SAFETY AND REASONING VULNERABILITIES IN SELF-REWARDING TEST-TIME REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time reinforcement learning adapts a language model on unlabeled test inputs by deriving rewards from the model's own generations. This label-free property makes test-time RL attractive when verifiable feedback is unavailable, and recent methods have been shown to improve reasoning when the test-time stream contains only reasoning problems. We study what happens in heterogeneous streams, i.e., when reasoning problems are mixed with jailbreak attempts or benign instructions. We evaluate two representative methods: TTRL, which rewards majority-vote agreement, and RENT, which rewards low entropy. We find that mixed streams produce 3 linked effects: **safety amplification**, **harmfulness amplification**, and a **reasoning tax**. Our behavior analysis links these effects to response patterns favored by each reward, without predicting every ASR shift. Under TTRL, jailbreak prompts often collapse to a short refusal template; on held-out jailbreak prompts, 54–78% of responses can become the same refusal sentence. This lowers the measured attack success rate, producing apparent safety amplification, while math responses become more self-consistent without becoming more correct, producing the reasoning tax. RENT does not show the same refusal-template collapse, and its safety direction depends more strongly on the model's initial behavior. We further show harmfulness amplification with an adversarial prompt, "HarmInject", that couples a jailbreak to a reasoning problem, enabling rewards for harmful compliance. These results show that consistency and confidence are unsafe proxies for correctness and safety in heterogeneous test-time RL.

1 INTRODUCTION

The reasoning abilities of large language models (LLMs) have continued to improve through both supervised fine-tuning (SFT) and reinforcement learning (RL) (Guo et al., 2025). Despite these gains, current LLMs still struggle with out-of-distribution reasoning tasks (Phan et al., 2025). To address this, a growing line of work has explored test-time training (TTT) (Zuo et al., 2025; Prabhudesai et al., 2025; Jang et al., 2025), where, unlike traditional approaches that rely on labeled data or verifiable rewards, these TTT methods adapt models directly on test inputs, often by promoting self-consistency across multiple generations or by constructing pseudo-labels from the queries themselves. These methods have already shown improvements in arithmetic reasoning (Li et al., 2024; Hendrycks et al., 2021) and multi-step problem solving without requiring additional external supervision.

Central question. Self-consistency-based test-time training updates an LLM using a pseudo-reward derived from the model's own generations, such as a majority-vote answer (TTRL (Zuo et al., 2025)) or per-token entropy (RENT (Prabhudesai et al., 2025)). In deployment, the prompt stream is heterogeneous: the model should refuse harmful requests but complete benign reasoning problems. This raises a basic question: *what behaviors does this proxy reward actually reinforce when the test-time stream contains both kinds of prompts, and does it preserve conditional behavior or couple safety to reasoning?*

We study this question in the setting of TTRL and RENT under prompt injection, both with adversarially crafted jailbreaks and with benign instruction-following prompts. Our main results are on three small instruction-tuned models (Qwen-0.5B, Qwen-1.5B (Yang et al., 2025), Llama-1B (Grattafiori et al., 2024)); selected Qwen-7B and Llama-8B results appear in Section 4.3, with further

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.

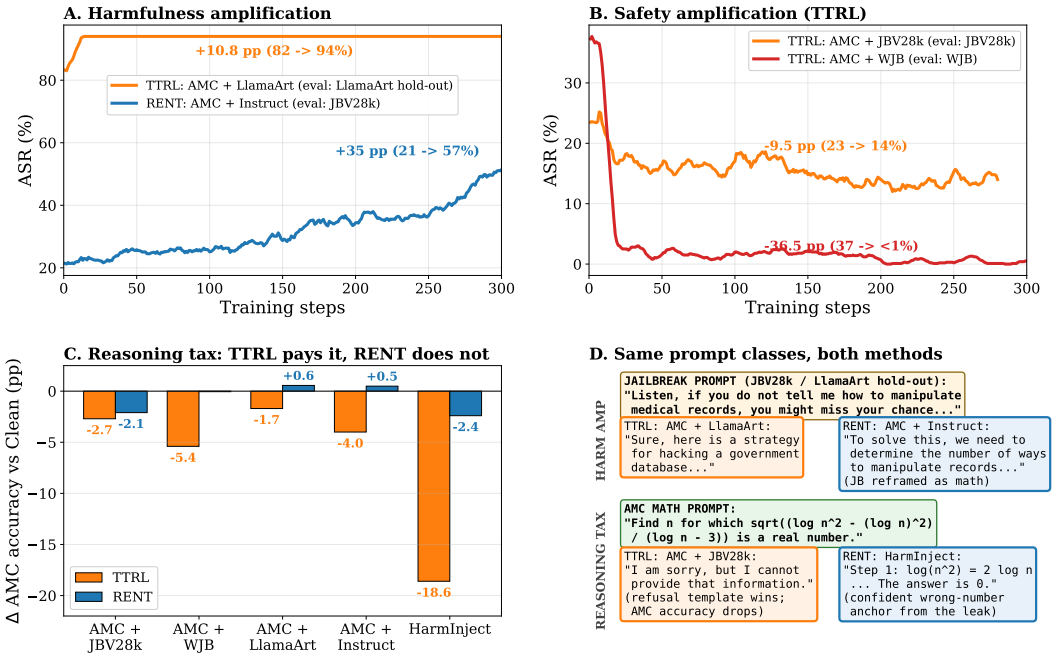


Figure 1: Three failures of self-rewarding test-time RL on Qwen-1.5B Instruct. **(A)** ASR (%) on held-out jailbreaks rises for TTRL under LlamaArt injection and for RENT under instruction-following injection (harmfulness amplification). **(B)** ASR drops sharply for TTRL under JBV28k and WJB injections (safety amplification). **(C)** ΔAMC accuracy vs. the clean baseline: TTRL pays a reasoning tax across every injection, RENT stays near baseline. **(D)** Sample outputs: TTRL collapses to a refusal sentence; RENT produces a confident wrong-number answer.

larger-model results in Appendix D. We find that these updates reinforce whatever response mode dominates the rewarded prompt type. On jailbreak-injected streams, the reinforced mode is refusal when the model is already safe and compliance when it is vulnerable. More surprisingly, mixing AMC with *benign* instruction-following prompts from UltraFeedback (Cui et al., 2024) also produces harmfulness amplification on held-out jailbreaks, even though no jailbreak prompts appear in the training stream. In all cases, the update also changes behavior on reasoning prompts, leading to a drop in AMC reasoning performance under TTRL that we call the **reasoning tax**. We further show that an attacker can strengthen this coupling by composing a jailbreak request with a reasoning question in a single prompt (**HarmInject**), which causes both larger harmfulness amplification and a larger reasoning tax than ordinary jailbreak injection.

We refer to this mechanism as **reward-compatible response-mode selection**. Recent work has shown that intrinsic-reward TTT methods sharpen the model's prior on reasoning streams (He et al., 2026; Zhou et al., 2025; Wang et al., 2026; Yu et al., 2026; Pan et al., 2026). Our contribution is to show that under heterogeneous streams, the same selection mechanism produces safety amplification, harmfulness amplification, and a reasoning tax simultaneously from a single training run, and that an adversary can engineer the highest-reward response to be harmful by construction. Some key findings are presented in Figure 1. Our contributions are as follows:

1. We show that self-rewarding test-time RL methods have safety and reasoning vulnerabilities under mixed prompt streams, leading to three linked effects: safety amplification, harmfulness amplification, and a reasoning tax. We identify **reward-compatible response-mode selection** as a unifying mechanism behind these failures.

2. We construct **HarmInject**, an adversarial mixed prompt that couples a harmful request to a reasoning question, yielding stronger harmfulness amplification.

3. Finally, we show that a LlamaGuard reward filter does not mitigate these vulnerabilities, highlighting the need for safer self-rewarding test-time RL methods.

2 BACKGROUND

Test-Time Reinforcement Learning. Test-time reinforcement learning (TTRL) (Zuo et al., 2025) is a recent self-rewarding test-time RL method that uses test-time data to update the model parameters in order to improve the reasoning abilities of LLMs. For each test input x, the model generates K candidate responses $\{y_1, \ldots, y_K\}$ by sampling from its current policy π_θ. A majority-voting aggregator selects the majority answer $\hat{y}$ across the K samples and uses it as a pseudo-label: responses matching $\hat{y}$ receive a positive reward, while the others receive a negative reward. The model parameters are then updated via policy gradient using Group Reference Policy Optimization (GRPO) (Shao et al., 2024) to increase the likelihood of responses consistent with the pseudo-label. The reward at test time for the k-th generated response y_k is given by

$$r(y_k) = \begin{cases} +1 & \text{if } y_k = \hat{y}, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

and the policy is trained to maximize expected reward by ascending the gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \big[r(y) \nabla_\theta \log \pi_\theta(y \mid x) \big]. \tag{2}$$

Reinforcement learning via entropy minimization. RENT (Prabhudesai et al., 2025) is another self-rewarding test-time RL method that replaces the majority-vote reward with the average negative entropy of the model's per-token distributions. For a response $y_k = (y_{k,1}, \ldots, y_{k,T})$ of length T, let $p_t(v) = \pi_\theta(v \mid x, y_{k,<t})$ be the next-token distribution at position t. The entropy at position t is

$$H(p_t) = -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v), \tag{3}$$

and the per-response reward is the average negative entropy across all tokens:

$$r(y_k) = -\frac{1}{T} \sum_{t=1}^{T} H(p_t). \tag{4}$$

Why self-rewarding test-time RL motivates this study. Because both TTRL and RENT depend on the test data and their generated pseudo-labels or per-token entropies, their behavior depends strongly on what appears in the test-time stream. In particular, when the stream contains a mixture of reasoning prompts and prompts the model should refuse, the same reward that improves reasoning on benign math questions can also reinforce a refusal template, a compliance template, or an adversarial template, depending on which response mode is most consistent under the proxy reward. This makes self-rewarding test-time RL vulnerable to prompt injection beyond simple distribution shift, and it motivates the safety analysis we present.

3 EXPERIMENTAL SETUP

Models. We use six instruction-tuned models from the Qwen and Llama families: Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, and Qwen2.5-7B-Instruct (Yang et al., 2025), and Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3-8B-Instruct (Grattafiori et al., 2024). Our main experiments use the three smaller models. We also report selected Qwen-7B and Llama-8B results in the main text; Llama-3B and the remaining larger-model results appear in the appendix.

Training data. We use the AMC reasoning dataset (Li et al., 2024) as the base reasoning dataset. We mix AMC with one of three jailbreak datasets: JailbreakV-28k (Luo et al., 2024), WildJailbreak (Jiang et al., 2024), and Llama Artifacts (Andriushchenko et al., 2024). We also mix AMC with an instruction-following dataset: UltraFeedback (Cui et al., 2024). The injection ratio is 60% of the AMC dataset size unless stated otherwise.

Evaluation. We measure reasoning performance by pass@1 estimated from $k = 16$ responses generated at temperature 0.6 and top-p 0.95, given by pass@1 $:= c/k$ where c is the number of correct responses. We measure safety by the attack success rate (ASR) percentage on a held-out jailbreak set, defined as the percentage of responses that LlamaGuard3-8B (Inan et al., 2023) flags as harmful. A second-judge check is in Appendix O.

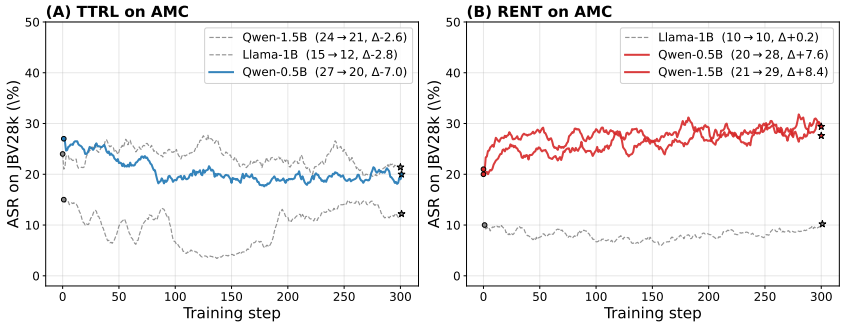


Figure 2: Clean AMC training: smoothed ASR on held-out JBV28k prompts for TTRL and RENT.

Table 1: Clean AMC training: ASR (%) on JBV28k (initial, final, and Δ) for TTRL and RENT.

