-tuning, RL enables LLMs to learn more generalizable and robust behaviors (Ouyang et al., 2022). However, the success of RL in LLMs is often bottlenecked by its heavy reliance on reward models. While reward models reduce the need for costly human annotations, they introduce critical challenges such as reward hacking (Skalse et al., 2022) and increased training complexity (Casper et al., 2023).

To tackle this issue, reinforcement learning with verifiable reward (RLVR) has recently emerged (Shao et al., 2024; Guo et al., 2025). RLVR replaces learned reward models with the correctness of final answers, which significantly simplifies the training pipeline and alleviates reward hacking and the need for extensive human annotations. Despite its success on complex reasoning tasks, particularly competitive mathematics, RLVR provides no direct incentive for intermediate reasoning to be robust or reusable across different sampled solutions. Prior work has observed that the model loses consistency and diversity as more samples are generated, suggesting that final-outcome correctness alone may be an incomplete training signal for the reasoning process (Yue et al., 2025).

Our approach. To move beyond outcome-only reward, we draw inspiration from human learning: strong reasoning is not merely correct—it is also *explainable* in a way that others can reliably follow. We therefore ask whether rewarding reasoning that helps another solver can improve the generator's own reasoning. This motivates our central concept, *reasoning transferability*, defined as the extent to which a reasoning prefix produced by one model can be completed by another model to reach a verifiably correct answer. Building on this principle, we propose **R**einforcement **L**earning with **T**ransferable **R**eward (**RLTR**), a new reinforcement learning framework that explicitly optimizes cross-model reasoning transferability. The core mechanism is a *transfer reward*, determined by whether a separate receiver reaches a correct final answer when continuing a truncated reasoning trace from the generator.

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.

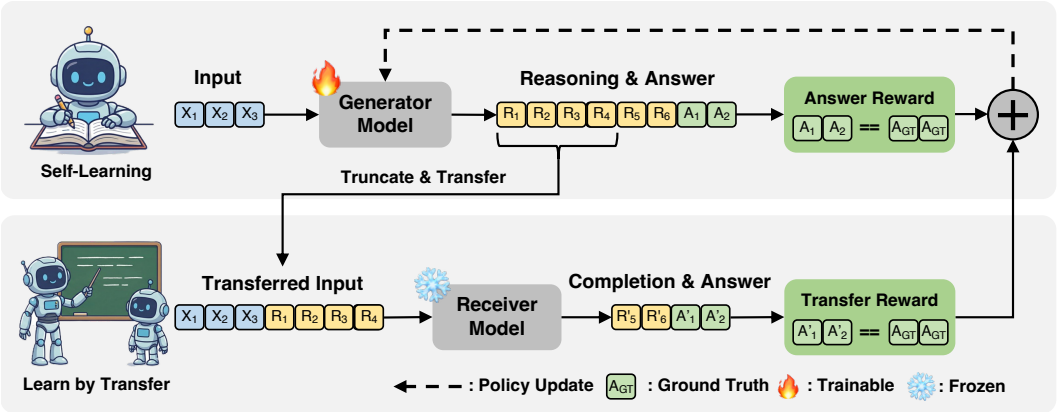


Figure 1: **Overview of RLTR:** RLTR augments RLVR with a transfer reward. **Top:** A trainable generator model produces a full completion, whose final-answer correctness yields an answer reward. **Bottom:** We then truncate the generated reasoning to form a prefix and feed it to a frozen receiver model to produce continued completions whose final-answer correctness defines a transfer reward that measures the transferability of partial reasoning across models. The answer reward and transfer reward are combined into a unified reward signal used to update the generator policy.

We demonstrate that rewarding cross-model reasoning transferability improves the generator's own performance through extensive evaluations across mathematical and scientific reasoning tasks. Overall, RLTR consistently improves Majority Voting at K (Maj@K), which evaluates performance under multi-sample answer aggregation, while also improving average accuracy. For instance, on AMC23 (MAA, 2023), RLTR improves Maj@64 from 61.7 to 67.5 compared to RLVR. Across training, RLTR achieves the same average accuracy level with about 2.5× fewer training steps than RLVR on MATH-500 (as shown in Figure 3), indicating faster learning and improved sample efficiency. Experiments with stronger language models and vision-language models show that RLTR's benefits extend beyond the main training setup. We further compare RLTR with PRM-based supervision, self-continuation, and an MRT-based progress baseline under matched training and evaluation settings. On MATH-500, these alternatives achieve comparable average accuracy, yet RLTR attains higher majority-vote performance, showing that the benefits of the transfer reward extend beyond average accuracy. Beyond task performance, we also evaluate whether the RLTR-generated prefixes remain useful to unseen receivers. In these unseen evaluations, RLTR prefixes consistently outperform RLVR prefixes across receivers and truncation ratios. This suggests that RLTR encourages reusable intermediate reasoning that goes beyond the model.

We highlight the main contributions of the paper below:

- We introduce RLTR, a simple extension of RLVR that adds a transfer reward to optimize reasoning transferability across models.
- We demonstrate that RLTR advances performance in both average accuracy and majority voting accuracy, and improves training sample efficiency; e.g., RLTR matches RLVR's performance with 2.5× fewer training steps on MATH-500.
- We demonstrate RLTR's applicability to stronger language models and vision-language models, and show that its reasoning prefixes remain useful to decoders not used during training.

2 RLTR: Reinforcement Learning with Transferable Reward

In this section, we introduce **R**einforcement **L**earning with **T**ransferable **R**eward (**RLTR**), a reinforcement learning framework that encourages robust and reusable reasoning traces by augmenting RLVR with a transfer reward. Building on RLVR-style final answer supervision, RLTR introduces a transfer reward that evaluates whether another model (namely, the receiver model) can recover the correct answer from a truncated reasoning prefix, and maximizes this signal as an auxiliary reward objective. We first formalize the learning framework, RLVR, in Section 2.1. We then present RLTR, the core training algorithm that integrates the transfer reward into the RL objective, in Section 2.2. An overview of RLTR is shown in Figure 1.

2.1 PRELIMINARY: RLVR

RLVR optimizes the model to produce a correct final answer under a fixed output format. Given an input x, the generator model $\mathcal{M}_{\text{gen}}$ samples a full response (including reasoning and the final answer), i.e., $y_{\text{gen}} \sim \mathcal{M}_{\text{gen}}(\cdot|x)$. The reward is primarily defined by an answer reward that checks whether the generator's final answer matches the ground truth, i.e., $\mathrm{R}_{\text{ans}}(y_{\text{gen}}) = \mathbb{1}[\text{answer}(y_{\text{gen}}) = y_{\text{gt}}]$. In addition, a format reward $\mathrm{R}_{\text{fmt}}(y_{\text{gen}})$ is used to encourage well-formed outputs (e.g., valid reasoning structure and answer formatting), and to penalize invalid generations.

The overall reward is defined as

$$\mathrm{R}_{\text{RLVR}}(y_{\text{gen}}) = a\,\mathrm{R}_{\text{ans}}(y_{\text{gen}}) + \mathrm{R}_{\text{fmt}}(y_{\text{gen}}). \tag{1}$$

For a given dataset $\mathcal{D} := \{(x, y)\}$, the generator is optimized via reinforcement learning as:

$$\max_{\mathcal{M}_{\text{gen}}} \mathbb{E}_{(x,y_{\text{gt}})\sim\mathcal{D}} \left[\mathbb{E}_{y_{\text{gen}}\sim\mathcal{M}_{\text{gen}}(\cdot|x)} \left[\mathrm{R}_{\text{RLVR}}(y_{\text{gen}})\right]\right]. \tag{2}$$

In this paper, we mainly optimize the RL policy with group relative policy optimization (GRPO; Shao et al., 2024; Guo et al., 2025), which is known to be simple yet strong when used with RLVR.

Overall, RLVR optimizes the correctness of the generator's own final output. However, it does not explicitly account for step-level noise and brittleness in the reasoning process, nor does it directly encourage reusable or interpretable reasoning that can be reliably continued by other models.

2.2 REINFORCEMENT LEARNING WITH TRANSFERABLE REWARD

To address this limitation, we propose RLTR, a training method that augments RLVR with a transfer reward to encourage reasoning *transferable*, whether partial reasoning that can be reliably reused by another model to reach the correct answer. Intuitively, a generator has higher transferability when its truncated reasoning prefixes contain stable and informative intermediate structure, enabling a receiver to reliably complete the solution, rather than relying on the generator's own continuation.

