When Does Execution Feedback Transfer?
Minimal-Sufficient Feedback for Internalized RLVR

Yucheng Yao
ycx@stanford.edu

CS224R Spring 2026 Custom Project

Extended Abstract

Reinforcement learning from verifiable rewards (RLVR) is attractive for mathematical reasoning because a deterministic verifier can score large numbers of model samples without human labels. However, binary correctness rewards are sparse and can fail as credit-assignment signals for group-relative methods: when every sampled completion for a prompt receives the same reward, the group supplies no relative advantage signal even if the completions fail in different ways. This project studies that failure mode in a controlled Countdown arithmetic setting and asks whether verifier-generated execution feedback can supply a minimally sufficient correction signal that later transfers into a no-feedback policy.

We use Qwen2.5-0.5B with a 1,000-example LoRA supervised fine-tuning (SFT) warm start. The main diagnostic evaluates 500 Countdown prompts with $K = 8$ samples per prompt and a deterministic verifier that checks answer format, arithmetic value, and valid number use. The SFT policy is not completely incapable: pass@1 is 0.086 and pass@8 is 0.408. Nevertheless, 59.2% of prompt groups are all-fail and therefore zero-informative for group-relative binary RLVR, while reward contrast appears in only 15.0% of sampled pairs. This confirms the proposal's central concern: binary reward sampling can spend substantial compute producing no within-prompt learning signal.

We then compare feedback-conditioned repair on failed attempts. The feedback taxonomy includes generic revision feedback (F1), error-class feedback (F2), localized execution feedback (F3), targeted WRONG_TARGET feedback (F4), shuffled localized feedback, and a same-budget fresh resampling control. In the expanded 800-example repair run, same-budget resampling achieves the best immediate repair success rate (0.0688), ahead of F3 localized feedback (0.0475), F1 generic feedback (0.0425), shuffled feedback (0.0375), F2 error-class feedback (0.0288), and F4 targeted feedback (0.0250). This is a negative but useful result: apparent repair gains from textual feedback must be compared against revision, shuffled-feedback, and resampling controls before being interpreted as evidence of better credit assignment.

Finally, we test whether successful trajectories internalize into no-feedback policies through self-distillation, and we run small binary and shaped RLOO baselines. Binary RLOO improves pass@8 from 0.408 to 0.438 but leaves pass@1 nearly unchanged; shaped RLOO improves pass@1 to 0.120 but does not improve pass@8. Both retain high zero-advantage group rates during training. Self-distillation is strongest for successful resampling: SD-resample reaches pass@1 0.168, while SD-F4 reaches pass@1 0.116 and pass@8 0.414. Overall, our evidence supports a diagnostic/control contribution rather than a broad positive claim about feedback. Binary RLVR exhibits a strong zero-signal failure mode, and in this Countdown setup successful trajectory selection transfers more clearly than targeted textual execution feedback.

1 Introduction

Verifiable rewards provide an appealing route for training reasoning models. In domains such as arithmetic, code execution, theorem checking, and tool use, an automatic verifier can produce outcome labels without direct human preference supervision. This makes RLVR a natural counterpart to RLHF [1] and policy-gradient alignment methods such as PPO [2]. Yet verifiability does not automatically imply dense credit assignment. A verifier can tell us that an answer is wrong without telling us which local decision caused the failure.

The issue is especially sharp for group-relative policy updates. If a policy samples several completions for the same prompt and every completion receives reward zero, then leave-one-out or group-normalized advantages collapse to zero. The training batch may contain diverse reasoning traces, but the binary reward provides no relative information within the group. We call such groups *zero-informative*. This project studies how often that phenomenon occurs and whether execution feedback can reduce its practical impact.