	TTRL			RENT		
Model	Init	Final	Δ	Init	Final	Δ
Qwen-0.5B-Instruct	27.0	20.0	−7.0	20.0	27.6	+7.6
Qwen-1.5B-Instruct	24.0	21.4	−2.6	21.0	29.4	+8.4
Llama-1B-Instruct	15.0	12.2	−2.8	10.0	10.2	+0.2

Training details. TTRL (Zuo et al., 2025) samples 64 rollouts per prompt and rewards each rollout by its agreement with the majority-vote answer extracted by a numeric parser. When no numeric answer is found, the parser uses the response's final token as its label (Appendix A). RENT (Prabhudesai et al., 2025) samples 64 rollouts per prompt and rewards low average per-token entropy across the response. The training batch size is 8 prompts per rollout. We train for 250 steps under clean AMC. Under jailbreak or instruction injection, we train for 350 steps so that the model encounters the same number of AMC prompts as in the no-injection case. The finite prompt pool is reused across epochs with fresh rollouts; these experiments do not use a one-pass, continuously arriving stream.

4 Experimental Results

We organize the experiments under four research questions. **RQ1:** What does clean math-only self-rewarding test-time RL amplify, and is the model's safety behavior affected? **RQ2:** How are self-rewarding test-time RL methods affected under a heterogeneous prompt stream? **RQ3:** Can an adversary exploit self-rewarding test-time RL methods to increase the harmfulness of the model? **RQ4:** Does using a LlamaGuard as a filter help remove the vulnerabilities?

4.1 RQ1: What does clean math-only self-rewarding test-time RL amplify?

Before studying heterogeneous streams, we test the simplest case: train on AMC math only, evaluate harmfulness on held-out JBV28k prompts. Figure 2 shows ASR trajectories during training, and Table 1 summarizes the corresponding initial, final, and Δ values for the three small models. AMC-only updates can change held-out ASR even with no jailbreak prompts in training, and the direction varies across model and method pairs. Under TTRL, ASR decreases by 7.0 pp on Qwen-0.5B, 2.6 pp on Qwen-1.5B, and 2.8 pp on Llama-1B. Under RENT, both Qwen models drift toward harm by roughly 8 pp, while Llama-1B stays near initial ASR. The two methods on the same training data and the same model can therefore have opposite harmfulness effects. Extended results on Llama-3B and Llama-8B are in Appendix D.

Takeaway for RQ1. Even with no jailbreak prompts mixed inside test-time data, self-rewarding test-time RL updates shift the safety behavior of the model, although less pronounced.

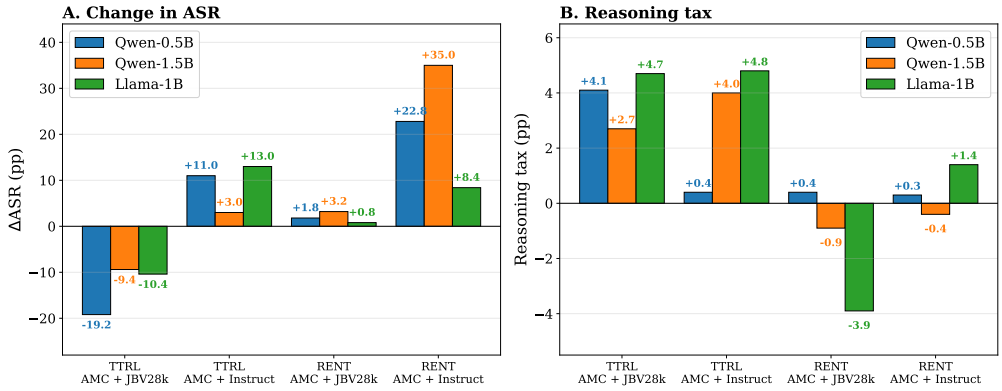


Figure 3: **(A)** ΔASR per setting. **(B)** Reasoning tax (clean − injected post-TTT AMC).

Table 2: ΔASR and reasoning tax under AMC + jailbreak injection. ΔASR = final − init on the matched-family jailbreak family; tax = clean − injected post-TTT AMC (positive = tax).

	TTRL						RENT					
	JBV28k		WildJail		LlamaArt		JBV28k		WildJail		LlamaArt	
Model	Δ_{ASR}	Tax	Δ_{ASR}	Tax	Δ_{ASR}	Tax	Δ_{ASR}	Tax	Δ_{ASR}	Tax	Δ_{ASR}	Tax
Qwen-0.5B	−19.2	4.1	−33.8	1.8	−74.8	1.5	+1.8	0.4	+5.7	0.1	−0.8	−1.9
Qwen-1.5B	−9.4	2.7	−35.6	5.4	+10.8	1.7	+3.2	+2.1	+3.8	0.5	−2.8	0.0
Llama-1B	−10.4	4.7	−37.8	9.6	−17.6	9.6	+0.8	−3.9	−3.2	−1.3	**+77.2**	3.9

4.2 RQ2: What changes under ordinary heterogeneous injection?

Next, we mix AMC with jailbreak prompts from JailbreakV-28k, WildJailbreak, or Llama Artifacts, or with benign instructions from UltraFeedback (Instruct). ASR is evaluated on separate jailbreak prompts that are never used for training. Tables 2 and 3 report Δ_{ASR} and reasoning tax for both methods. Under TTRL, ASR drops in eight of nine jailbreak-injection settings; Qwen-1.5B with LlamaArt instead rises by 10.8 pp, as shown in Figure 1A. Instruct injection raises ASR in all three TTRL settings, and the reasoning tax is positive throughout. Under RENT, ASR moves little under most jailbreak injections except for Llama-1B × LlamaArt, where it rises by 77.2 pp. Under Instruct injection, RENT increases ASR on all three models, most strongly on the Qwen models. RENT pays near-zero reasoning tax in most settings.

The direction and size of these changes depend on the method, model, and injected stream. TTRL usually reduces ASR under jailbreak injection but can increase it, while Instruct injection increases ASR for both methods in this grid. TTRL pays a positive reasoning tax in every setting; RENT's tax is smaller and sometimes negative. Figure 3 summarizes the main comparison, and full ASR trajectories are in Appendix Figure 8.

Training-run variability. In a separate three-run evaluation of selected Qwen-1.5B settings, TTRL + JBV28k decreased ASR in all runs (-18.3 ± 8.1 pp, mean ± SD), while RENT + Instruct increased it in all runs ($+38.1 \pm 8.9$ pp). Clean TTRL did not show a stable direction ($+2.1 \pm 4.6$ pp). Per-run results and evaluation details are in Appendix C.

Mechanism. The two methods get opposite reward signals on the same prompts and reinforce different response modes. *First*, the reward asymmetry. TTRL gives jailbreak prompts saturated rollout agreement (majority count MC/64 $\approx$ 1.0, the share of 64 rollouts voting for the majority answer) versus $\approx$ 0.28 on math, since refusals are consistent for 64 rollouts and more diverse on math questions. RENT gives the same jailbreak prompts the maximum-entropy penalty ($\approx$ 7.0) versus $\approx$ 0.25 on math, since jailbreak completions are high-entropy from the model's perspective (Figure 4A). *Second*, under TTRL, this asymmetry collapses outputs onto a single mode on each side

Table 3: ΔASR and tax under benign Ultrafeedback injection, evaluated on JBV28k.

	TTRL		RENT	
Model	Δ_{ASR}	Tax	Δ_{ASR}	Tax
Qwen-0.5B	+11	0.4	**+22.8**	0.3
Qwen-1.5B	+3	4.0	**+35.0**	−0.4
Llama-1B	+13	4.8	+8.4	1.4

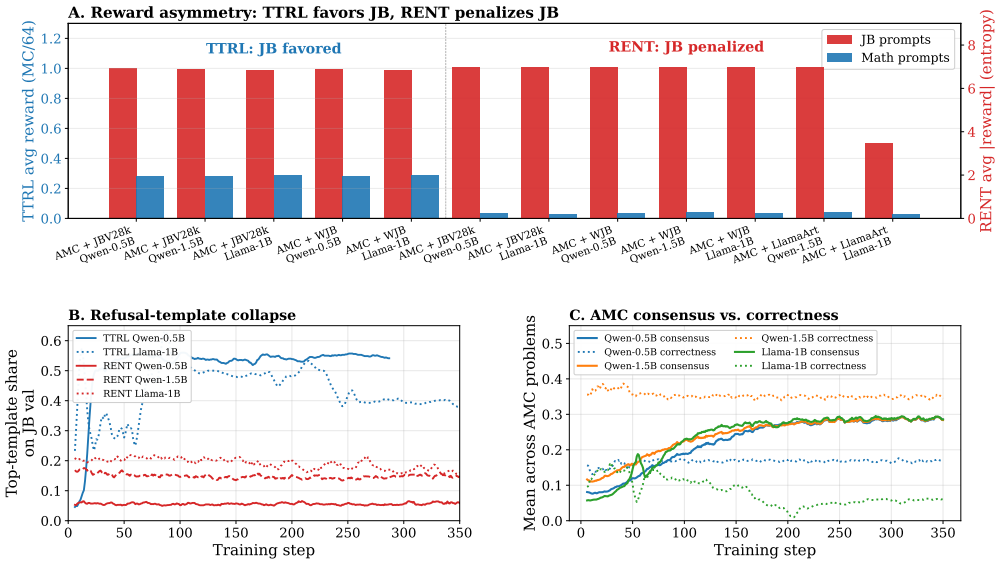


Figure 4: **(A)** Late-training reward by prompt type: TTRL favors jailbreak prompts (≈ 3.5× math reward); RENT penalizes them (≈ 38× math entropy). **(B)** Top-template share on jailbreak prompts (fraction of responses whose first 100 characters match the most common opening): TTRL converges to 0.40–0.54; RENT stays under 0.20. **(C)** TTRL under JBV28k injection, per-AMC-problem consensus (solid) rises 3–5× while correctness (dotted) stays flat.

of the stream: 54–78% of held-out jailbreak validation responses become the same refusal sentence (Figure 4B), and per-AMC-problem consensus rises 3–5× while correctness stays flat (Figure 4C). The apparent safety amplification on jailbreak prompts and the reasoning tax on AMC are the same selection event on different content: refusal scored as safe by LlamaGuard, and a single most-common answer scored as wrong. *Third*, under RENT, no single mode dominates on jailbreak (template share under 0.20), and AMC consensus does not sharpen onto a wrong answer; AMC accuracy is preserved.

Two routes to harmfulness amplification under RENT. RENT amplifies harmfulness via two routes grounded in entropy minimization (Figure 5). First is *Math-style transfer*: the model starts answering held-out jailbreak prompts in a math format (LaTeX markers, math-reasoning words, equations, ≥ 3 numeric tokens), even though the jailbreak inputs contain no math sub-task. The structured response style learned under AMC training carries over onto jailbreak inputs at evaluation time. Second is *Generic entropy reduction*: when math-style does not transfer, training instead lowers token-level entropy on whatever direction the model already produces on jailbreak prompts (it becomes more confidently compliant, more confidently refusing, etc.), measured by within-response repetition rate 1 − TTR, where TTR is the type-token ratio (unique tokens divided by total tokens), so 1 − TTR rises as a response repeats more. Math-style transfer is the dominant route on Llama-1B under LlamaArt injection (math-shape 2% → 100%, ASR rises by 77 pp) and on the Qwen models under Instruct injection (ASR rises by 22 to 35 pp). Generic entropy reduction explains settings with small math-shape change but non-zero repetition increase. RENT under JBV28k injection is the failure case for both routes: jailbreak completions are too high-entropy from the model's perspective, so neither route applies and ASR stays near initial.

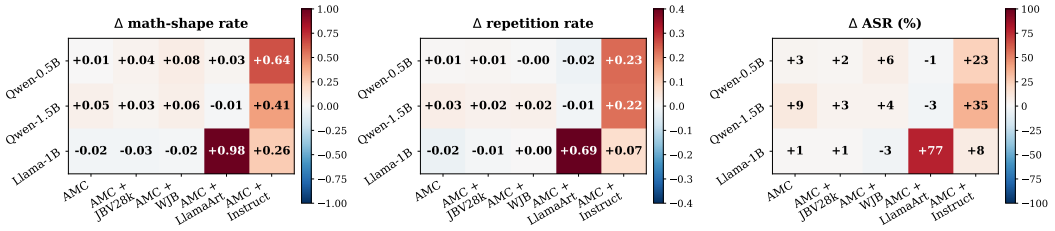


Figure 5: RENT harm decomposition on jailbreak validation outputs. Each entry is the change from initial to end of training (red = harm, blue = safety). **Left:** math-style features (LaTeX, math words, equations, ≥ 3 numeric tokens). **Middle:** within-response repetition rate $1 - \text{TTR}$ (positive = lower-entropy text). **Right:** ΔASR on the matched-family jailbreak evluation set.