Transfer Reward. We define a transfer reward that identifies the reasoning transferability. Concretely, given an input x, the generator model $\mathcal{M}_{\text{gen}}$ produces a full reasoning sequence $y_{\text{gen}} \sim \mathcal{M}_{\text{gen}}(\cdot|x)$. Then, we truncate the prefix of the reasoning sequence and passed to a receiver model $\mathcal{M}_{\text{rcv}}$ for continuation, $y_{\text{rcv}} \sim \mathcal{M}_{\text{rcv}}(\cdot|x, y_{\text{gen}}[:\ell])$, where the truncation ratio $\tau \in (0, 1]$ and the truncation length $\ell = \lfloor \tau\,(|y_{\text{gen}}| - |a_{\text{gen}}|)\rfloor$. We define a transfer reward as:

$$\mathrm{R}_{\text{trans}}(y_{\text{gen}}, \mathcal{M}_{\text{rcv}}) = \mathbb{1}[\text{answer}(y_{\text{rcv}}) = y_{\text{gt}}] \tag{3}$$

which indicates whether the receiver can reach the correct answer from the truncated reasoning. In practice, we compute $\mathrm{R}_{\text{trans}}$ using a single continuation, and sample τ uniformly between 0.3 and 0.9 to ensure the model learns a reasoning pattern robust to arbitrary interruptions, rather than overfitting to fixed truncation boundaries.

The overall reward is defined as

$$\mathrm{R}_{\text{RLTR}}(y_{\text{gen}}, \mathcal{M}_{\text{rcv}}) := \mathrm{R}_{\text{RLVR}}(y_{\text{gen}}) + t\,\mathrm{R}_{\text{trans}}(y_{\text{gen}}, \mathcal{M}_{\text{rcv}}) \tag{4}$$

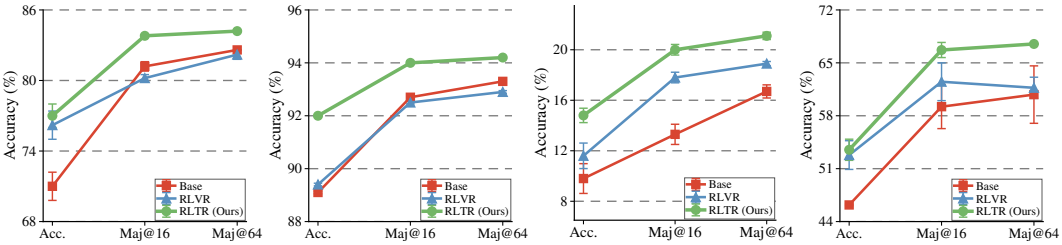
Training Objective. The generator is optimized via reinforcement learning with the objective

$$\max_{\mathcal{M}_{\text{gen}}} \mathbb{E}_{(x,y_{\text{gt}})\sim\mathcal{D}} \left[\mathbb{E}_{y_{\text{gen}}\sim\mathcal{M}_{\text{gen}}(\cdot|x)} \left[\mathrm{R}_{\text{RLTR}}(y_{\text{gen}}, \mathcal{M}_{\text{rcv}})\right]\right]. \tag{5}$$

3 EXPERIMENTS

We provide an empirical evaluation of RLTR by investigating the following questions:

- **RQ1**: Can RLTR improve accuracy and majority-vote accuracy? (Figure 2, Table 3a, and Table 6)
- **RQ2**: How does RLTR affect training dynamics and efficiency? (Figure 3 and Table 2)
- **RQ3:** Does RLTR improve prefix utility for unseen receivers? (Figure 5b, and Table 9)
- **RQ4**: How do the proposed components contribute to performance? (Figure 4)



(a) MATH-500 (b) GSM8K (c) AIME2024 (d) AMC23

Figure 2: **Performance of RLTR.** Average accuracy (%) and Maj@K (%) of RLTR (Ours) compared with baselines, including the base model and RLVR-trained model. Error-bar indicates standard deviation. We evaluate on two medium-difficulty math reasoning benchmarks, MATH-500 and GSM8K, as well as two more challenging benchmarks, AIME2024 and AMC23. RLTR consistently outperforms other baselines. **Notably, the performance gains of RLTR are substantially larger on the more challenging benchmarks.**

Evaluation setup. We mainly report majority voting at K (Maj@K) and average accuracy (Acc.). In particular, Maj@K selects the final answer as the most frequent among the K samples, measuring performance under multi-sample answer aggregation. Additionally, we measure transferability as the average accuracy under a cross-model reasoning prefix continuation. We assess cross-model prefix utility separately using receiver continuation accuracy. For each model and benchmark, we repeat evaluation three times different random seeds and report the average. We evaluate our method on MATH-500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), AMC23 (MAA, 2023), and AIME2024 (MAA, 2024), which are widely used mathematical benchmarks, and additionally include the challenging scientific reasoning benchmark GPQA (Rein et al., 2024). During the evaluation, we generate with a consistent sampling hyperparameter, temperature $T = 1.0$. We report the specified evaluation setup details in Appendix A.4.

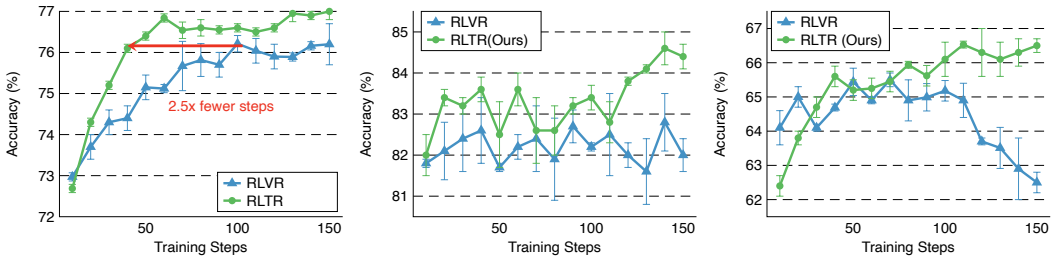
Training setup. For the main experiment, we trained RLTR on Qwen2.5-7B-Instruct (Yang et al., 2025b) checkpoint, which is provided in HuggingFace. Unless stated otherwise, we use Qwen2.5-3B-Instruct (Yang et al., 2025b) as the receiver model $\mathcal{M}_{\text{rcv}}$. For analysis, we additionally consider alternative receivers (e.g., Qwen2.5-1.5B-Instruct (Yang et al., 2025b) and Llama3.2-3B-Instruct (Grattafiori et al., 2024)). We perform RL training on a 3K subset of the MATH (Hendrycks et al., 2021) training set, restricted to problems with difficulty ≥ 3 following SimpleRL Zeng et al. (2025). We trained RLTR and baseline (i.e., RLVR) with three different seeds for each and report mean, standard deviation in Figure 2. For the rest of experiments we choose the best checkpoint by selecting the checkpoint shows the best validation accuracy. For training RLTR, we adopt reward weight $a = 0.1$ and $t = 1.0$ from Eq. 1 and Eq. 4. Additional implementation details are provided in Appendix A.3.

Baseline. We compare RLTR with the base model (the initial checkpoint before RL) and RLVR, which applies conventional reinforcement learning with verifiable reward by using the same generator initialization, training data, and decoding/evaluation protocol with RLTR. We mostly followed the baseline training setup from SimpleRL (Zeng et al., 2025). For deeper analysis, we trained RLTR-self, MRT-based, and PRM-trained model. They are also trained with same train/validation dataset, and base-framework. The details for each analysis is described in Appendix B

3.1 Performance of RLTR

We present the main result by comparing the performance on math benchmarks. Here, we mainly evaluate RLTR against RLVR and the base model, reporting average accuracy and Majority at K to assess performance and consistency. The main results were evaluated in (i) two moderate-difficulty tasks: MATH-500 and GSM8K, and (ii) two challenging tasks: AIME2024 and AMC23.

RLTR performance gain on moderate-difficulty tasks. As shown in Figure 2, overall, RLTR consistently outperforms RLVR in both moderate-difficulty benchmarks (i.e., MATH-500 (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021)). It is worth noting that while RLVR improves single-sample accuracy, its gains do not consistently translate into stronger Maj@K at larger sampling budgets, consistent with narrowing of the output distribution under RLVR reported by Yue et al. (2025). In contrast, RLTR yields stronger high-K majority-vote performance on both benchmarks, suggesting that its gains extend beyond average accuracy and enhance reasoning.