Our research question is:

> *Which verifier-generated feedback, if any, is minimally sufficient for repair, and does successful repair transfer into a no-feedback policy?*

The original proposal hypothesized that localized execution feedback would outperform generic or shuffled feedback and that successful feedback-conditioned repairs could be distilled into a single-turn policy. Our current results refine that hypothesis. The zero-signal diagnostic is strongly supported. The feedback-specific claim is not: same-budget resampling is a stronger repair and internalization control than the textual feedback variants tested so far.

Contributions. First, we implement a Countdown RLVR diagnostic that measures pass@K, all-fail groups, all-correct groups, zero-informative group rate, and reward contrast. Second, we define a feedback/control taxonomy that separates generic revision, error-class feedback, localized execution feedback, targeted feedback, shuffled feedback, and fresh resampling. Third, we compare SFT, repair, self-distillation, binary RLOO, and shaped RLOO on the same 500-prompt evaluation split. Fourth, we show that a resampling control materially changes the interpretation of feedback-conditioned repair.

2 Related Work

RLHF and policy optimization. RLHF trains language models from human preference signals, commonly using reward modeling followed by policy optimization [1]. PPO remains a standard policy-gradient method for stable updates under a KL constraint [2]. For reasoning tasks, however, human preference data may be unnecessary when a verifier can directly judge final correctness.

Verifiers for reasoning. Verifier-based mathematical reasoning has been studied in settings where generated solutions can be checked or ranked by a learned or deterministic verifier [3]. RLVR extends this idea by using verifier rewards as reinforcement signals. Such rewards scale well, but binary correctness can be sparse and can hide local failure structure.

Group-relative RL and zero-advantage batches. Recent math-reasoning RL systems use group-level comparisons to improve policy optimization efficiency, notably DeepSeekMath's GRPO [4]. Group-relative methods reduce variance by comparing samples for the same prompt. This is useful only when the reward distribution within a group has contrast. In Countdown, all-fail groups are common even after an SFT warm start. Our work operationalizes this issue with explicit zero-informative group and reward-contrast metrics.

Feedback, refinement, and distillation. Self-improvement and refinement methods train models from generated rationales or feedback-conditioned revisions [5, 6]. Our setting differs by using a deterministic verifier rather than an LLM judge, and by including shuffled-feedback and resampling controls. This lets us ask whether feedback text adds information beyond giving the model another decoding attempt.

3 Method

3.1 Countdown Environment and Verifier

Each Countdown prompt provides a small set of numbers and a target value. A completion must output an arithmetic expression using the provided numbers exactly as required. The verifier parses the answer, checks format, validates number use, evaluates the expression, and assigns both a binary correctness reward and a structured failure type. We track common failure classes such as WRONG_TARGET, NUMBER_REUSED, NUMBER_OMITTED, FORMAT_MISSING, PARSE_FAIL, and ARITHMETIC_ERROR.

3.2 Zero-Signal Diagnostic

For each prompt, we sample K completions from the policy and compute:

- pass@1 and pass@K;
- average verifier score;
- all-fail and all-correct group rates;
- zero-informative group rate, where all samples in a group share the same binary reward;
- reward contrast rate, the fraction of within-group sample pairs with different binary reward.

The diagnostic is designed to answer whether binary RLVR has useful within-prompt contrast, not merely whether the model occasionally solves the task.

3.3 Feedback Taxonomy

We repair failed SFT attempts under six conditions:

- **F1 generic:** a short instruction that the answer is incorrect and should be retried. This tests whether revision alone is enough.
- **F2 error-class:** a verifier-derived natural language description of the failure class. This should help if the model can condition on a coarse error label.
- **F3 localized:** feedback containing local execution information, such as the evaluated value and target. This was expected to be stronger than F2 because it exposes a concrete verifier trace.
- **F4 targeted:** a longer WRONG_TARGET-specific message including the evaluated value, target, delta, and an instruction to search for a different expression. This was expected to be strongest on the dominant WRONG_TARGET failures.
- **F-shuffled:** localized feedback from another failed attempt, preserving style but breaking semantic alignment. This controls for the presence of feedback-like text without correct credit assignment.
- **R-resample:** a same-budget fresh attempt without previous completion or feedback. This controls for the possibility that a second attempt succeeds through sampling variance.