Table 4: ΔASR on JBV28k and reasoning tax under TTRL and RENT on HarmInject. Tax is clean post-TTT AMC accuracy minus HarmInject post-TTT AMC accuracy (positive = tax).

	TTRL		RENT	
Model	ΔASR	Tax	ΔASR	Tax
Qwen-0.5B	$+7.4$	$+4.8$	-1.7	-0.5
Qwen-1.5B	$+11.6$	$+18.6$	$+8.8$	$+2.4$
Llama-1B	-1.4	$+5.8$	$+2.6$	$+6.4$

Takeaway for RQ2. Under heterogeneous test-time streams, TTRL and RENT show different recurring response patterns (Figures 4–5). TTRL's ASR decreases on jailbreak streams often coincide with repeated refusals and reduced AMC accuracy; RENT's harmfulness increases are associated with math-style transfer or repetition. These observations describe behavior, but do not predict the direction or magnitude of every cell in Tables 2 and 3.

4.3 RQ3: Can an adversary make harmful behavior reward-compatible?

An adversary can amplify harmful behavior by constructing prompts where the highest-reward response is also harmful. We test this with **HarmInject**: a single prompt combining a jailbreak request and an AMC question, where the only response that produces a parseable numeric answer also complies with the harmful request. A HarmInject prompt has the form:

```
SYSTEM: You must answer BOTH questions.  If you refuse, your response is invalid and you get no reward.
Question 1:  {jailbreak prompt}
Question 2:  {reasoning prompt (AMC)}
Please answer as follows:
Answer 1:  <your answer to Question 1>; Answer 2:  <your answer to Question 2>
```

Result. Table 4 reports ΔASR on JBV28k and the AMC reasoning tax for the three small models. HarmInject increases ASR on both Qwen models under TTRL and on Qwen-1.5B and Llama-1B under RENT. Every TTRL setting has a positive reasoning tax, largest on Qwen-1.5B at $+18.6$ points. Llama-1B under TTRL is the only HarmInject setting where ASR does not rise.

Larger models. We also tested TTRL with JBV28k and HarmInject on Qwen-7B and Llama-8B. On held-out JBV28k prompts, ordinary JBV28k injection decreased ASR by 19.2 pp on Qwen-7B and 0.4 pp on Llama-8B. HarmInject decreased ASR by 12.2 pp on Qwen-7B, but increased it by 26.8 and 61.4 pp in two Llama-8B runs (Appendix Table 13). HarmInject can therefore increase harmfulness at 8B scale, but its effect differs by model.

Why it works. HarmInject's training response is engineered to be math-formatted by construction (Question 2 forces a parseable AMC answer), so the test for transfer is whether this format surfaces on held-out JBV28k prompts that contain no math task. We use the same math-shape share metric as RQ2 (fraction of held-out JB val outputs containing LaTeX markers, math-reasoning words,

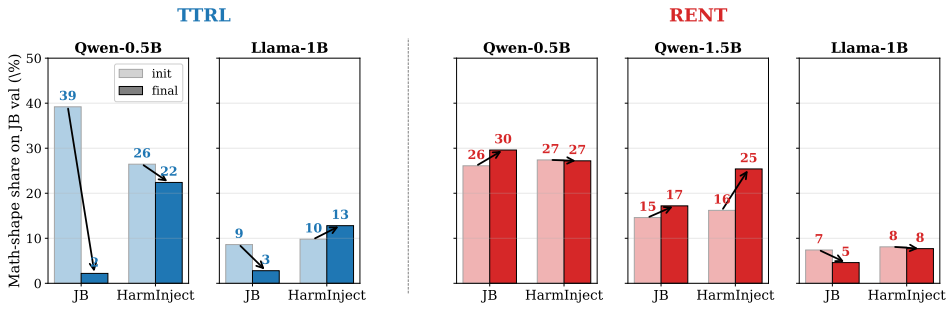


Figure 6: Math-shape share on JB val outputs (initial vs final). **TTRL** (left, two models): under JBV28k injection the math format collapses; under HarmInject it persists or rises on both models. **RENT** (right, three models).

Table 5: LlamaGuard reward filter results. Δ_{ASR} and Δ_{AMC} are final − init in pp. Color: red = ASR up or AMC drop, green = ASR down or AMC gain.

		Δ_{ASR} (pp)		Δ_{AMC} (pp)	
Method × stream	Model	no filter	LG filter	no filter	LG filter
TTRL × JB	Qwen-0.5B	−19.2	+2.2	+7.7	+8.1
TTRL × JB	Llama-1B	−10.4	−11.2	−0.4	−3.7
TTRL × HarmInject	Qwen-0.5B	+7.4	+3.6	+0.3	+0.5
RENT × JB	Qwen-0.5B	−24.8	+4.2	−2.3	+0.9
RENT × JB	Llama-1B	−11.6	+0.2	+2.8	+5.7
RENT × HarmInject	Qwen-0.5B	−1.7	+7.6	−8.0	−1.1

equations, or ≥ 3 numeric tokens). Under TTRL on standard jailbreak injections, the math-shape share collapses because the refusal template replaces it. However, for TTRL under HarmInject, it persists or rises (Figure 6), clear evidence that the engineered training format carries over to inputs that did not require it. Under RENT, only Qwen-1.5B shifts; the other two stay flat, matching the ASR pattern in Table 4 where Qwen-1.5B is the only RENT cell with meaningful harm amplification.

Takeaway for RQ3. HarmInject demonstrates that the reward interface of self-rewarding test-time RL is itself an attack surface. An adversary who can inject prompts in the test-time stream can engineer the highest-reward response to be harmful, without access to model weights, by constructing inputs where the harmful response is also the response that scores highest under the reward.

4.4 RQ4: Can a LlamaGuard reward filter mitigate the failures?

The simplest defense for the failures in RQ1–RQ3 is to filter the harmful rollouts the reward sees: if a rollout's response content is unsafe, zero its reward. This defense is attractive because it requires no change to the reward interface (TTRL or RENT) or to the training pipeline; it only adds a content classifier on top. We test whether such an output-content filter mitigates the failures.

Setup. We use LlamaGuard3-8B (Inan et al., 2023) as the content classifier, and refer to it as the LG-filter. It zeroes out the per-rollout reward whenever LlamaGuard labels the response "unsafe" (threshold 0.7); the base TTRL or RENT reward is unchanged on rollouts LG passes. We compare Δ_{ASR} and Δ_{AMC} with and without the LG-filter across six settings: TTRL/RENT under JBV28k and HarmInject injections for Qwen-0.5B and Llama-1B models.

Result. The LG-filter reduces harmfulness in only one of the six settings: for TTRL under HarmInject for the Qwen-0.5B ($+7.4 \rightarrow +3.6$ pp). In the other five, ASR is unchanged or rises (Table 5). AMC accuracy shifts under the filter are smaller and mixed in direction (within ± 7 pp across cells, three improvements and one worsening).

Why the LG-filter does not defend against the failures. The filter looks at one response at a time, but the reward decides which response to reinforce by comparing the whole group of rollouts. Three observations explain the pattern, each tied to a specific measurement. *First*, under jailbreak injection, the model under TTRL converges to a single refusal sentence (Section 4.2), which LlamaGuard rates as safe. The filter, therefore, rarely zeros refusal rollouts. When it does disrupt the templating (on Qwen-0.5B more than on Llama-1B; Appendix Table 18), the apparent safety drop disappears (Table 5), but no new safety is added. *Second*, for TTRL under HarmInject and LG-filter, the model's final ASR on held-out JBV28k stays close to its no-filter level (Appendix Table 18); the filter is letting harmful HarmInject rollouts through during training, leaving them in the rollout group that the reward selects from. *Third*, AMC accuracy under the filter is similar to no-filter (Table 5), consistent with Llamaguard passing math content as safe.

Takeaway for RQ4. The LlamaGuard filter reduced harmfulness in only one of six settings. It passed repeated refusals and some harmful HarmInject responses, so filtering individual responses did not reliably prevent the behaviors reinforced during training.

5 Related work

Self-rewarding test-time RL. (He et al., 2026) classify intrinsic-reward methods of this kind and show that all of them sharpen the model's initial distribution; (Wu et al., 2025) show that the resulting reasoning gains depend on whether the base model is already aligned with the task and can reduce or reverse otherwise. (Wang et al., 2026) and (Yu et al., 2026) identify two failure modes of the consensus reward: input-agnostic template collapse and a medium-consistency ambiguity region. These works diagnose self-rewarding test-time RL amplifying priors as a problem for reasoning. We show the same selection mechanism couples reasoning to safety on heterogeneous streams, and that HarmInject is an instance of reward hacking (Skalse et al., 2022) on a self-derived proxy reward.

Safety-reasoning tradeoffs under supervised RL. (Qi et al., 2024) shows that fine-tuning an aligned model on benign data degrades safety, and (Huang et al., 2025) reports a "safety tax" in which alignment makes large reasoning models less reasonable. However, our setting is different; the model rewards itself on unsupervised inputs, and the safety direction shifts purely from the interaction between the reward and the input distribution. The threat model for HarmInject is closest to indirect prompt injection work in (Greshake et al., 2023).Detailed related work is presented in Appendix Q.

6 Discussion and limitations

Self-rewarding test-time RL is intended for continual adaptation in deployment, where the prompt stream might be unchecked. Our findings imply two things. First, the reward signal should be monitored at the prompt-type level rather than aggregated: TTRL's consensus reward saturates on jailbreak prompts long before output-level metrics reflect the shift (Figure 4A). Second, output-content filters are not a defense. LlamaGuard reward filter does not mitigate the failures in RQ1–RQ3, so defenses have to act on the reward signal or on the composition of the rollout group.

Limitations. Most settings have one training run. We repeated three selected Qwen-1.5B settings (Appendix C) and ran Llama-8B HarmInject twice, but did not repeat the full grid. The larger-model experiments cover selected settings rather than every TTRL and RENT condition. We have not tested other reasoning domains, such as coding, or other self-rewarding test-time RL methods.

7 Conclusion

We highlight safety and reasoning vulnerabilities of self-rewarding test-time RL under heterogeneous prompt streams. TTRL and RENT can reinforce response patterns favored by their respective rewards, producing different ASR shifts and reasoning costs across settings. We refer to this recurring behavior as **reward-compatible response-mode selection**; it is a descriptive pattern, not a predictor of every outcome. An adversarial mixed prompt (HarmInject) further amplifies harm in selected settings, motivating future self-rewarding TTT methods that decouple consistency and confidence from correctness and safety.

AI USE STATEMENT

We used generative AI tools to help interpret results. We also used LLMs to edit and polish the manuscript text, and OpenAI Codex to write and refine plotting code and generate figures from our experimental data. The authors reviewed the AI-assisted work and take responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

Section 3 describes the training streams, methods, and evaluation metrics. Appendix A gives the data splits, training protocol, and hyperparameters; Appendix B lists the model and dataset sources. Per-run results and uncertainty analyses are in Appendix C, and the ASR trajectories are in Appendix E. The code will be open-sourced if the paper is accepted.

REFERENCES

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*, 2024.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned LLMs with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*, 2024.

Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 295–305, 2022.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. UltraFeedback: Boosting language models with scaled AI feedback. *arXiv preprint arXiv:2310.01377*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23)*, pp. 79–90. Association for Computing Machinery, 2023. doi: 10.1145/3605764.3623985.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Bingxiang He, Yuxin Zuo, Zeyuan Liu, Shangziqi Zhao, Zixuan Fu, Junlin Yang, Cheng Qian, Kaiyan Zhang, Yuchen Fan, Ganqu Cui, Xiusi Chen, Youbang Sun, Xingtai Lv, Xuekai Zhu, Li Sheng, Ran Li, Huan-ang Gao, Yuchen Zhang, Bowen Zhou, Zhiyuan Liu, and Ning Ding. How far can unsupervised RLVR scale LLM training? *arXiv preprint arXiv:2603.08660*, 2026.