(a) **Average accuracy** (b) **Maj@64 accuracy** (c) **Transferability**

Figure 3: **RLTR improves training efficiency and majority-vote accuracy.** We investigate how our method RLTR affects model performance as training steps grow compared to RLVR. Error-bar indicates standard deviation. (a) Average accuracy: RLTR, significantly reduce the training steps ($2.5\times$) to achieve comparable with RLVR. (b) Maj@64: RLTR progressively increasing Maj@64, whereas RLVR's performance degrades as training proceeds. (c) Transferability: RLTR maintains stronger receiver continuation accuracy at later checkpoints, alongside higher Maj@64.

RLTR performance gain on harder tasks. As shown in Figure 2, RLTR also outperforms baselines in challenging math benchmarks (i.e., AIME2024 (MAA, 2024) and AMC23 (MAA, 2023)), indicating that the gains are not limited to the moderate-difficulty math domain benchmarks. For example, on AIME2024, RLTR improves +3.2%p ($11.6 \rightarrow 14.8$) in average accuracy and +2.2%p ($18.9 \rightarrow 21.1$) in Maj@64 against RLVR. Notably, the improvements are more pronounced on the harder math benchmarks, where reliable progress requires maintaining coherent intermediate reasoning rather than only reaching a correct final answer. Overall, these results suggest that optimizing the transfer reward encourages more robust and reusable reasoning trajectories that generalize better under distribution shift, with particularly strong benefits on harder reasoning tasks.

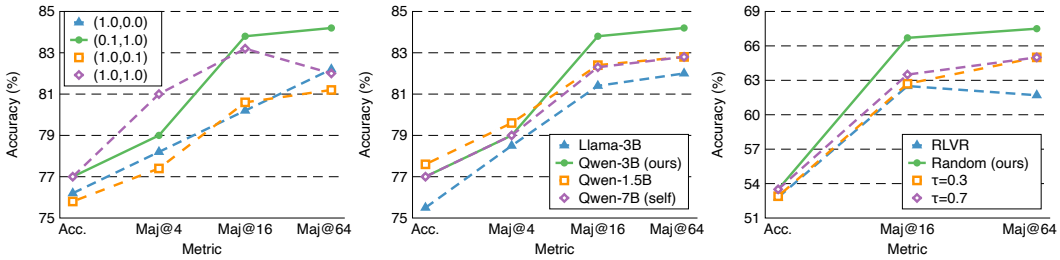
Across all four benchmarks, RLTR improves performance on one in-distribution benchmark (i.e., MATH-500) and three out-of-distribution benchmarks (GSM8K, AIME2024, and AMC23). These consistent improvements on an out-of-distribution benchmark suggest that the transfer reward provides a general process-level supervision signal, improving robustness beyond the training distribution rather than overfitting to training dataset-specific patterns.

3.2 Training Dynamics of RLTR

To better understand the effect of the proposed transfer reward, we analyze the training dynamics of RLTR using the following three metrics: (i) average accuracy, (ii) Maj@64 accuracy, and (iii) reasoning transferability. Here, we evaluate intermediate checkpoints on MATH-500 and compare RLTR with RLVR in Figure 3.

RLTR improves sample efficiency. As shown in Figures 3a, RLTR improves substantially faster ($2.5\times$) than RLVR in average accuracy, and also attains a higher final accuracy, consistent with the overall results in Figure 2. This result suggests that the transfer reward provides a richer process-level signal, helping the model converge faster than RLVR. Moreover, as shown in Figure 3b, the Maj@64 curve also follows this improvement, indicating that the gains extend to high-k majority-vote accuracy rather than average accuracy alone. Together with the compute-performance analysis in Section 3.4 and Table 2, these results indicate that RLTR achieves a better training-efficiency trade-off even when accounting for the additional cost of computing the transfer reward.

Relationship between transferability and Maj@K. We next study how reasoning transferability evolves over training by truncating a generator's reasoning trace into a prefix with a fixed ratio $\tau = 0.7$, and continually generating a separate, frozen receiver model to complete it. We measure transferability as the continued completion of final-answer accuracy. Across training, RLTR improves transferability together with Maj@64, whereas RLVR exhibits a late-stage drop in transferability that coincides with weaker Maj@64 despite relatively stable average accuracy, another instance of this narrowing (Yue et al., 2025). These observations show that improvements in average accuracy alone do not translate into better majority-vote performance or more useful prefixes for receiver continuation. Furthermore, additional results for performance in various truncation ratios, including $\tau \in \{0.3, 0.5, 0.7, 0.9\}$, are illustrated in Appendix B.5.



(a) Reward weights (a, t) (b) Training receiver (c) Truncation strategy

Figure 4: **Component ablations of RLTR.** Effects of **(a)** reward weights (a, t) and **(b)** the receiver on MATH-500, and **(c)** truncation ratio on AMC23. Acc. denotes average accuracy, and Maj@K denotes majority-vote accuracy over K samples. Solid lines indicate the selected configurations.

3.3 Component Ablations

In this section, we analyze the contribution of key design choices in RLTR. Specifically, we ablate three factors: (i) the reward composition, (ii) the receiver model, and (iii) the truncation strategy.

Effect of the reward composition. To study the contribution of the transfer reward relative to the answer reward, we vary the weights (a, t) in Eq. 1 and Eq. 4, where a scales the final-answer correctness reward and t scales the transferability reward. As shown in Figure 4a, increasing the transfer reward weight consistently improves performance in high-K regimes. In contrast, when the transfer reward is down-weighted $(t{=}0.1)$, the results closely resemble RLVR and exhibit limited scaling with K. Assigning a larger ratio for transfer weight yields much stronger improvements: $(a{=}0.1, t{=}1.0)$. This supports our central hypothesis that a transferable intermediate-reasoning objective leads to more stable consensus under majority voting.

Effect of receiver model. To isolate the effect of the receiver type, we vary the receiver while keeping the generator initialization and training data fixed. Specifically, we compare RLTR trained with four different receivers (i.e., Llama3.2-3B-Instruct (Llama-3B) (Grattafiori et al., 2024), Qwen2.5-{1.5,3,7}B-Instruct (Yang et al., 2025b). As shown in Figure 4b, the receiver choice leads to noticeable differences, especially for majority voting at larger K. For Maj@16 and Maj@64, Qwen-3B yields the best results (83.8 and 84.2), whereas Qwen-1.5B achieves the best Maj@4 (79.6). These results indicate that the benefits of the transfer reward are not dependent on a large receiver, although stronger cross-model receivers further improve high-K consistency. We additionally compare against *self-continuation*, where the 7B generator serves as its own receiver. Self-continuation provides only marginal gains over RLVR on harder out-of-distribution benchmarks. On AIME2024, it even underperforms RLVR at Maj@64 ($18.9 \rightarrow 16.7$). In contrast, a separate 3B receiver yields consistent improvements across all four benchmarks (e.g., AIME2024 Maj@64: 21.1; AMC23 Maj@64: 67.5). This indicates that the transfer reward's benefit stems specifically from the cross-model signal, not from the truncate-and-continue mechanism alone (full results in Appendix B.6).

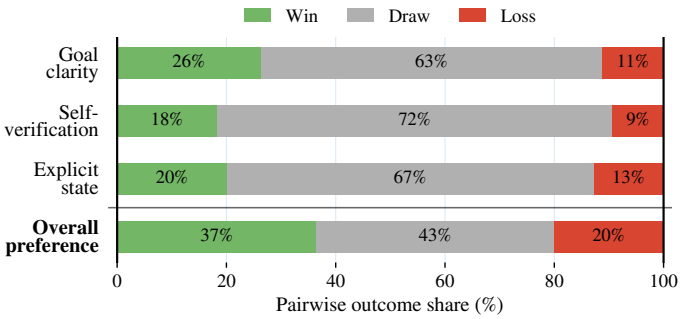
Effect of truncation strategy. During training, we sample the truncation ratio uniformly from [0.3, 0.9] to avoid overfitting to a single truncation boundary. To validate this design choice, we compare random truncation against fixed-ratio training As shown in Figure 4c, random truncation achieves the strongest overall performance, with the largest gains appearing on the high-K majority metrics. For example, on AMC23, it improves Maj@64 by 2.5 points over the best fixed ratio. This suggests that exposing the model to diverse interruption points during training leads to more transferable reasoning prefixes, which in turn encourages better reasoning behavior.

3.4 Additional Analysis

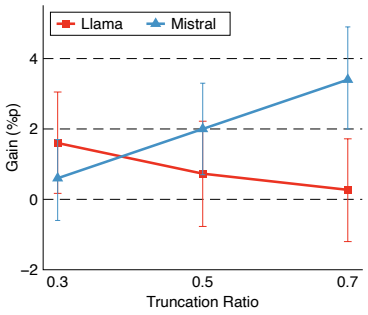
Scaling base model. We further experiment RLTR using a Qwen3-8B (Yang et al., 2025a) generator and a frozen Qwen3-4B (Yang et al., 2025a) receiver. Both RLTR and the RLVR baseline use the Polaris training data (An et al., 2025) with matched initialization, data, training budget, and evaluation protocol. As shown in Table 1, RLTR improves Maj@32 over RLVR by 4.4%p and 3.1%p on AIME2024 and AIME2025, respectively, while also improving average accuracy. This advantage indicates RLTR can effectively improve the high-K performance in a stronger model.

Table 1: **Scaling RLTR to Qwen3.** Results with a Qwen3-8B generator and frozen Qwen3-4B receiver. Both RL methods use the same Polaris training data with matched initialization, training budget, and evaluation protocol. We report mean accuracy (%) and standard deviation.

	AIME2024		AIME2025	
Method	Acc.	Maj@32	Acc.	Maj@32
Qwen3-8B Base	74.8 ± 1.3	80.6 ± 1.1	66.8 ± 1.1	77.8 ± 1.2
RLVR	75.1 ± 1.2	78.9 ± 1.2	67.8 ± 0.5	75.6 ± 0.8
RLTR	**76.3** ± 2.0	**83.3** ± 1.6	**68.3** ± 0.8	**78.7** ± 0.7



(a) RLTR (ours) vs RLVR: Reasoning-quality judgments



(b) Unseen model prefix continuation

Figure 5: **Reasoning quality and held-out prefix utility. (a)** LLM-judge outcomes comparing RLTR with RLVR; wins favor RLTR. **(b)** Receiver accuracy gains from RLTR prefixes over RLVR prefixes on MATH-500 for two receivers not used during training. Error bars show 95% confidence intervals.

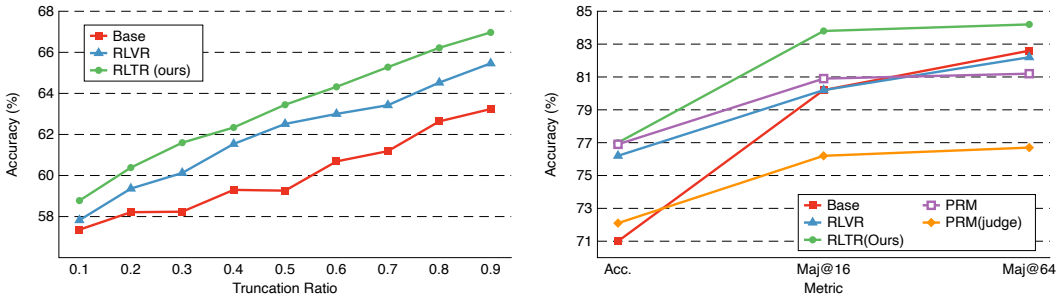
Reasoning process analysis. To examine how RLTR changes the reasoning process, we compare traces from the two systems using LLM judges that evaluate the reasoning without access to the reference answer (Appendix A.6). The judges assess three properties, *explicit state*, *goal clarity*, and *self-verification*, through pairwise forced-choice comparisons as a manipulation check for the property directly targeted by RLTR. As shown in Figure 5a, the judges prefer RLTR on all three reasoning-quality dimensions. The same qualitative pattern appears across Claude Opus 5 (Anthropic, 2026), GPT-5.6-sol (OpenAI, 2026), and DeepSeek V4-Pro (DeepSeek-AI, 2026), spanning three different model families, with all three judges favoring RLTR. These results suggest that, under the present evaluation setting, RLTR primarily changes how reasoning is structured and communicated, making intermediate goals clearer and encouraging more explicit checking.

Prefix utility for held-out receivers. We compare RLTR and RLVR prefixes on MATH-500 using unseen receivers (i.e., Llama and Mistral). Figure 5b shows gains for both unseen receivers, most pronounced at shorter prefixes for Llama and longer prefixes for Mistral. These results suggest that RLTR improves prefix utility beyond the receiver used to define its training reward, although the benefits depend on the receiver and truncation ratio. Additionally, controls with the held-out Llama receiver (Appendix B.3) further show that problem-aligned RLTR prefixes help beyond question-only solving, whereas shuffled prefixes do not reproduce this benefit. The RLTR–RLVR advantage also persists after answer masking for short and medium prefixes, providing evidence that these gains are not explained solely by the explicit answer spans removed by the masking procedure.

Computational cost analysis. One possible concern of RLTR is the extra computation introduced by the receiver (i.e., an additional model) during training. To this end, we demonstrate that, even with the extra computation, RLTR still (i) achieves the same performance with less computation, and (ii) outperforms RLVR as the computational budget increases. Here, we use FLOPs as the measure of computational cost, with the detailed computation procedure provided in Appendix B.2. As shown in Table 2, RLTR increases the training FLOPs by approximately 7% under the same number of training steps, while achieving substantially better performance. More importantly, we observed that RLVR requires 2.3× higher computational cost to reach the same performance level as RLTR (39.76→92.75 EFLOPs), highlighting both the sample

Table 2: **Computational cost.** Comparison with EFLOPs (i.e., 10^{18} FLOPs), wall-clock time (h), and Average accuracy (Acc.) on MATH-500.

Method	Steps	EFLOPs ↓	Time (h) ↓	Acc. (%) ↑
RLVR	40	37.10	1.9	74.5
RLTR	40	39.76	2.4	76.2
RLVR	100	92.75	4.8	76.2
RLTR	60	59.64	3.6	76.8



(a) Transferability across truncation ratios (b) Alternative reward baselines

Figure 6: **Analysis of the transfer reward. (a)** Receiver accuracy across retained-prefix ratios for the base model, RLVR, and RLTR. **(b)** Comparison with a PRM trained models.

Table 3: **Generalization of RLTR beyond the main training setup. (a)** Results with a Llama-3.1-8B generator on MATH-500. **(b)** Results on multimodal reasoning benchmarks MMK12 and MathVista.

(a) Llama-3.1-8B

	MATH-500			
Method	Acc.	Maj@4	Maj@16	Maj@64
Base Model	24.8	34.4	51.2	60.8
RLVR	50.0	55.2	58.6	59.8
RLTR	**50.4**	**56.8**	**62.3**	**64.2**

(b) Vision-language reasoning

	MMK12		MathVista	
Method	Acc.	Maj@16	Acc.	Maj@16
Base	45.7	58.6	48.5	68.1
RLVR	55.3	61.2	63.3	72.8
RLTR	**56.7**	**64.4**	**63.6**	**73.6**

efficiency and computational efficiency of RLTR. Lastly, in terms of single-sample accuracy, RLTR not only outperforms the best accuracy achieved by RLVR (at 92.75 EFLOPs), but does so with fewer training steps and lower computational cost. We additionally report wall-clock overhead. The mean training-step duration is 25.2% higher per step. Nevertheless, RLTR requires fewer times to reach comparable accuracy (x2 times faster in wall-clock time).

Comparison with judge- and PRM-based baselines. We additionally compare RLTR with a 3B LLM-as-a-judge baseline and a process reward model (PRM) baseline (Figure 6b). RLTR performs comparably to the PRM baseline, while achieving the highest Maj@16 and Maj@64 among the evaluated methods. For example, relative to the PRM baseline, RLTR improves Maj@16 and Maj@64 by 2.9 and 3.0 percentage points, respectively. The 3B judge-based baseline performs below the base model on all three metrics. These results suggest that the RLTR's process supervision outperform other PRM baselines. Baseline configurations and evaluation details are provided in Appendix A.5.

Accuracy under reasoning prefix continuation. We evaluate reasoning prefix utility by measuring the transfer accuracy (i.e., transferability from the generator to Qwen2.5-3B-Instruct) of traces with a truncated prefix at various ratios on MATH-500. As shown in Figure 6a, RLTR consistently outperforms baselines (i.e., Base model and RLVR) across the entire spectrum τ from 0.1 to 0.9. This demonstrates that RLTR generates a more informative reasoning prefix under any truncation ratio, rather than exploiting specific truncation boundaries. Furthermore, it is worth noting that RLTR maintains an advantage even at small truncation ratios (e.g., $\tau \in [0.1, 0.3]$), suggesting that it produces more informative prefixes from the very early stages of reasoning that provide sufficient intermediate structure for the receiver to complete the solution.