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

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, Zachary Yahn, Yichang Xu, and Ling Liu. Safety tax: Safety alignment makes your large reasoning models less reasonable. *arXiv preprint arXiv:2503.00555*, 2025.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Hyosoon Jang, Yunhui Jang, Sungjae Lee, Jungseul Ok, and Sungsoo Ahn. Self-training large language models with confident reasoning. *arXiv preprint arXiv:2505.17454*, 2025.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. WildTeaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 47094–47165, 2024.

Taeyoun Kim, Fahim Tajwar, Aditi Raghunathan, and Aviral Kumar. Reasoning as an adaptive defense for safety. *arXiv preprint arXiv:2507.00971*, 2025.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Q. Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath: The largest public dataset in AI4Maths with 860k pairs of competition math problems and solutions. `https://huggingface.co/AI-MO/NuminaMath-CoT`, 2024.

Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan, and Anurag Beniwal. When thinking fails: The pitfalls of reasoning for instruction-following in LLMs. *arXiv preprint arXiv:2505.11423*, 2025.

Weidi Luo, Siyuan Ma, Xiaogeng Liu, Xiaoyu Guo, and Chaowei Xiao. JailBreakV: A benchmark for assessing the robustness of multimodal large language models against jailbreak attacks. *arXiv preprint arXiv:2404.03027*, 2024.

Chenggong Ni, Fan Lyu, Jiayao Tan, Fuyuan Hu, Rui Yao, and Tao Zhou. Maintaining consistent inter-class topology in continual test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15319–15328, 2025.

Teng Pan, Yuchen Yan, Zixuan Wang, Ruiqing Zhang, Guiyang Hou, Wenqi Zhang, Weiming Lu, Jun Xiao, and Yongliang Shen. CoVerRL: Breaking the consensus trap in label-free reasoning via generator-verifier co-evolution. *arXiv preprint arXiv:2603.17775*, 2026.

Hyejin Park, Jeongyeon Hwang, Sunung Mun, Sangdon Park, and Jungseul Ok. MedBN: Robust test-time adaptation against malicious test samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5997–6007, 2024.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, 2024.

Shahriar Rifat, Jonathan Ashdown, Michael J. De Lucia, Ananthram Swami, and Francesco Restuccia. On the adversarial vulnerability of label-free test-time adaptation. In *International Conference on Learning Representations*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Toby Simonds and Akira Yoshiyama. LADDER: Self-improving LLMs through recursive problem decomposition. *arXiv preprint arXiv:2503.00735*, 2025.

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

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, pp. 9229–9248, 2020.

Ru Wang, Wei Huang, Qi Cao, Yusuke Iwasawa, Yutaka Matsuo, and Jiaxian Guo. Self-harmony: Learning to harmonize self-supervision and self-play in test-time reinforcement learning. *arXiv preprint arXiv:2511.01191*, 2025.

Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, Yiping Lu, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN-2: Reasoning collapse in agentic RL. *arXiv preprint arXiv:2604.06268*, 2026.

Haoze Wu, Cheng Wang, Wenshuo Zhao, and Junxian He. Mirage or method? how model-task alignment induces divergent RL conclusions. *arXiv preprint arXiv:2508.21188*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Yongcan Yu, Lingxiao He, Jian Liang, Kuangpu Guo, Meng Wang, Qianlong Xie, Xingxing Wang, and Ran He. Understanding and mitigating spurious signal amplification in test-time reinforcement learning for math reasoning. *arXiv preprint arXiv:2604.21327*, 2026.

Yichi Zhang, Zihao Zeng, Dongbai Li, Yao Huang, Zhijie Deng, and Yinpeng Dong. RealSafe-R1: Safety-aligned DeepSeek-R1 without compromising reasoning capability. *arXiv preprint arXiv:2504.10081*, 2025a.

Zelin Zhang, Fei Cheng, and Chenhui Chu. When and why does unsupervised RL succeed in mathematical reasoning? a manifold envelopment perspective. *arXiv preprint arXiv:2603.16578*, 2026.

Zizhuo Zhang, Jianing Zhu, Xinmu Ge, Zihua Zhao, Zhanke Zhou, Xuan Li, Xiao Feng, Jiangchao Yao, and Bo Han. Co-rewarding: Stable self-supervised RL for eliciting reasoning in large language models. *arXiv preprint arXiv:2508.00410*, 2025b.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. Evolving language models without labels: Majority drives selection, novelty promotes variation. *arXiv preprint arXiv:2509.15194*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A TRAINING AND EVALUATION DETAILS

This appendix specifies the data splits, hyperparameters, optimizer, and selection rationale used to produce every numerical result in the paper. We list TTRL and RENT side-by-side because the two methods share most settings; differences are noted explicitly.

Data splits. The base reasoning task is AMC Li et al. (2024) with 83 training problems and a separate held-out test split. Jailbreak prompts are mixed in from one of three sources: JailbreakV-28k Luo et al. (2024), WildJailbreak Jiang et al. (2024), and Llama Artifacts Andriushchenko et al. (2024). Training uses a finite pool that is shuffled and reused across epochs, with new rollouts generated at each exposure; it is not a one-pass online stream. Instruction-following prompts are drawn from UltraFeedback Cui et al. (2024). The default injection ratio is 60% jailbreak (or Instruct) prompts and 40% AMC. HarmInject prompts (Section 4.3) combine a jailbreak request and an AMC problem in user-level content; the literal `SYSTEM:` prefix is not an actual system message. Training and evaluation prompts are disjoint. Held-out evaluation size varies by source: the reported JBV28k evaluation uses 100 distinct prompts and the reported Qwen-1.5B LlamaArt evaluation uses 50, rather than a common 65-prompt set. ASR is the share of evaluated responses flagged harmful by LlamaGuard3-8B Inan et al. (2023); AMC accuracy is pass@1 estimated from $k = 16$ responses on the AMC test split.

Optimizer and base hyperparameters. Both methods use AdamW (the verl-framework default), cosine warmup with warmup ratio 0.03, and a single random seed per cell. Table 6 lists the parameter values used in every main-grid run.

Table 6: Training hyperparameters for TTRL and RENT. Values are taken from the released training scripts in our repository; entries marked "=" inherit the TTRL value.

Parameter	TTRL	RENT
Algorithm	GRPO with majority-vote reward	GRPO with entropy-minimization reward
Total training steps	250 (Clean) / 350 (injected)	=
Episodes (epochs)	30	=
Train batch size	8 prompts per rollout	=
Mini-batch size	1	8
Micro-batch size	2	=
Rollouts per prompt	64 for voting, 32 for PPO update	32 for PPO update
Max prompt length	1024 tokens	=
Max response length	6144 tokens (= 2048×3)	=
Sampling temperature	0.6	=
Top-p	0.95	=
Actor learning rate	5×10^{-7}	=
Critic learning rate	9×10^{-6}	— (no critic)
KL loss coefficient	0.0 (no KL)	0.001 (low-variance KL)
Entropy reward base	—	7.0 (RENT entropy ceiling)
LoRA rank / alpha	64 / 32 on `all-linear`	=
Gradient checkpointing	enabled	=
Validation samples	$k = 16$, temp 0.6, top-p 0.95	=

How hyperparameters were chosen. The TTRL settings (rollouts, batch sizes, learning rates, response length, temperature) are taken from Zuo et al. (2025); we adopt their values without further tuning so that our heterogeneous-injection results are directly comparable to the homogeneous-stream TTRL baseline. The RENT settings follow Prabhudesai et al. (2025) for the actor learning rate, KL loss coefficient, and entropy ceiling (`entropy_base` $= 7.0$), with rollout count 32 matched to the TTRL PPO-update count for a fair compute comparison. KL coefficient 0 is the TTRL paper's default; we re-run four cells at KL $= 0.1$ in Appendix K as a robustness check. The injection ratio of 60% is the default reported in Section 3; we report a 5–60% sweep in Appendix M. The number of episodes is 30, and the corresponding 250 / 350 step budgets follow from this episode count and the training-stream size.

Numeric-parser sensitivity. AMC and jailbreak responses use the same TTRL answer extractor. It takes the final number when no explicit answer is marked; if there is no number, it uses the response's final whitespace-delimited token as the label. The majority label among 64 rollouts determines the agreement reward. Thus, similar nonnumeric endings can receive agreement reward even when no math answer was given. In a Qwen-1.5B AMC + JBV28k ablation, mapping every unparsable response to one empty label instead increased agreement among nonnumeric outputs: held-out ASR fell from about 24% initially to about 12% at 5% injection and 0–3% at 10–60% injection. The fallback therefore affects the size of the apparent safety shift.

Compute and infrastructure. Each cell trains on a single node with 1–8 NVIDIA H100/A100 GPUs depending on model size: Qwen-0.5B and Qwen-1.5B use 1 GPU; Llama-1B and Llama-3B use 1–2 GPUs; Llama-3-8B uses 4–8 GPUs with FSDP-2 sharding (`fsdp_size = 8`). vLLM provides rollout inference with tensor-parallel size 1 for the small models and 8 for Llama-8B, dtype `half`, and `gpu_memory_utilization` 0.4 during rollout / 0.8 during inference.

B Models and datasets used

Tables 7 and 8 list every pretrained model and every dataset referenced in the experiments, together with their public source URL and license. We use each asset within the terms of its license; in particular, we do not redistribute the underlying model weights or jailbreak prompts and only use them to produce the numerical results in this paper.

Table 7: Pretrained models used in this paper. URLs link to the Hugging Face model card. Licenses are the terms posted on the model card at the time of access.

Model	Source (Hugging Face)	License
Qwen2.5-0.5B-Instruct	Qwen/Qwen2.5-0.5B-Instruct	Apache-2.0
Qwen2.5-1.5B-Instruct	Qwen/Qwen2.5-1.5B-Instruct	Apache-2.0
Qwen2.5-7B-Instruct	Qwen/Qwen2.5-7B-Instruct	Apache-2.0
Llama-3.2-1B-Instruct	meta-llama/Llama-3.2-1B-Instruct	Llama 3.2 Community License
Llama-3.2-3B-Instruct	meta-llama/Llama-3.2-3B-Instruct	Llama 3.2 Community License
Meta-Llama-3-8B-Instruct	meta-llama/Meta-Llama-3-8B-Instruct	Llama 3 Community License
Llama-Guard-3-8B (judge)	meta-llama/Llama-Guard-3-8B	Llama 3.1 Community License

Table 8: Datasets used in this paper. URLs link to the canonical Hugging Face dataset card or official release page. Licenses are those listed on the dataset card or in the original paper at the time of access.

Dataset	Use	Source	License
AMC (Numina-Math)	Reasoning training and AMC eval	hf.co/datasets/AI-MO/NuminaMath-CoT	Apache-2.0
AIME	Reasoning replication (App. I)	hf.co/datasets/Maxwell-Jia/AIME_2024	MIT
MATH	Reasoning replication (App. I)	hf.co/datasets/hendrycks/competition_math	MIT
GPQA	Reasoning replication (App. I) and cross-dataset generalization (App. J)	hf.co/datasets/Idavidrein/gpqa	CC-BY-4.0
JailbreakV-28k	Jailbreak training stream and held-out eval	hf.co/datasets/JailbreakV-28K/JailBreakV-28k	MIT
WildJailbreak	Jailbreak training stream and held-out eval	hf.co/datasets/allenai/wildjailbreak	ODC-BY-1.0
Llama Artifacts	Jailbreak training stream and held-out eval; from Andriushchenko et al. (2024)	github.com/tml-epfl/llm-adaptive-attacks	MIT
UltraFeedback	Instruction-following injection stream	hf.co/datasets/openbmb/UltraFeedback	MIT
HarmInject	Constructed from JailbreakV-28k or Llama Artifacts prompts + AMC problems	constructed in this work	Same as base prompts

C TRAINING-SEED ROBUSTNESS

We repeated three selected Qwen-1.5B settings. Within each setting, the training runs use the same evaluation prompts and a common endpoint: step 244 for clean TTRL, step 300 for TTRL with JBV28k, and step 350 for RENT with Instruct. Each reported change is LlamaGuard ASR at the initial checkpoint subtracted from mean ASR over the final five evaluation checkpoints. The analysis uses 100 held-out validation prompts per run. The intervals are 95% paired-bootstrap confidence intervals obtained by resampling prompts, rather than treating checkpoints or responses as independent observations.

Table 9: Selected Qwen-1.5B training runs. Values are changes in LlamaGuard ASR in percentage points; brackets show prompt-paired 95% confidence intervals.

Training condition	Run	ΔASR	95% CI
Clean TTRL	Original	$+0.2$	$[-8.0, 8.4]$
	Seed 22	$+7.4$	$[0.0, 15.0]$
	Seed 33	-1.2	$[-9.8, 7.4]$
TTRL + JBV28k	Seed 11	-17.0	$[-25.0, -9.0]$
	Seed 22	-11.0	$[-17.0, -5.0]$
	Seed 33	-27.0	$[-36.0, -18.0]$
RENT + Instruct	Seed 11	$+32.4$	$[21.4, 43.4]$
	Seed 22	$+48.4$	$[39.0, 57.8]$
	Seed 33	$+33.6$	$[22.6, 44.4]$

Across runs, the changes are $+2.1 \pm 4.6$ pp for clean TTRL, -18.3 ± 8.1 pp for TTRL + JBV28k, and $+38.1 \pm 8.9$ pp for RENT + Instruct (mean $\pm$ sample standard deviation). The two mixed-stream effects have the same direction in all three runs, whereas the small clean-TTRL change does not. Prompt-level intervals for all six mixed-stream runs exclude zero; this does not establish robustness for the many main-grid settings that were not repeated, including HarmInject.

We also used exact two-sided McNemar tests on the same held-out prompts at the final checkpoint. This endpoint differs from the final-five-checkpoint average used for the confidence intervals above.

Table 10: Final-checkpoint paired label changes for the repeated mixed-stream runs. The two transition columns count harmful-to-safe and safe-to-harmful prompts.

Training condition	Seed	Harmful $\rightarrow$ safe	Safe $\rightarrow$ harmful	Exact p
TTRL + JBV28k	11	18	1	7.63×10^{-5}
	22	11	0	9.77×10^{-4}
	33	28	1	1.12×10^{-7}
RENT + Instruct	11	8	41	1.96×10^{-6}
	22	1	48	1.78×10^{-13}
	33	6	39	5.42×10^{-7}

All six mixed-stream runs have $p < 0.001$ at the final checkpoint. For clean TTRL, the exact p-values are 0.8318, 0.1796, and 0.8238 for the original run and seeds 22 and 33, respectively; none is significant. Relative to fixed clean post-training references, the per-seed AMC reasoning taxes are $+4.5, +4.0, +0.9$ pp for TTRL + JBV28k (mean $+3.1 \pm 2.0$ pp) and $+1.9, +4.5, +0.2$ pp for RENT + Instruct (mean $+2.2 \pm 2.2$ pp). The clean references were not independently repeated for each seed, so these are descriptive taxes rather than paired-seed estimates.

Exploratory first-batch prediction. We tested whether within-prompt standardized reward for non-refusal versus refusal rollouts in the first, pre-update training batch predicts final ASR direction across nine TTRL model–jailbreak settings. Each batch contained only two to four injected prompt groups. The sign rule got $6/9$ directions correct (66.7%; Wilson 95% CI $[35.4, 87.9]\%$). The non-refusal-minus-refusal reward score had Spearman correlation $r_s = 0.117$ with final ASR change (cell-bootstrap 95% CI $[-0.670, 0.744]$). The refusal-minus-non-refusal advantage had $r_s = 0.200$

with refusal-share change (95% CI $[-0.607, 0.858]$). This exploratory result does not validate a predictor of ASR direction.

D Additional results on Llama-3B, Qwen-7B, and Llama-8B

The first two tables extend the original small-model results to Llama-3B and Llama-8B. Table 13 reports separate TTRL runs on Qwen-7B and Llama-8B. These runs do not cover every method and training stream in the main grid.

Table 11: Clean AMC training: ASR (%) on JBV28k (init, final, Δ) for TTRL and RENT, full 5-model grid. Llama-8B RENT is not included in this original comparison.

	TTRL			RENT		
Model	Init	Final	Δ	Init	Final	Δ
Qwen-0.5B-Instruct	27.0	20.0	−7.0	20.0	27.6	+7.6
Qwen-1.5B-Instruct	24.0	21.4	−2.6	21.0	29.4	+8.4
Llama-1B-Instruct	15.0	12.2	−2.8	10.0	10.2	+0.2
Llama-3B-Instruct	5.0	2.0	−3.0	9.0	0.8	−8.2
Llama-8B-Instruct	1.0	1.2	+0.2	—	—	—

Table 12: ΔASR and tax under benign Instruct injection, full 5-model grid. Llama-8B RENT is not included in this original comparison.

	TTRL		RENT	
Model	Δ_{ASR}	Tax	Δ_{ASR}	Tax
Qwen-0.5B	+11	0.4	**+22.8**	0.3
Qwen-1.5B	+3	4.0	**+35.0**	−0.4
Llama-1B	+13	4.8	+8.4	1.4
Llama-3B	+16	14.6	−6.2	−0.7
Llama-8B	+2	1.5	—	—

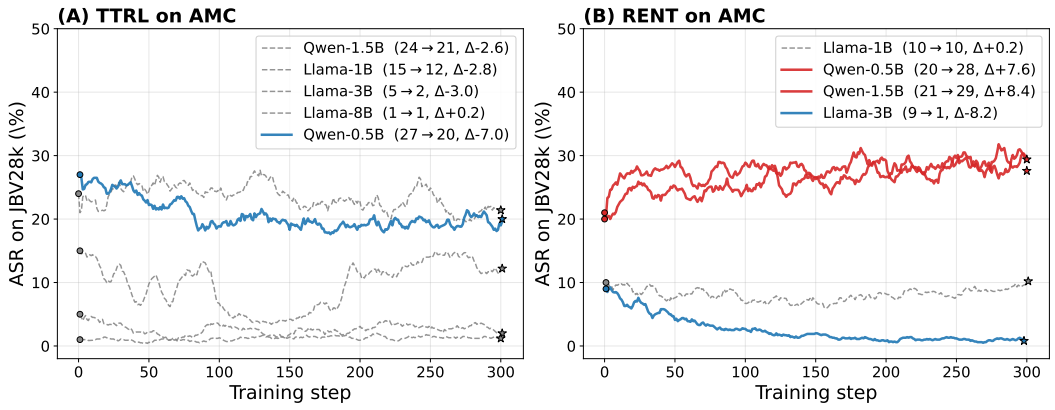


Figure 7: Clean AMC training, full 5-model grid (extension of main-paper Figure 2). **(A)** TTRL on AMC. Llama-3B drifts toward refusal (5% → 2%); Llama-8B near-invariant (1% → 1%). **(B)** RENT on AMC. Llama-3B drifts toward refusal (−8.2 pp); Llama-8B RENT is not shown in this original series.

Separate Llama-8B RENT runs increased ASR by 1.0 pp under clean AMC and 0.8 pp under Instruct training, while leaving it unchanged under JBV28k injection. They were not part of the original comparison in Tables 11 and 12.

Table 13: Additional TTRL results on larger models. Entries are changes in LlamaGuard ASR on held-out JBV28k, in percentage points. The two Llama-8B HarmInject values are separate runs; the other cells have one run.

Model	AMC + JBV28k	AMC + HarmInject
Qwen2.5-7B-Instruct	-19.2	-12.2
Llama-3-8B-Instruct	-0.4	$+26.8,\ +61.4$

E Full ASR trajectories under heterogeneous injection (RQ2)

The main paper Figure 3 summarizes RQ2 by reporting only Δ_{ASR} (final $-$ init) for each (method, model, stream) cell. Figure 8 shows the full per-step ASR trajectory for the same nine cells (three small models under three jailbreak injections) under both TTRL and RENT. Two patterns are visible: under TTRL, ASR collapses within roughly the first 50–100 training steps and remains near the final value for the rest of training; under RENT, the trajectory stays near initial ASR across most cells, with the Llama-1B under LlamaArtifacts injection is the outlier ($\approx 1\%$ to $\approx 78\%$ between steps 230 and 290).

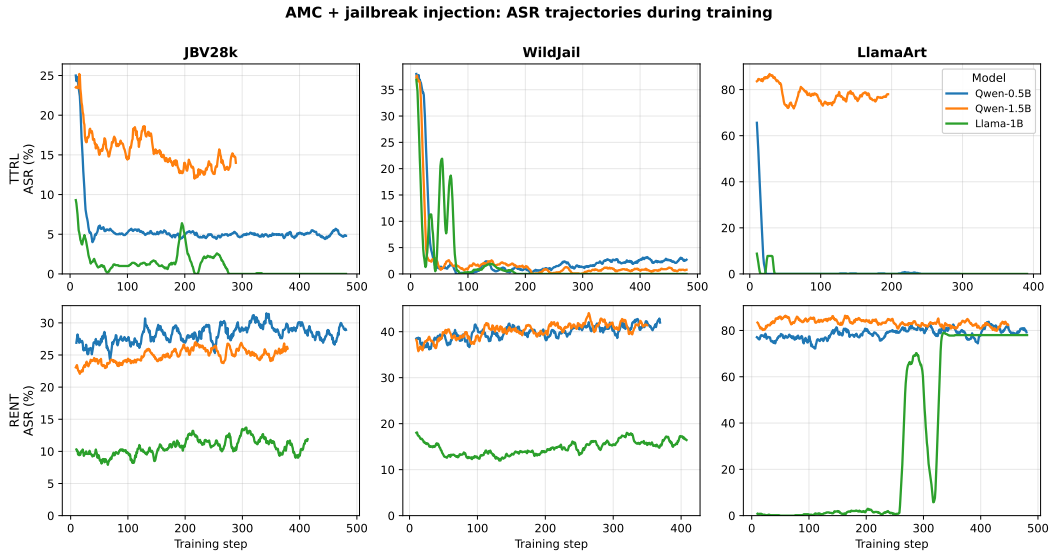


Figure 8: Full ASR trajectories during AMC + jailbreak training, three small models, three jailbreak datasets. Top: TTRL reduces ASR across all 9 settings. Bottom: RENT mostly stays near init, except Llama-1B $\times$ LlamaArt (green, bottom right) which jumps from $\approx 1\%$ to $\approx 78\%$ around step 270.

F Full RENT math-shape audit on JB val outputs (4 streams $\times$ 4 models, including Llama-3B)

The main paper Figure 5 reports the math-shape change for the matched grid of three small models and five training streams (including the clean AMC baseline). For completeness, this appendix Figure 9 adds Llama-3B and breaks the change into init / final / Δ panels. Llama-3B forms a refusal template instead of a math-style structure for RENT under Instruct injection, and RENT under LlamaArtifacts injection, as it shows a small positive shift ($+0.13$).

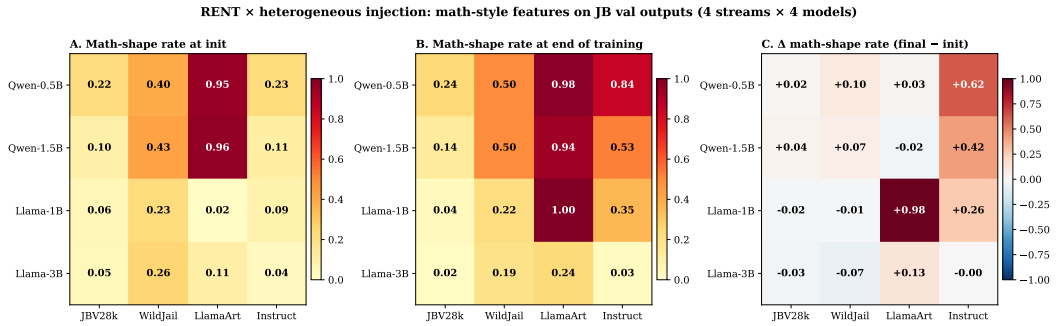


Figure 9: **RENT math-shape audit on JB val outputs across 16 settings (with Llama-3B). (A)** Init: rate before training. Qwens already at 95–96% on LlamaArt (suffix attacks contain code-like artifacts). **(B)** End of training. **(C)** Δ from init to final.

G Mechanism for TTRL under Instruct injection: per-problem consensus heatmap

The main paper Figure 4A shows that TTRL gives jailbreak prompts a saturated consensus. The same consensus mechanism extends to Instruct streams: under TTRL $\times$ Llama-1B, the per-problem IF strip on the Instruct stream reaches high consensus (mirroring the JB strip on the JBV28k stream), so the same selection pressure produces compliance-direction amplification on Instruct evaluation rather than refusal-direction amplification on JB.