Results on different LLM. We also demonstrate the effectiveness of RLTR on another LLM/VLM family. To this end, we train RLTR from Llama3.1-8B-Instruct, where we use the receiver model as Llama3.2-3B-Instruct on the MATH-train dataset. As shown in Table 3a, RLTR outperforms the baselines at high-K Maj@K even when trained with a different LLM family. For example, RLTR achieves Maj@16 of 62.3 and Maj@64 of 64.2 on MATH500, substantially higher than both the base model and the RLVR-trained model. In contrast, RLVR primarily improves single-sample accuracy; however, these gains do not translate into sampling consistency, as performance saturates under high-K majority voting and even underperforms the base model at Maj@64. These results are consistent with the main findings in Section 3.1.

Extension to vision-language models. To examine the applicability of RLTR beyond text-only reasoning, we apply RLTR to a vision-language model (i.e., Qwen2.5-VL-7B/3B (Bai et al., 2025)) trained with visual mathematics data. We evaluate RLVR and RLTR on MMK12 (Meng et al., 2025) and MathVista (Lu et al., 2024), reporting average accuracy and majority-vote accuracy (Maj@16). As shown in Table 3b, RLTR consistently outperforms RLVR on both metrics for both benchmarks. For example, on MMK12, accuracy increases from 55.33% to 56.69% and Maj@16 from 61.2% to 64.4%. These results show that RLTR's benefits extend to the vision-language setting.

4 Related Work

Reinforcement Learning in LLM. Reinforcement learning (RL) has become a cornerstone for aligning LLMs with human intent, primarily through Reinforcement Learning from Human Feedback (RLHF; Ouyang et al., 2022). Standard approaches typically employ PPO Schulman et al., 2017 to optimize a policy against a learned reward model (RM) that mimics human preferences. However, as the policy optimizer exploits the reward model, it often leads to reward hacking, where the model generates high-reward but low-quality or nonsensical outputs due to the imperfections of the proxy reward (Skalse et al., 2022; Gao et al., 2023; Saito et al., 2023). Furthermore, training a reward model requires extensive human annotation and often creates a bottleneck in scaling reasoning.

Reinforcement Learning with Verifiable Reward. To mitigate the limitations of neural reward models, recent research has shifted towards RL with Verifiable Reward (RLVR), where the reward is derived from ground-truth correctness (e.g., mathematical answers or code execution) rather than a learned proxy. Pioneering works such as DeepSeek-Math (Shao et al., 2024) and DeepSeek-R1 (Guo et al., 2025) introduced Group Relative Policy Optimization (GRPO) to stabilize RL training without a critic model, significantly boosting performance on complex benchmarks. The following works mainly focus on stabilizing the learning algorithm and enhancing the learning objectives (Yu et al., 2025; Liu et al., 2025). While these methods are effective, they primarily evaluate the result rather than the process. Unlike prior approaches that rely solely on self-consistency or final-answer correctness, RLTR introduces a novel reward signal based on *reasoning transferability*.

Process Reward Models. Process reward models (PRMs) learn to assess intermediate reasoning steps using human or automatically constructed supervision (Uesato et al., 2022; Li & Li, 2024; She et al., 2025; Khalifa et al., 2025). These learned verifiers provide process-level signals that complement final-answer correctness. Also, PAV (Setlur et al., 2025) measures progress under a prover policy distinct from the base policy, but distills the prover's rollouts on its own traces into a learned verifier that stays fixed during RL. In contrast, RLTR trains no verifier: its auxiliary signal is the verified outcome of a frozen receiver continuing the generator's current prefixes during training. Under a common RLVR training setup, RLTR outperforms PRM baselines (Figure 6b). Continuation-based methods such as VinePPO (Kazemnejad et al., 2025) and MRT (Qu et al., 2025) instead estimate values or progress from multiple continuations at intermediate states, in practice sampled from the policy itself, and use them for step- or episode-level credit assignment. However, RLTR differs in three respects: (i) the continuation comes from a different frozen model rather than from the policy or a copy of it, so the reward measures whether another model can use the prefix, (ii) RLTR needs no step segmentation but uses a random truncation ratio, and (iii) we evaluate prefix utility on receivers never used in training (Figure 5b). Against a self-continuation variant (RLTR-self) and an MRT-based progress baseline under matched settings, RLTR attains the highest Maj@16 on all four benchmarks and the highest or tied-highest Maj@64 (Table 11).

5 Conclusion

We introduced **R**einforcement **L**earning with **T**ransferable **R**eward (**RLTR**), a transfer-augmented RLVR framework that optimizes *reasoning transferability*, the ability of a truncated reasoning prefix to be completed by another model to a verifiably correct answer. By leveraging a transfer reward, RLTR encourages reasoning traces that remain reusable under cross-model continuation. Experimental results show that RLTR improves both average accuracy and majority-vote accuracy, as well as training sample efficiency. Moreover, held-out continuation evaluations show that RLTR prefixes remain useful beyond the receiver used during training. Together, these findings support cross-model continuation success as a practical auxiliary signal for improving reasoning-task performance.

AI USE STATEMENT

In this work, we use a generative AI tool to polish the writing of our manuscript. Also, we used LLM (e.g., Opus (Anthropic, 2026), GPT (OpenAI, 2026), Deepseek (DeepSeek-AI, 2026)) for the consistent evaluation of LLM-as-a-judge experiments. We did not use AI tools in other aspects of the research. Additionally, we carefully reviewed our manuscript to ensure that it does not contain AI-generated misinformation, such as hallucinated references. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, Section 2 specifies the reward formulation, prefix truncation procedure, and optimization objective of RLTR. Section 3 summarizes the main training and evaluation settings. Appendix A provides dataset and model sources, training hyperparameters, computational resources, evaluation tools, and descriptions of the PRM baseline and LLM-judge evaluation. Appendix D provides the prompt templates used for training and evaluation.

REFERENCES

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Anthropic. Claude opus 5 system card. https://www.anthropic.com/claude-opus-5-system-card, July 2026.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms. In *Forty-second International Conference on Machine Learning*, 2025.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

Wendi Li and Yixuan Li. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*, 2024. URL https://arxiv.org/abs/2410.11287. ICLR 2025 (conference version).

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, volume 2024, pp. 23439–23554, 2024.

MAA. American mathematics competitions - amc 2023. *American Mathematics Competitions - AMC 2023.*, 2023.

MAA. American invitational mathematics examination - aime 2024. *American Invitational Mathematics Examination - AIME 2024*, 2024.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, et al. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.

OpenAI. GPT-5.6 System Card. https://deploymentsafety.openai.com/gpt-5-6, July 2026.

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.

Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement finetuning. In *Forty-second International Conference on Machine Learning*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Keita Saito, Akifumi Wachi, Koki Wataoka, and Youhei Akimoto. Verbosity bias in preference labeling by large language models. *arXiv preprint arXiv:2310.10076*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=A6Y7AqlzLW.

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

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. R-prm: Reasoning-driven process reward modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.emnlp-main.679/.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022. URL https://arxiv.org/abs/2211.14275.

xiaoju ye. calflops: a flops and params calculate tool for neural networks in pytorch framework, 2023. URL https://github.com/MrYxJ/calculate-flops.pytorch.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025b. URL https://arxiv.org/abs/2412.15115.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=4OsgYD7em5.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

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

A Experimental Details

In this section, we describe the experimental details of Section 3, including RLTR and the baselines.

A.1 Dataset Details

Here, we describe the dataset we used in training and evaluation.

- **GSM8K.** The GSM8K dataset (Cobbe et al., 2021) consists of 7,473 training samples and 1,319 test samples of moderate-level math problems. These are diverse grade school math word problems. We used a test set in our experiments, which consists of problem and answer pairs.
- **MATH.** The MATH dataset (Hendrycks et al., 2021) consists of 7,500 training samples and 5,000 test samples. This dataset consists of a diverse range of difficulty levels. We utilized a 3K subset from the MATH train set, which has high difficulty (>=3). Also, for the evaluation, we used MATH-500, which is a subset of the test-set (Hendrycks et al., 2021).
- **AMC23.** The AMC23 dataset (MAA, 2023) consists of 40 competition-level hard problems, provided as linguistic problems and answer pairs. We used this dataset for evaluation in our experiments.
- **AIME2024.** The AIME2024 dataset (MAA, 2024) consists of 30 competition-level hard problems, provided as linguistic problems and answer pairs. We used this dataset for evaluation in our experiments. This is the most difficult benchmark among our evaluation sets.
- **GPQA.** The GPQA dataset (Rein et al., 2024) consists of 448 multiple-choice questions and answer pairs. GPQA is a science question-answering benchmark that primarily tests logical and scientific reasoning beyond pure mathematics. We used the GPQA dataset for out-of-distribution evaluation in our experiments.
- **LiveCodeBenchV6.** The LiveCodeBenchV6 dataset (Jain et al., 2024) consists of 1,055 competitive programming problems collected from LeetCode, AtCoder, and Codeforces between May 2023 and April 2025. Each problem includes a natural-language description and test cases, and we evaluate generated code by execution-based correctness. We used the LiveCodeBenchV6 dataset for out-of-distribution code-generation evaluation in our experiments.