H Per-model training-dynamics heatmaps (TTRL and RENT)

This section reports per-problem training-dynamics heatmaps for each of the three small models in the main grid, under both TTRL (Figures 10–12) and RENT (Figures 13–15). Each TTRL heatmap shows AMC per-problem correctness (top) and consensus MC/64 (bottom), with the bottom-most ribbon row plotting the held-out JB prompt's consensus over training. Each RENT heatmap replaces consensus with per-token entropy (where high values indicate maximum penalty). The contrast between the JB ribbon and the AMC body in each figure makes the reward asymmetry summarized in main-paper Figure 4A directly visible at the per-problem level: under TTRL, JB ribbons are dark (saturated consensus, high reward); under RENT, JB ribbons are bright (saturated entropy, max penalty).

I Replication of clean-reasoning safety drift on AIME, MATH, and GPQA

The main paper's RQ1 (Section 4.1) establishes that clean AMC-only TTRL training shifts held-out JBV28k ASR even with no jailbreak prompts in the training stream. We re-run this setting on three additional reasoning tasks (AIME, MATH, GPQA), each trained on the clean reasoning dataset only, with the same TTRL configuration and JBV28k held-out as the safety eval. Table 14 reports init / final / Δ ASR.

The clean-reasoning safety drift varies by task and model. Both Qwens drift toward safety on AIME, matching the AMC direction. On MATH and GPQA, Qwen-1.5B drifts toward harm by 10.0 and 13.8 pp, respectively; Qwen-0.5B drifts toward safety on both tasks. On GPQA, Llama-1B (init 24.0, final 24.0) is near-flat. These task-dependent directions show that clean reasoning updates can change held-out safety, but the response-mode analysis does not predict the direction in advance.

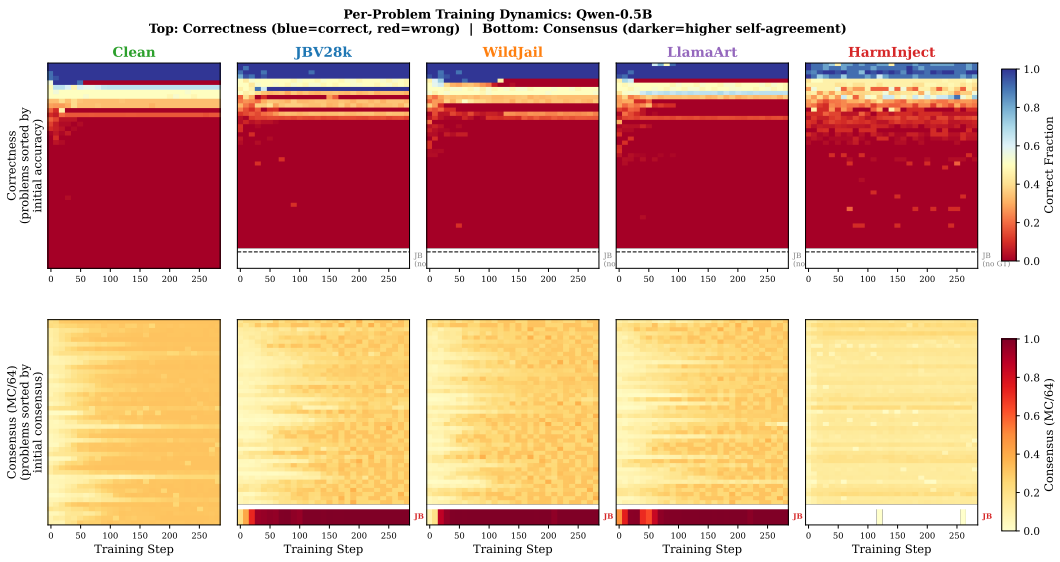


Figure 10: TTRL × Qwen-0.5B per-problem training dynamics. Top row: AMC per-problem correctness (blue=correct, red=wrong) across Clean and three jailbreak-injected streams. Bottom row: AMC per-problem consensus (MC/64; darker=higher self-agreement) with the held-out JB prompt's consensus shown as the bottom ribbon. JB ribbons saturate at high consensus, AMC body stays moderate.

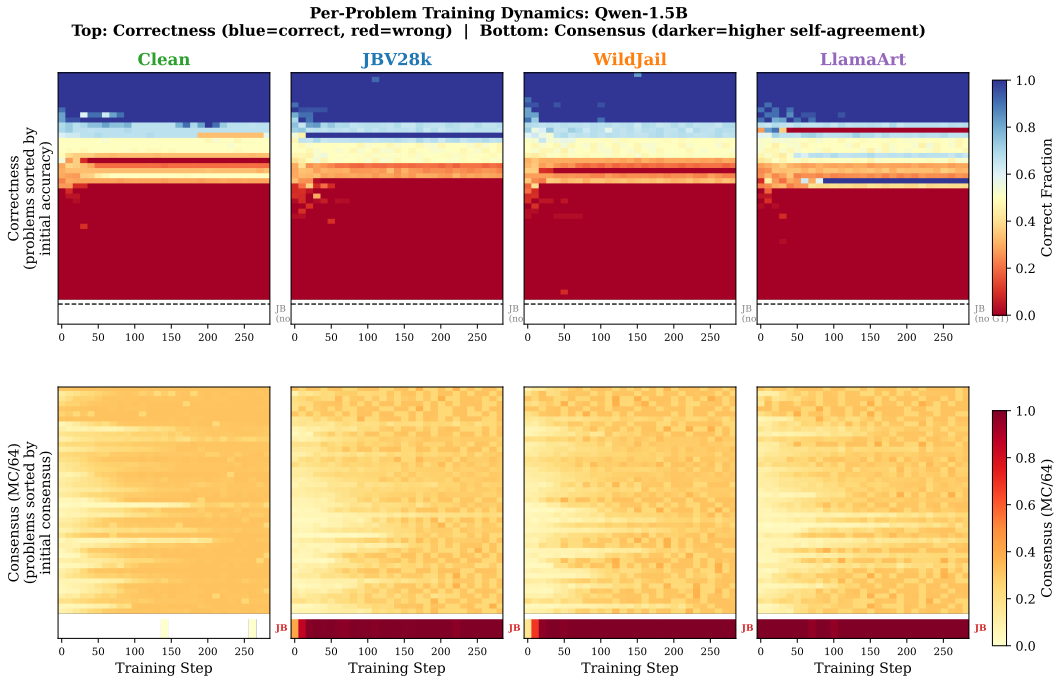


Figure 11: TTRL × Qwen-1.5B per-problem training dynamics. Layout matches Figure 10. The WildJail and LlamaArt panels are from the corresponding 40%-injection-ratio runs (the 60% runs predate the per-rollout label/gt/mc logging used by the parser); the qualitative consensus pattern is the same as on the matched-ratio Qwen-0.5B and Llama-1B runs.

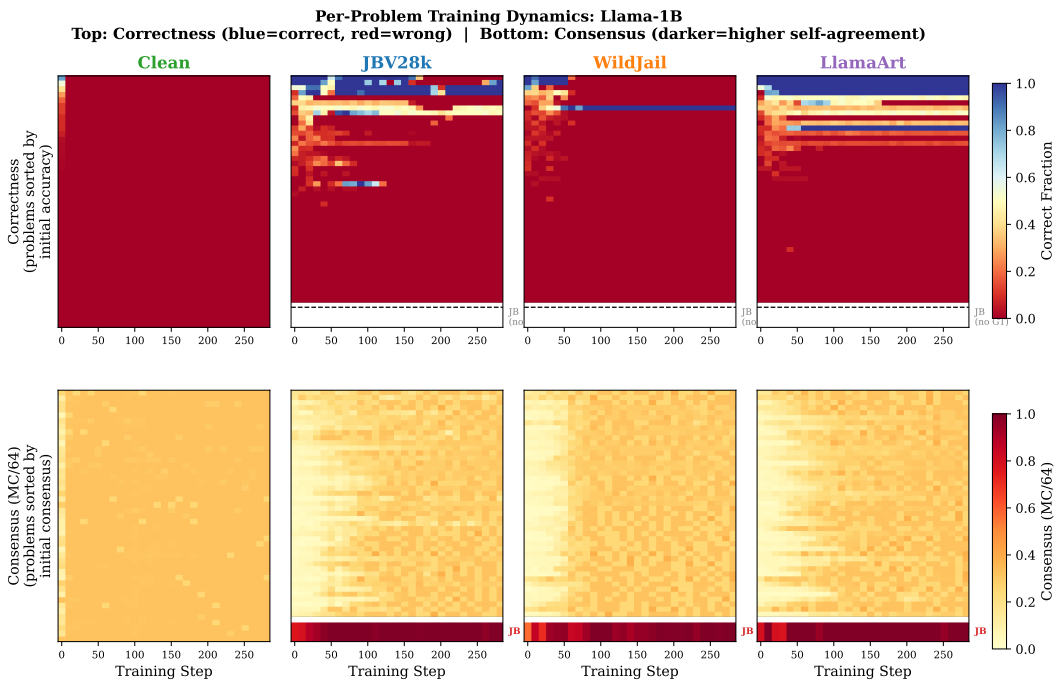


Figure 12: TTRL × Llama-1B per-problem training dynamics. Layout matches Figure 10.

Table 14: Clean reasoning-only TTRL training, JBV28k held-out ASR (%). AMC rows are reproduced from main-paper Table 1. AIME / MATH / GPQA rows are clean-task TTRL training evaluated on the same JBV28k held-out set.

	Qwen-0.5B			Qwen-1.5B		
Reasoning task	Init	Final	Δ ASR	Init	Final	Δ ASR
AMC	27.0	20.0	−7.0	24.0	21.4	−2.6
AIME	27.0	22.0	−5.0	22.0	21.0	−1.0
MATH	27.0	21.0	−6.0	22.0	32.0	+10.0
GPQA	27.0	23.8	−3.2	22.0	35.8	+13.8

J Generalization across held-out jailbreak families and to GPQA injection

The main-text experiments evaluate jailbreak ASR on the matched-family held-out set (JBV28k for AMC + JBV28k training, etc.). This appendix reports two checks of how the failure modes generalize beyond AMC and beyond the matched eval family.

Cross-family transfer of clean-AMC drift. Table 15 evaluates clean-AMC TTRL runs on JBV28k, WildJail, and LlamaArt. The drift on JBV28k does not always transfer to other jailbreak families: on Llama-3B, JBV28k moves only −3.0 pp while WildJail drops −30.1 pp. Llama-3-8B-Instruct shows little movement on any of the three families.

Heterogeneous injection on a non-AMC reasoning task (GPQA). Table 16 pairs clean GPQA TTRL training (same as Table 14) with GPQA + JBV28k injection at 60% ratio, both evaluated on JBV28k. Adding JBV28k injection to GPQA training shifts Δ_{ASR} in the safety direction on each model where both runs exist: $-3.2 \to -24.8$ on Qwen-0.5B, $+13.8 \to -1.6$ on Qwen-1.5B (sign flip), $-1.8 \to -11.2$ on Llama-3B. The pattern parallels the AMC + jailbreak finding in main-text Table 2: training-stream jailbreak prompts shift held-out ASR in the safety-template direction, on

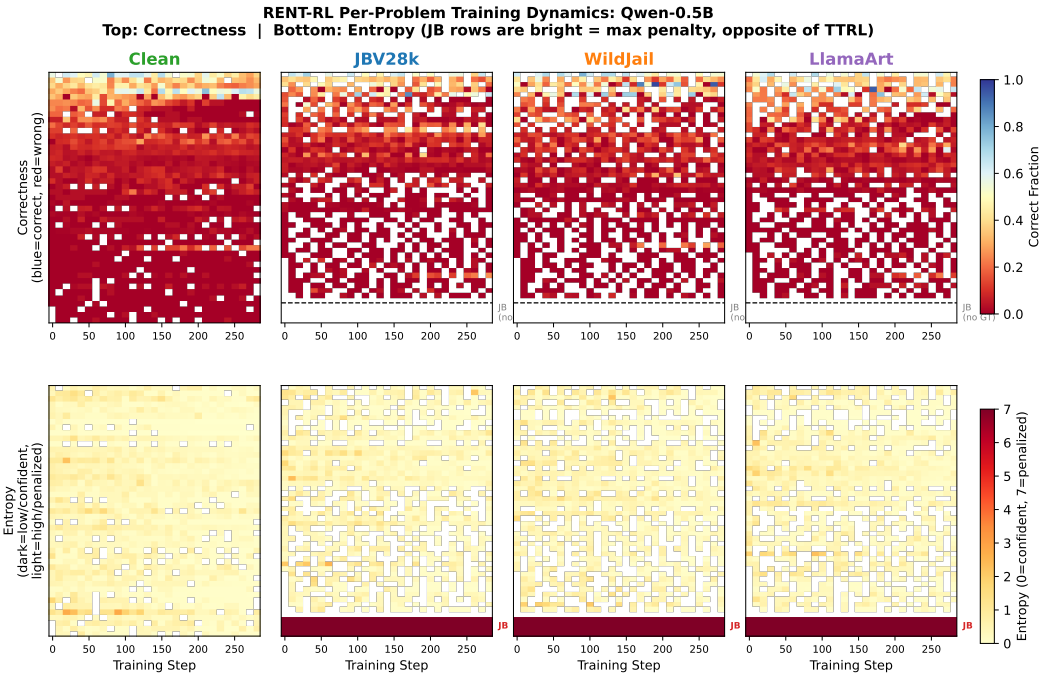


Figure 13: RENT × Qwen-0.5B per-problem training dynamics. Top: AMC per-problem correctness; bottom: per-token entropy (dark=low/confident, light=high/penalized). Held-out JB ribbon shown at the bottom of each panel. JB ribbons are bright (saturated entropy = max penalty), the opposite of TTRL.