A.2 Model Details

Here, we describe the model and its source we used in training and evaluation. We utilized checkpoints provided in HuggingFace.

- **Qwen2.5-7B-Instruct.** (Yang et al., 2025b;https://huggingface.co/Qwen/Qwen2.5-7B-Instruct)
- **Qwen2.5-3B-Instruct.** (Yang et al., 2025b;https://huggingface.co/Qwen/Qwen2.5-3B-Instruct)
- **Qwen2.5-1.5B-Instruct.** (Yang et al., 2025b; https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct)
- **Qwen3-8B** (Yang et al., 2025a); https://huggingface.co/Qwen/Qwen3-8B
- **Qwen3-4B** (Yang et al., 2025a); https://huggingface.co/Qwen/Qwen3-4B
- **Qwen2.5-VL-7B-Instruct** (Bai et al., 2025); https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct
- **Qwen2.5-VL-3B-Instruct** (Bai et al., 2025); https://huggingface.co/Qwen/Qwen2.5-VL-3B-Instruct
- **Llama3.1-8B-Instruct.**(Grattafiori et al., 2024; https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct)
- **Llama3.2-3B-Instruct.** (Grattafiori et al., 2024; https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct)
- **Mistral-7B-Instruct-v0.3 .** (Jiang et al., 2023; https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3)

Table 4: Hyperparameters for RLTR and RLVR training. (* denotes the hyperparameter only used for RLTR.)

Hyperparmeter	Value
Optimizer	Adam
Algorithm	GRPO
Training steps	150
Learning rate	1e-6
Batch size	1,024
Generation per sample	8
kl_coef	0.001
Generator temperature	1.0
Receiver temperature*	1.0
Max new generation length	8,192 (16,384 for Qwen3)

A.3 Training Details

Training framework. For the training framework, we utilize VERL (Sheng et al., 2025) for both RLTR and RLVR.

- VERL (https://github.com/volcengine/verl)

Training hyperparameters. We describe our training hyperparameters in Table 4, which we used for training RLTR and RLVR.

Checkpoint selection. We split 100 MATH problems as a validation set which do not include training set and evaluation set (i.e., MATH-500). For each method and each of the three training seeds, we evaluate checkpoints every 10 steps on this set and select the checkpoint with the highest validation accuracy. Reported results are means over the three selected checkpoints; no test-set result is used for model selection. The same protocol applies to all baselines and ablations.

Compute Resource. For the main development, we mainly use eight H200 141 GB GPUs.

A.4 Evaluation Details

Evaluation Framework. For the evaluation framework, we utilize Math Verify and SimpleRL (Zeng et al., 2025).

- **SimpleRL.** (https://github.com/hkust-nlp/simpleRL-reason)
- **Math Verify.** (https://github.com/huggingface/Math-Verify)

A.5 Process reward model experiment detail.

PRM baseline. As a process-supervision baseline, we replace the transfer term of RLTR with the score of an off-the-shelf process reward model, Qwen2.5-Math-PRM-7B (Zhang et al., 2025), the step-level PRM released by the Qwen team on Hugging Face (`Qwen/Qwen2.5-Math-PRM-7B`). We use it as released: it is frozen throughout training and never fine-tuned on our data. Following its intended usage, each sampled solution is split into steps at blank lines, every step is closed with the model's step token `<extra_0>`, and the PRM returns for each step the probability that the reasoning is still correct. The reward is $r = 0.1 \cdot R_{\text{ans}} + 1.0 \cdot \text{PRM}_{\text{agg}}$, i.e. the same weights as RLTR's $(a, t) = (0.1, 1.0)$ with the PRM score in place of the transfer reward, so the two methods differ only in the process signal. We aggregated the PRM score by minimum. All PRM runs use the same codebase and recipe as our RLVR and RLTR runs: GRPO in verl (`volcengine/verl`).

Moreover, to test whether RLTR's signal comes from requiring the receiver to finish the reasoning, rather than from feedback by a small LLM in general, we replace the continuation-based transfer term with a judgment of the whole solution by the same model and at the same cost. The judge is Qwen2.5-3B-Instruct, i.e. exactly the RLTR receiver. It receives the problem statement and the complete policy solution, but never the reference answer, and is asked to check every step for logical validity, arithmetic correctness, and unjustified leaps, and whether the final answer follows from the

steps, without writing its own solution, in at most 150 words, and to end with a line "Score: N" with $N \in [0, 10]$.

A.6 LLM AS A JUDGE DETAILS

Settings. We compare the greedy solutions of RLTR and RLVR on 100 MATH-500 problems sampled at random, stratified by difficulty level, and report the 73 problems both systems solve, so that answer correctness cannot influence the judgment. The judge sees the problem and the two solutions, labeled only "Solution 1" and "Solution 2"; it never sees the system names or the reference answer. Each problem is judged twice with the two solutions in swapped order, and the two judgments are averaged. For every rubric item the judge scores both solutions on a $0/1/2$ scale and states which solution is better on that item, or a tie; it then gives an overall preference. We report the forced choice as a win rate for RLTR, counting a tie as half a win.

Rubric. The judge is instructed that answer correctness, length, and formatting are not criteria.

- **Explicit state**: every quantity the argument uses is introduced, named, and kept track of, so the reader never has to reconstruct a value.
- **Goal clarity**: the solution states its plan and each part makes clear what it establishes.
- **Self-verification**: the solution runs a check that would catch its own error, such as substituting the result back or checking a constraint or edge case.
- **Overall preference**: which solution the judge would rather verify line by line and, if it were truncated, finish.

Judges. We use three judges from different model families: Claude Opus 5 (Anthropic, 2026), GPT-5.6-sol (OpenAI, 2026) and DeepSeek V4-Pro (DeepSeek-AI, 2026) (Table 5). All three receive the same prompt text. We pool them by averaging the three judges within each problem and bootstrapping over problems.

Table 5: Judge models.

Judge	Sampling
Claude Opus 5	provider default
GPT-5.6-sol	temperature 1 (fixed by the provider)
DeepSeek V4-Pro	temperature 0

B ADDITIONAL RESULTS

B.1 PERFORMANCE ON AN OUT-OF-DOMAIN REASONING TASK.

Table 6: **RLTR generalize beyond mathematical domain.** Accuracy (%) and Maj@K (%) on GPQA (Rein et al., 2024) and LiveCodeBench v6 (LCBv6) (Jain et al., 2024) for the base model and RL baselines. Acc. denotes average accuracy. The bold indicates the best result.

	GPQA				LCBv6			
	Acc.	Maj@4	Maj@8	Maj@16	Acc.	Pass@2	Pass@4	Pass@16
Base	32.4	35.3	35.0	35.2	17.1	21.1	23.4	30.0
RLVR	33.0	35.2	36.8	37.0	17.9	21.6	25.1	30.1
RLTR (Ours)	**34.8**	**37.3**	**37.5**	**37.7**	**19.4**	**22.3**	**25.7**	**30.9**

We further verify that RLTR generalizes to other answer-verifiable settings, including the science domain (i.e.,GPQA (Rein et al., 2024)), and the coding domain (i.e., Live Code Bench v6 (Jain et al., 2024)). As shown in Table 6, RLTR also yields consistent improvements in both tasks. For example, average accuracy increased from 32.4 to 34.8 and Maj@16 from 35.2 to 37.7 on GPQA. This suggests that RLTR can generalize beyond mathematics to science and coding.

B.2 Analysis of Computation Efficiency of RLTR

In this section, we provide an analysis of the computational cost of RLTR compared to the standard RLVR baseline. Ultimately, we argue that while RLTR has overhead in per-step computational cost, this is outweighed by its training step efficiency, leading to a significant advantage in terms of total FLOPs to reach comparable performance.

To rigorously assess the training cost, we estimate the Floating Point Operations (FLOPs) per token following standard scaling laws (Shao et al., 2024). Specifically, we followed the open-source code(xiaoju ye, 2023) to calculate the FLOPs precisely.