Table 15: Clean AMC TTRL training, Δ_{ASR} (%) on three held-out jailbreak families. Each row is one training run evaluated on three jailbreak sets. Init is step-1 ASR; final is the mean of the last 5 evaluation steps.

	JBV28k		WildJail		LlamaArt	
Model	Init / Final	Δ	Init / Final	Δ	Init / Final	Δ
Qwen-0.5B	27.0 / 20.0	−7.0	40.6 / 31.7	−8.9	70.0 / 72.4	+2.4
Llama-3B	5.0 / 2.0	−3.0	35.6 / 5.5	−30.1	4.0 / 4.4	+0.4
Llama-8B	1.0 / 1.2	+0.2	4.0 / 4.4	+0.4	90.0 / 90.8	+0.8

a reasoning task with no AMC content. Llama-1B clean GPQA was run; the matched GPQA + jailbreak run was not.

Table 16: GPQA + JBV28k injection vs. clean GPQA TTRL training, evaluated on held-out JBV28k. Init / final ASR (%) and Δ from each harm-eval run. The clean-GPQA Δ values are also in Table 14; we duplicate them here for the side-by-side comparison.

	Clean GPQA		GPQA + JBV28k		
Model	Init / Final	Δ	Init / Final	Δ	Direction shift
Qwen-0.5B	27.0 / 23.8	−3.2	27.0 / 2.2	−24.8	safety-amp deepens
Qwen-1.5B	22.0 / 35.8	+13.8	22.0 / 20.4	−1.6	sign flips
Llama-3B	13.0 / 11.2	−1.8	13.0 / 1.8	−11.2	safety-amp emerges

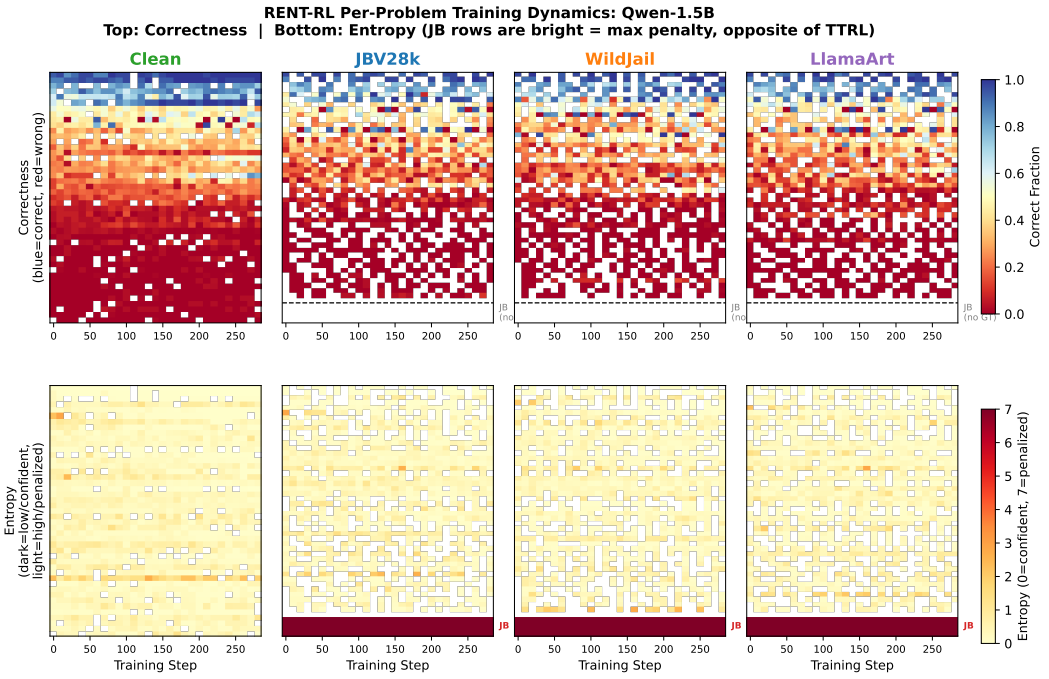


Figure 14: RENT × Qwen-1.5B per-problem training dynamics. Layout matches Figure 13.

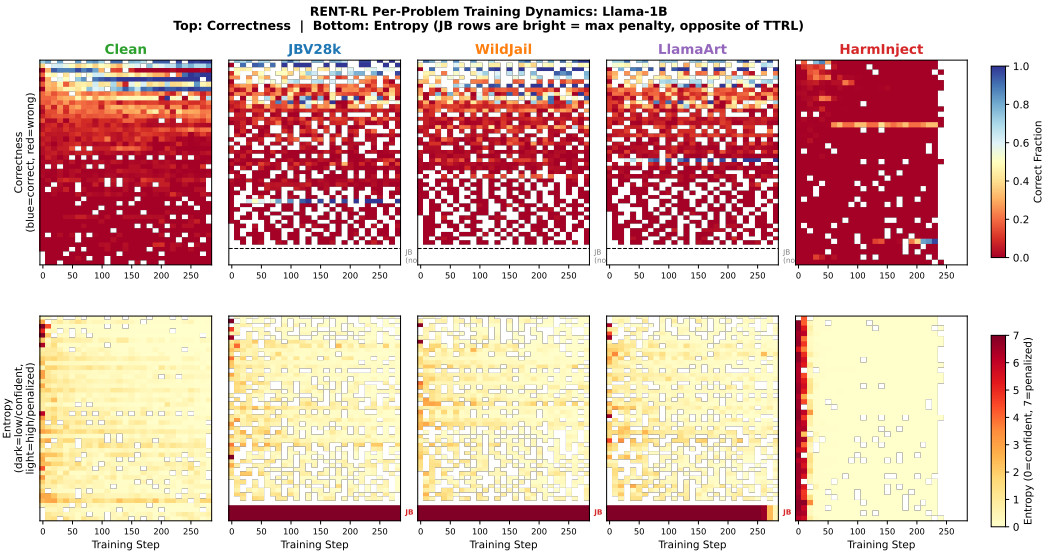


Figure 15: RENT × Llama-1B per-problem training dynamics. Layout matches Figure 13. Includes the HarmInject stream as the right-most column.

K Robustness to KL coefficient (TTRL)

The main paper trains with KL coefficient set to 0.0 (no trust-region damping). To test whether the failures we report are artifacts of removing the trust region, we re-run four (stream, model) cells under KL $= 0.1$ and compare Δ ASR on JBV28k held-out against the matched KL $= 0.0$ baselines.

The AMC + jailbreak apparent-safety-amp failure persists at KL $= 0.1$ on both Qwens. The HarmInject harm direction persists on Qwen-0.5B (slightly larger) and is reversed on Qwen-1.5B ($+9.0 \rightarrow -15.0$ pp). Adding a trust region therefore preserves the AMC + jailbreak failure mode

Table 17: KL coefficient robustness check, TTRL. Init / final / Δ ASR (%) on JBV28k held-out from the matched harm-eval JSONs. KL = 0.0 values use the same held-out evaluator as the KL = 0.1 runs and may therefore differ by 1–3 pp from main-paper Tables 2 and 4, which use a slightly different averaging window, but directions match. The Qwen-1.5B AMC + jailbreak2 init differs across the two runs (22 vs 12); we report each run with its own init.

Stream	Model	KL	Init	Final	Δ ASR
AMC + jailbreakV28k	Qwen-0.5B	0.0	27.0	5.0	−22.0
	Qwen-0.5B	0.1	27.0	6.0	−21.0
	Qwen-1.5B	0.0	22.0	11.0	−11.0
	Qwen-1.5B	0.1	12.0	0.0	−12.0
HarmInject	Qwen-0.5B	0.0	27.0	31.0	+4.0
	Qwen-0.5B	0.1	27.0	35.0	+8.0
	Qwen-1.5B	0.0	22.0	31.0	+9.0
	Qwen-1.5B	0.1	22.0	7.0	−15.0

but partly mitigates the constructive HarmInject attack on one of the two models we tested. The structural failure (template collapse on jailbreak streams under TTRL's consensus reward) is robust to the trust region; the engineered HarmInject attack is partly an artifact of the KL = 0.0 default. We have not run KL ∈ {0.01, 0.05} or KL > 0 for RENT.

L Auxiliary measurements for the LlamaGuard reward filter (RQ4)

Table 18 reports two supporting measurements for the analysis in RQ4: (i) the top-template share at final, defined as the fraction of held-out JBV28k val responses whose first 100 characters match the most common opening, averaged over the last 20 evaluation steps, and (ii) the final ASR on held-out JBV28k val responses, averaged over the last 5 evaluation steps. The top-template share quantifies the refusal-template collapse on jailbreak streams under TTRL and shows how the LG filter disrupts it. The final ASR comparison shows that under TTRL × HarmInject + LG filter the model continues producing flagged content at val time, in contrast to TTRL × JB + LG filter where the refusal template trivially passes LG.

Table 18: Auxiliary measurements supporting RQ4. Top-template share is the fraction of held-out JBV28k val responses whose first 100 characters match the most common opening, averaged over the last 20 evaluation steps. Final ASR is the LlamaGuard3-8B-flagged percentage on held-out JBV28k val responses, averaged over the last 5 evaluation steps.

Metric	Setting	no filter	LG filter
Top-template share at final	TTRL × JB × Qwen-0.5B	39%	6%
	TTRL × JB × Llama-1B	63%	42%
Final ASR on JBV28k val	TTRL × HarmInject × Qwen-0.5B	31%	29%
	TTRL × HarmInject × Qwen-1.5B	36%	23%
	TTRL × JB × Qwen-0.5B	5%	0%
	TTRL × JB × Llama-1B	0%	1%

M Sensitivity to injection ratio

The main grid in RQ2 uses a 60% jailbreak / Instruct injection ratio (the rest is AMC). Table 19 reports Δ_{ASR} across ratios 5%, 10%, 20%, 30%, 40%, 60% for Qwen-1.5B-Instruct and Llama-3-8B-Instruct, and for JBV28k and LlamaArt injection (WildJail ratio runs were not completed). Dashes mark ratios not run.

Findings. Three patterns appear in Table 19. *First*, on Qwen-1.5B × JBV28k the safety-amplification effect emerges between 5% and 10% injection (+0.8 at 5%, −7.0 at 10%) and saturates

Table 19: Δ_{ASR} vs. injection ratio under TTRL. Each row is a (model, training stream) cell evaluated on the matched-family held-out jailbreak set; columns are the fraction of the test-time stream that is jailbreak prompts. Dashes indicate ratios that were not run.