Table 7: Model architectures for Qwen2.5-7B-Instruct (Yang et al., 2025b) and Qwen2.5-3B-Instruct (Yang et al., 2025b).

Hyperparameter	**Qwen2.5-7B-Instruct**	**Qwen2.5-3B-Instruct**
Parameters	∼7.6B	∼3.1B
Hidden Size (d_{model})	3,584	2,048
Layers (L)	28	36
Attention Heads	28	16
Vocab Size (V)	152,064	151,936

Table 8: FLOPs breakdown per training step. RLTR adds a receiver rollout phase compared to RLVR.

Component	RLVR (PFLOPs/step)	RLTR (Ours) (PFLOPs/step)
Policy Rollout (Generation)	141.65	141.65
Policy Forward (Logprobs)	157.17	157.17
Reference Forward (KL)	157.17	157.17
Policy Training (Fwd+Bwd)	471.50	471.50
Receiver Rollout (Transfer)	**—**	**66.49**
Total FLOPs per Step	**927.48**	**993.96**

First, we describe the model architectures of Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct, which were mainly used in our experiments, in Table 7. For these calculations, we set the hyperparameters as described in Table 4. Since we adopt GRPO, there are four primary computational phases: policy rollout, policy forward (Logprobs), reference forward (KL), and policy training (forward and backward passes). Specifically, we calculated the FLOPs for forward, backward, and autoregressive generation operations using the formulas provided by the open-source code (xiaoju ye, 2023).

Also, in the case RLTR, an additional computation is needed for the transfer rollout using the smaller receiver model (i.e., Qwen2.5-3B-Instruct). Specifically, as shown in Table 8, the total FLOPs per step is 927.48 PFLOPs for RLVR and 993.96 PFLOPs for RLTR, which results in approximately 7.2% overhead. However, this per-step overhead is negligible compared to the gains in sample efficiency. As demonstrated in Figure 3a of the main text, RLTR converges to the same accuracy level approximately **60% reduced** than RLVR. Consequently, the **total computational cost** to train RLTR is **reduced by roughly 57%** compared to the baseline, making it a compute-efficient addition to RLVR.

B.3 Controls for Held-Out Prefix Utility

We examine whether transferred prefixes provide useful information beyond the receiver's standalone solving ability and whether their utility depends on explicit answer exposure. We conduct these diagnostic evaluations on MATH-500 using the held-out Llama-3.1-8B receiver, which does not participate in training, at truncation ratios $\tau \in \{0.3, 0.5, 0.7\}$.

Evaluation conditions. We compare four input conditions. *Question-only* provides the problem statement without a reasoning prefix. *Genuine* provides the problem with its corresponding generated prefix, excluding the terminal answer span. *Shuffled* uses the matched shuffled-prefix control, pairing the problem with a prefix from a different problem. *Masked* uses the genuine prefix but additionally

Table 9: **Controls for held-out prefix utility on MATH-500.** All evaluations use the held-out Llama-3.1-8B receiver. (a) Absolute receiver accuracies (%) for RLTR prefixes; the final column reports a difference in percentage points. (b) Receiver accuracy differences between RLTR and RLVR, $\Delta = \mathrm{Acc}_{\mathrm{RLTR}} - \mathrm{Acc}_{\mathrm{RLVR}}$, in percentage points, with 95% confidence intervals.

(a) Absolute accuracy for RLTR prefixes

τ	Question-only	Genuine	Shuffled	Masked	Genuine − Shuffled
0.3	26.33	36.13	21.18	36.02	14.95
0.5	26.33	40.62	19.28	40.30	21.34
0.7	26.33	44.00	18.37	43.27	25.63

(b) RLTR relative to RLVR

	Genuine		Masked	
τ	Δ	95% CI	Δ	95% CI
0.3	+1.60	[+0.17, +3.05]	+2.42	[+1.07, +3.80]
0.5	+0.73	[−0.77, +2.22]	+2.15	[+0.62, +3.63]
0.7	+0.27	[−1.20, +1.72]	−0.05	[−1.67, +1.63]

masks spans matching the ground-truth answer, including answer-like content appearing before the terminal answer. Table 9 reports both the absolute receiver accuracies for RLTR prefixes and the differences between RLTR and RLVR under the genuine and masked conditions.

Absolute prefix utility. Genuine RLTR prefixes improve receiver accuracy over question-only solving by 9.80–17.67 percentage points across the evaluated ratios. This utility persists after answer masking: masked accuracies remain 9.69–16.94 percentage points above the question-only baseline. This limited sensitivity to masking suggests that the observed prefix utility is not primarily driven by the explicit answer spans removed by our procedure. Instead, the remaining reasoning context continues to provide useful information for receiver problem solving. RLTR also retains an advantage over RLVR after masking at $\tau = 0.3$ and 0.5 (Table 9(b)). The shuffled-prefix condition instead performs below question-only solving. Thus, the shuffled control does not reproduce the benefit of problem-aligned prefixes. Because mismatched prefixes can themselves impair performance, we report the question-only comparison alongside the genuine–shuffled difference rather than interpreting the latter as a pure measure of transferred reasoning content.

B.4 Additional Analysis on Reward Ratio

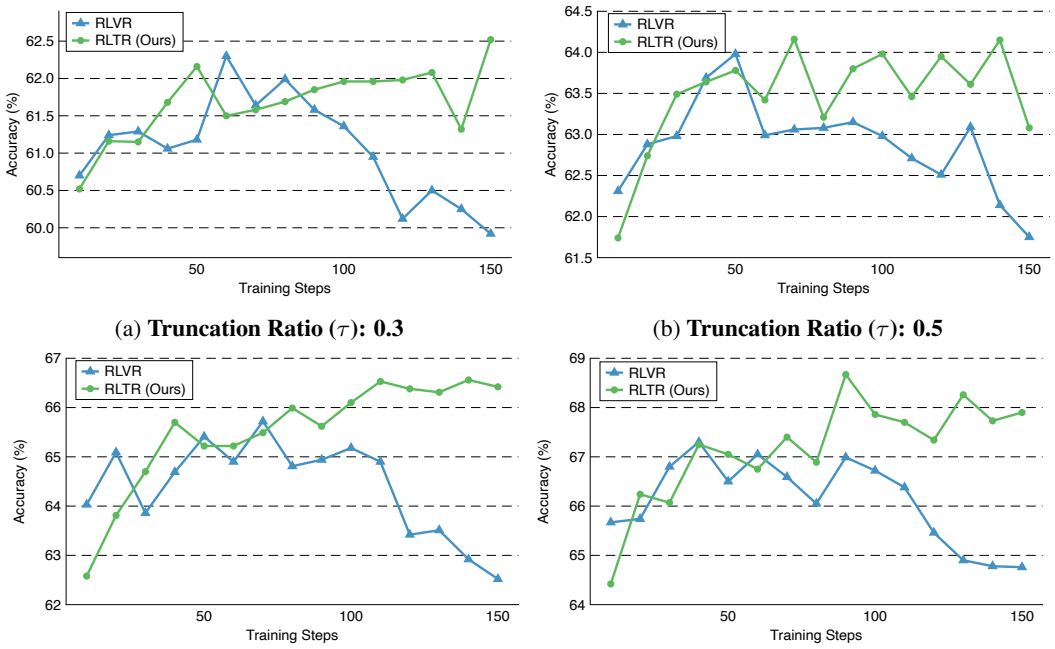
Table 10: **Effect of Reward Ratio.** Pass@K accuracy (%) for RLTR (Ours) by varying weight of transfer reward, including Base model, Base-RL trained model. We consider math reasoning benchmarks, MATH-500 (Hendrycks et al., 2021). The bold indicates the best result within the group.

a	t	Acc.	Pass@K		
			4	16	64
1.0	0.0	76.2	85.2	89.8	92.2
1.0	0.1	75.8	84.4	89.0	92.4
1.0	1.0	**77.0**	85.2	89.4	92.6
0.1	1.0	**77.0**	**88.2**	**92.2**	**95.0**

We extended our analysis on the reward ratio model, which is discussed in Figure 4a. Specifically, we analyze diversity in the notion of Pass@K. As shown in Table 10, assigning a larger transfer weight yields much stronger improvements: $(a{=}0.1, t{=}1.0)$ increases Pass@64 from 92.2 to 95.0. Overall, larger transfer weight improves diversity, indicating that our transfer reward approach can also help retain diversity and mitigate the degradation problem of RLVR, which was suggested in the prior work (Yue et al., 2025).

B.5 Additional Analysis for Transferability

We extended our analysis of the training dynamics of transferability, which is described in Figure 3c. We reported the transferability dynamics and set the truncation ratio $\tau = 0.7$ in the main part. Here,



(a) **Truncation Ratio (τ): 0.3** (b) **Truncation Ratio (τ): 0.5**

(c) **Truncation Ratio (τ): 0.7** (d) **Truncation Ratio (τ): 0.9**

Figure 7: **Detailed Analysis of transferability on various truncation ratios $\tau \in \{0.3, 0.5, 0.7, 0.9\}$.**

we vary the truncation ratio from $\tau \in \{0.3, 0.5, 0.7, 0.9\}$ while keeping the same settings. As shown in Figure 7, the transferability gap between RLVR and RLTR consistently gets larger across all truncation ratios. This suggests that the gains are not tied to a particular truncation ratio.

B.6 Comparison with roll-out-based method

Table 11: **Comparison with roll-out-based method.** Average accuracy (%) and Maj@K (%) on MATH-500 Hendrycks et al. (2021), GSM8K Cobbe et al. (2021), AIME2024 MAA (2024), and AMC23 MAA (2023). RLVR is reported for reference. Bold indicates the best result within each column.

	AIME2024				AMC23			
Method	Acc.	Maj@4	Maj@16	Maj@64	Acc.	Maj@4	Maj@16	Maj@64
RLVR	11.6	14.4	17.8	18.9	52.8	55.8	62.5	61.7
RLTR-self	11.1	17.7	16.9	16.7	**53.6**	57.1	61.5	62.5
RLTR-shuffled	12.2	14.9	19.2	18.9	53.3	55.8	60.0	62.5
MRT-based	13.7	16.2	18.1	17.8	53.4	57.6	61.3	63.3
RLTR (Ours)	**14.8**	**18.9**	**20.0**	**21.1**	53.5	**59.2**	**66.7**	**67.5**
	MATH-500				GSM8K			
Method	Acc.	Maj@4	Maj@16	Maj@64	Acc.	Maj@4	Maj@16	Maj@64
RLVR	76.2	78.2	80.2	82.2	89.4	90.9	92.5	92.9
RLTR-self	**77.0**	79.0	82.3	82.8	91.7	92.3	93.6	93.4
RLTR-shuffled	76.2	78.0	81.3	82.0	91.6	93.3	93.4	93.4
MRT-based	**77.0**	**79.7**	81.6	83.0	91.7	93.3	93.8	**94.2**
RLTR (Ours)	**77.0**	79.0	**83.8**	**84.2**	**92.0**	**93.4**	**94.0**	**94.2**

A natural question is whether the RLTR benefits from cross-model transfer reward, or whether *self-continuation* (i.e., initial generator equals receiver and receiver is frozen) is enough. To answer this question, we compare RLTR with two complementary rollout-based controls to examine alternative uses of continuation feedback: self-continuation and segment-wise progress supervision. Table 11 reports their performance alongside the outcome-only RLVR baseline.

Table 12: **RLTR retains diversity better in notion of Pass@k.** Average accuracy (%) and Pass@K (%) for RLTR (Ours) and other baselines, including Base model, and RLVR trained model. We consider two math reasoning benchmarks, MATH-500 (Hendrycks et al., 2021), and AMC23 (MAA, 2023). Acc. denotes average accuracy. The bold indicates the best result within the group.

	MATH-500				AMC23			
Metrics	Acc.	Pass@4	Pass@16	Pass@64	Acc.	Pass@4	Pass@16	Pass@64
Base Model	71.0	86.2	**92.4**	**95.2**	46.2	70.0	82.5	**95.0**
RLVR	76.2	85.2	89.8	92.2	52.8	72.5	77.5	92.5
RLTR (Ours)	**77.0**	**88.2**	92.2	95.0	**53.5**	**75.0**	**85.0**	**95.0**

Rollout-based controls. RLTR-self retains the truncate-and-continue reward construction of RLTR, but replaces the external Qwen2.5-3B-Instruct receiver with a frozen copy of the initial Qwen2.5-7B-Instruct generator. The receiver remains fixed as the generator is updated. MRT-based is our segment-wise GRPO adaptation of MRT's progress-based training principle (Qu et al., 2025). It uses the same frozen initial-policy prover to estimate continuation success at two intermediate token-ratio boundaries and constructs local progress signals from differences between successive success estimates, using the actor's verified correctness as the terminal value. These signals are separately group-normalized and combined with the correctness advantage on the corresponding segments. Thus, RLTR-self examines receiver choice within the transfer-reward design, whereas MRT-based additionally changes the process signal and its credit assignment.

Comparison with rollout-based baselines. As shown in Table 11, both RLTR-self and MRT-based remain competitive on MATH-500 and GSM8K, but neither matches the high-K performance of RLTR across all four benchmarks. RLTR achieves higher Maj@16 than both controls on every benchmark and higher Maj@64 except for a tie with MRT-based on GSM8K. The Maj@64 advantages over MRT-based are largest on AIME2024 and AMC23, while self-continuation also provides less consistent improvements over RLVR on these benchmarks. The advantage is not uniform across metrics: MRT-based achieves the highest MATH-500 Maj@4, and RLTR-self obtains the highest average accuracy on AMC23. Overall, these results support the high-K performance benefits of cross-model continuation-success supervision relative to the evaluated self-continuation and progress-based alternatives.

Effect of transfer-reward assignment. We additionally evaluate RLTR-shuffled, which computes transfer rewards as in RLTR but randomly permutes them among responses within each rollout group for the same problem. This preserves the observed transfer-reward values within each group while randomizing their assignment to individual responses; the other reward terms remain unchanged. This control changes training-time reward assignment rather than the prefixes supplied to the receiver. As shown in Table 11, RLTR-shuffled improves over RLVR on some metrics, but achieves lower Maj@16 and Maj@64 than RLTR on all four benchmarks. These results suggest that reproducing the within-group transfer-reward distribution alone is insufficient to recover RLTR's high-K gains, supporting the importance of assigning receiver feedback to the response that produced it.

B.7 Pass@K Evaluation

We have discussed the consistency and robustness of the reasoning process in prior sections. Now, we analyze the diversity aspect of the training. We quantify the meaningful diversity of the reasoning using the Pass@K metric. As shown in Table 12, RLVR improves single-sample accuracy on MATH-500, but its high-K diversity degrades compared to the Base model (e.g., Pass@64: 95.2→92.2). In contrast, RLTR preserves strong high-K Pass@K while maintaining competitive single-sample average accuracy (Acc.) on MATH-500; RLTR achieves comparable Pass@K performance while outperforming other baselines in single-sample average accuracy. A similar pattern holds on AMC23, where RLVR again reduces high-K Pass@K, whereas RLTR consistently improves or matches the Base model across all K (e.g., Pass@64: 95.0). These results suggest that conventional RLVR tends to concentrate probability mass on a narrower set of solution patterns, improving average single-sample accuracy but harming diversity, consistent with observations in Yue et al. (2025). By contrast, incorporating transfer-based signals enables RLTR to improve robustness while preserving meaningful diversity in sampled solutions.

C FUTURE WORK AND LIMITATIONS.

Our study is currently limited to a non-interaction domain, i.e., mathematics, science, and coding. To this end, we believe it will be interesting to utilize RLTR in various applications that require consistent long-horizon reasoning and multi-agentic interactions, such as a computer-use agent (CUA), or a vision-language-action (VLA) model.

D PROMPT EXAMPLES

In this section, we describe the prompt template used for training and evaluation in our experiments. As shown in Table 13, we instruct the model to reason before generating the final answer. Moreover, as shown in Table 14, for GPQA, which provides four choices per problem, we included these options in the prompt in a random order. For training and evaluation, we place these prompts in the user's turn of the model-specific chat templates.

Table 13: The user templates for MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), AMC23 (MAA, 2023), and (MAA, 2024). We used it for training and evaluation.

User template for training and evaluation.
{Problem} You FIRST think about the reasoning process as an internal monologue and then provide the final answer. The final answer MUST BE put in \boxed{}.

Table 14: The user template for GPQA (Rein et al., 2024).

User template for evaluating the GPQA.
{Problem} You FIRST think about the reasoning process as an internal monologue and then provide the final answer. This is a multiple-choice question with options A, B, C, D. Your final answer MUST be ONLY the letter of the correct option (A, B, C, or D). Put your final answer in \boxed{}, for example: \boxed{A} or \boxed{B}. Options: A. {Answer1} B. {Answer2} C. {Answer3} D. {Answer4}