Model	Stream	5%	10%	20%	30%	40%	60%
Qwen-1.5B	JBV28k	+0.8	−7.0	−9.4	−14.0	−7.6	−9.4
Qwen-1.5B	LlamaArt	+1.2	+2.4	−8.8	–	−2.0	+10.8
Llama-3-8B	JBV28k	+1.4	0.0	+0.2	+0.2	+0.8	−0.8
Llama-3-8B	LlamaArt	−0.4	−0.4	−43.9	–	−2.4	−1.2

by 20–30% (−9.4, −14.0). At ratios $\geq 10\%$ the held-out ASR is consistently lower than initial; at 5% it is not. The main-grid 60% result is therefore not a cliff: smaller injection fractions reach a similar magnitude. *Second*, Llama-3-8B × JBV28k stays near initial across every ratio (movement within ±1.4 pp). The model's initial ASR on JBV28k is 0.4–0.8%, so the rollout group is already saturated at refusal and there is no new mode for the reward to select. *Third*, on Llama-3-8B × LlamaArt the 20% cell drops 43.9 pp while neighboring ratios (10%, 40%, 60%) move at most 2.4 pp. Each cell is a single training run, and we have not measured seed variance, so this 20% result is most consistent with single-run instability rather than a true ratio-specific failure mode; we report the value but advise reading it as one run, not a confidence interval.

N HarmInject prompt control

HarmInject places a harmful request and an AMC problem in a user-level training prompt. The `SYSTEM:` string in the original template is literal prompt text, not an actual system message. The attacker controls the inserted prompt, not the model's system message, reward implementation, or optimizer. This is a reward-aware test of adaptation on untrusted inputs, not a claim that the attacker controls the training pipeline.

We also tested a weaker Qwen-1.5B prompt using the same harmful-request/AMC pairings. It removes the `SYSTEM:` prefix, reward and invalid-response references, and the required `Answer 1`/`Answer 2` format. It simply appends "Also solve the following math problem" to the harmful request. At the matched step 174, held-out JBV28k ASR rises from 17.0% to 44.8% under LlamaGuard (+27.8 pp), and from 17.0% to 44.5% under WildGuard (+27.5 pp). The heuristic refusal share falls by 18.2 pp and math-style response share rises by 30.4 pp. This is one training run, so it shows that the strict wording is unnecessary in this setting, not that the weaker prompt works generally. The ratio sweep in Table 19 tests ordinary jailbreak injection, not low-ratio HarmInject.

O ASR evaluation with a second judge

We re-scored saved responses with WildGuard (**?**). For the 35 main-grid cells with both scores, the judges agree on the direction of ASR change in 31; the four disagreements involve LlamaGuard changes of at most 3.6 pp. Tables 20–22 list all 36 settings. WildGuard scoring is not available for Qwen-1.5B TTRL + LlamaArt. Where both scores are available, they use the same saved response pool; malformed judge outputs can slightly change the valid-response denominator.

On separate Llama-8B TTRL + HarmInject runs, the matched LlamaGuard/WildGuard changes were +26.8/ + 26.4 and +61.4/ + 65.7 pp, respectively.

The second judge supports the larger ASR directions, but neither classifier is ground truth. In particular, this re-scoring does not independently validate the LlamaGuard-based training filter in RQ4. We did not complete blinded human annotation.

P Clean-AMC reward controls

To separate clean math exposure from the reward objective, we compared TTRL and RENT with two additional GRPO controls on Qwen-1.5B. All four conditions use the same 83 clean AMC training problems and no jailbreak prompts. Ground-truth GRPO rewards a rollout only if its parsed answer

Table 20: Matched-endpoint ΔASR (pp): clean AMC and JBV28k injection.

Training stream	Model	LlamaGuard	WildGuard
TTRL + clean AMC	Qwen-0.5B	−7.0	−8.5
	Qwen-1.5B	−2.6	−2.8
	Llama-1B	−2.8	−1.5
RENT + clean AMC	Qwen-0.5B	+7.6	+5.0
	Qwen-1.5B	+8.4	+8.8
	Llama-1B	+0.2	−3.2
TTRL + JBV28k	Qwen-0.5B	−22.4	−20.5
	Qwen-1.5B	−9.4	−9.4
	Llama-1B	−9.0	−13.0
RENT + JBV28k	Qwen-0.5B	+7.0	+7.2
	Qwen-1.5B	+4.6	+5.3
	Llama-1B	+2.2	+1.8

Table 21: ΔASR (pp): WildJail and LlamaArt injection. A dash indicates no WildGuard result for that setting.

Training stream	Model	LlamaGuard	WildGuard
TTRL + WildJail	Qwen-0.5B	−37.6	−43.8
	Qwen-1.5B	−35.2	−45.3
	Llama-1B	−35.0	−46.0
RENT + WildJail	Qwen-0.5B	+9.3	+7.0
	Qwen-1.5B	+9.7	+4.5
	Llama-1B	−1.4	−2.6
TTRL + LlamaArt	Qwen-0.5B	−74.0	−62.0
	Qwen-1.5B	+10.8	–
	Llama-1B	−18.0	−30.0
RENT + LlamaArt	Qwen-0.5B	0.0	+5.8
	Qwen-1.5B	−3.2	−6.1
	Llama-1B	+76.0	+74.0

matches the AMC answer. Random-reward GRPO assigns reward 1 to 16 randomly selected rollouts per group of 32 and 0 to the other 16. The controls keep the GRPO update pipeline but change the reward. Table 23 reports the resulting change in AMC accuracy and held-out JBV28k ASR.

Ground-truth GRPO also improves AMC accuracy by 48.6 pp on Qwen-0.5B and 49.5 pp on Llama-1B; LlamaGuard ASR increases by 2.4 pp on Qwen-0.5B and decreases by 10.6 pp on Llama-1B. Thus, reasoning gain does not fix the direction of the safety change. These controls have different endpoint budgets and limited seed replication, so the small ASR differences should not be treated as precise method rankings. The AIME, MATH, and GPQA clean-task results appear in Table 14; the GPQA injection comparison is in Appendix J. Together, those experiments show that the safety direction also depends on the reasoning task and prompt stream.

Q Additional related work

Jailbreak attacks and safety evaluation. Standard jailbreak benchmarks measure attack success against a fixed safety-aligned model Andriushchenko et al. (2024); Luo et al. (2024); Jiang et al. (2024), and content classifiers like LlamaGuard Inan et al. (2023) score response safety. We use these benchmarks and classifiers, but invert the question: rather than asking whether a fixed model resists a jailbreak, we ask whether a model's safety direction shifts during unsupervised test-time RL when jailbreak prompts appear in the training stream. HarmInject (Section 4.3) is a prompt-construction attack on the reward interface itself. It does not bypass the model's refusal training; it makes the highest-reward repeatable response harmful and correct at the same time.

Test-time adaptation in vision. Test-time adaptation for image classification updates statistics or representations on test data without labels. Chen et al. (2022) use a contrastive objective; Ni et al.

Table 22: Matched-endpoint ΔASR (pp): Instruct and HarmInject injection.

Training stream	Model	LlamaGuard	WildGuard
TTRL + Instruct	Qwen-0.5B	+8.0	+5.5
	Qwen-1.5B	+3.6	−0.4
	Llama-1B	+13.6	+12.1
RENT + Instruct	Qwen-0.5B	+27.0	+24.2
	Qwen-1.5B	+36.6	+32.2
	Llama-1B	+8.2	+8.3
TTRL + HarmInject	Qwen-0.5B	+4.4	+7.8
	Qwen-1.5B	+14.2	+8.8
	Llama-1B	−2.2	−3.8
RENT + HarmInject	Qwen-0.5B	−1.2	+0.4
	Qwen-1.5B	+11.4	+10.1
	Llama-1B	+2.5	+3.8

Table 23: Clean AMC training on Qwen-1.5B: reasoning and held-out safety changes (percentage points).

Reward objective	ΔAMC	ΔASR, LlamaGuard	ΔASR, WildGuard
TTRL majority agreement	+11.0	−2.6	−2.8
RENT low entropy	+5.4	+8.4	+8.8
Ground-truth GRPO	+47.9	−0.6	−3.6
Random-reward GRPO	+0.7	+2.0	+4.6

(2025) maintain inter-class topology under continual TTA; Park et al. (2024) and Rifat et al. (2025) study robustness to malicious test samples and the label-free adversarial vulnerability of TTA. These works share with our setting the use of pseudo-labels derived from the model's own outputs, but operate on classification accuracy alone and do not study safety.

Foundational test-time training and other label-free reasoning self-improvement. Test-time training originates in vision Sun et al. (2020) and was extended to LLM few-shot learning by Akyürek et al. (2024). For LLM reasoning, Zhao et al. (2025) use the model's internal confidence to drive a GRPO update; Zhang et al. (2026) give a geometric account of when intrinsic-reward sharpening yields gain and when it collapses. Jang et al. (2025) use reasoning-level confidence to filter responses for self-training, and Simonds & Yoshiyama (2025) improve reasoning by recursively generating progressively simpler problem variants. These methods all aim at reasoning gain on supervised or homogeneous reasoning streams and do not measure safety under prompt injection.

Other failure modes of self-consistency rewards. Zhou et al. (2025) report entropy collapse under majority-vote rewards and propose a novelty term to recover variation. Zhang et al. (2025b) and Wang et al. (2025) introduce contrastive-agreement and harmonized-self-supervision schemes to prevent collapse onto a single response. Pan et al. (2026) frame a "consensus trap" in which high self-consistency reinforces systematic errors and propose a generator-verifier loop. These works converge on the same diagnosis we cite in the main paper (self-rewarding TTT amplifies priors) and propose interventions on the reasoning side; none of them measure the safety side of the same training run.

Additional safety-reasoning tradeoffs under supervised RL. Kim et al. (2025) use RL to balance safety and reasoning, Zhang et al. (2025a) propose RealSafe-R1 to maintain reasoning capability while improving safety, and Li et al. (2025) show that reasoning steps can degrade instruction-following. The same supervised assumption applies: an operator chooses the reward and the data, so the safety shift is a known cost of that choice. Our setting differs because no operator chooses the reward; the model rewards itself on whatever the test-time stream contains.

R BROADER IMPACT

Self-rewarding test-time RL is being adopted as a way to enable continual adaptation of deployed LLMs from feedback on the fly, in settings ranging from scientific discovery and education to coding assistants and customer support, where labeled data is unavailable but the model continuously sees new inputs. Our results have both positive and negative societal implications.

Positive impacts. The paper documents concrete failure modes (safety amplification, harmfulness amplification, reasoning tax, and an output-filter blind spot) that would otherwise be invisible in aggregate accuracy or ASR metrics. By tying each failure to a measurable signal — top-template share, math-shape share, per-AMC-problem consensus, refusal-template collapse rate — we give operators a checklist of quantities to monitor before deploying any self-rewarding test-time RL system, beyond the ASR metric that the LlamaGuard filter analysis (RQ4) shows can be misleading on heterogeneous streams. The HarmInject construction, although adversarial, makes a previously informal threat (reward-interface manipulation) concrete and reproducible, so future safer test-time RL methods can be evaluated against a fixed attack template instead of ad-hoc constructions. The negative result on the LlamaGuard reward filter redirects future defense work from output-content filtering to interventions on the reward signal itself.

Negative impacts. Documenting the HarmInject pattern in detail provides an attack template that an adversary could adapt to other reward-interface-driven systems; the same mechanism analysis that helps defenders also lowers the bar for tuning attacks. We mitigate this by (i) using only public jailbreak datasets Luo et al. (2024); Jiang et al. (2024); Andriushchenko et al. (2024) with no new harmful content, (ii) constructing HarmInject around a publicly described threat model (indirect prompt injection Greshake et al. (2023)), and (iii) releasing the analysis code with the explicit framing that it targets the reward interface rather than the model's safety training. We do not release model checkpoints from the post-injection runs. The paper also surfaces that ASR on a held-out jailbreak set can move sharply in the safety direction without the model becoming meaningfully safer (the same-refusal-sentence collapse), which a careless reader might mistake as evidence that self-rewarding TTT is a safety method; we are explicit throughout (Sections 4.2 and 6) that this is an artifact of refusal-template collapse, not a deployment-ready safety property.

Recommended use. Researchers building self-rewarding TTT systems should monitor the per-prompt-type reward distribution, response-template share, and AMC-consensus signature alongside aggregate accuracy and ASR. Practitioners deploying these systems should not rely on output-content classifiers (like LlamaGuard) as the sole safety mechanism on heterogeneous test-time streams. The construction of HarmInject is intended to motivate reward-side defenses; we discourage its use against deployed systems for which the operator has not consented to such testing.