Before running the repair experiment, the intended informativeness ordering was F4/F3 > F2 > F1, with F-shuffled and R-resample serving as controls. The actual templates are shown in Table 1. Bracketed fields are filled with the current Countdown problem, previous attempt, or verifier trace.

Table 1: Prompt and feedback message templates used in the repair experiments.

Component	Template or message
Solve prompt	"You are solving a Countdown arithmetic puzzle. Numbers: [numbers]. Target: [target]. Use each provided number exactly once. You may use +, -, *, and /. Show your reasoning briefly. Put your final expression inside ⟨answer⟩. . . ⟨/answer⟩."
Repair wrapper	"Problem: Numbers: [numbers]. Target: [target]. Previous attempt: [completion]. Verifier feedback: [feedback]. Revise the solution. Use each provided number exactly once. Put the final expression inside ⟨answer⟩. . . ⟨/answer⟩."
F1 generic	"The answer is incorrect. Try again."
F2 error-class	One of several coarse verifier messages, e.g. "The final expression is not parseable," "The equation uses the input numbers incorrectly," or "The equation is valid but does not reach the target."
F3 localized	Verifier-localized messages, e.g. "Your expression evaluates to [value], but the target is [target]," "Your equation reuses [numbers] and omits [numbers]," or "The expression inside ⟨answer⟩. . . ⟨/answer⟩ is not parseable: [parser message]."
F4 targeted	For WRONG_TARGET: "The expression is syntactically valid and uses the allowed numbers exactly once, but it evaluates to [value] instead of the target [target]. It must change the final value by [delta]. Search for a different expression over [numbers] that reaches [target]; do not just reformat." For other failure types, F4 backs off to F3.
F-shuffled	Uses the F3 message format, but the localized feedback string is taken from a different failed attempt.
R-resample	Uses the solve prompt plus: "Generate a fresh solution attempt. Do not copy any previous expression."

All repair conditions use the same failed attempts and decoding budget. The resampling condition is important because any second attempt can improve purely through sampling variance.

3.4 Internalization by Self-Distillation

For each condition, we filter successful repaired or resampled completions and train a no-feedback LoRA adapter from those successful trajectories. We then evaluate the resulting policy without feedback on the same 500-prompt, $K = 8$ evaluation split. This tests whether the successful behavior is internalized rather than only produced in a feedback-conditioned context.

3.5 RLOO Baselines

We run binary and shaped RLOO baselines from the SFT warm start. The binary baseline uses correctness reward. The shaped baseline uses verifier-derived partial credit. During training, we track mean reward and zero-advantage group rate, then evaluate the trained adapter on the same no-feedback 500-prompt split.

4 Experimental Setup

The base model is Qwen2.5-0.5B [7]. We first train a LoRA [8] SFT warm start on 1,000 synthetic Countdown solutions. The main diagnostic uses 500 prompts and $K = 8$ samples per prompt. The expanded repair experiment uses 800 failed attempts sampled from the SFT diagnostic rollouts, with the same failed-attempt set used across feedback/control conditions.

Binary and shaped RLOO baselines use 500 generated training prompts, 60 update steps, $K = 4$ samples per prompt group, and post-training evaluation on 500 prompts with $K = 8$. Self-distillation adapters are trained from successful repairs or resamples and evaluated with no feedback under the same 500-prompt, $K = 8$ protocol. All experiments were run on Modal and all metrics below are computed from deterministic verifier outputs.

5 Results

5.1 Binary Rewards Often Provide No Relative Signal

Figure 1 shows the central diagnostic. The SFT policy has pass@1 0.086 and pass@8 0.408, so the task is not impossible. However, 59.2% of prompt groups are all-fail and zero-informative under binary rewards. The reward

contrast rate is only 0.1498.

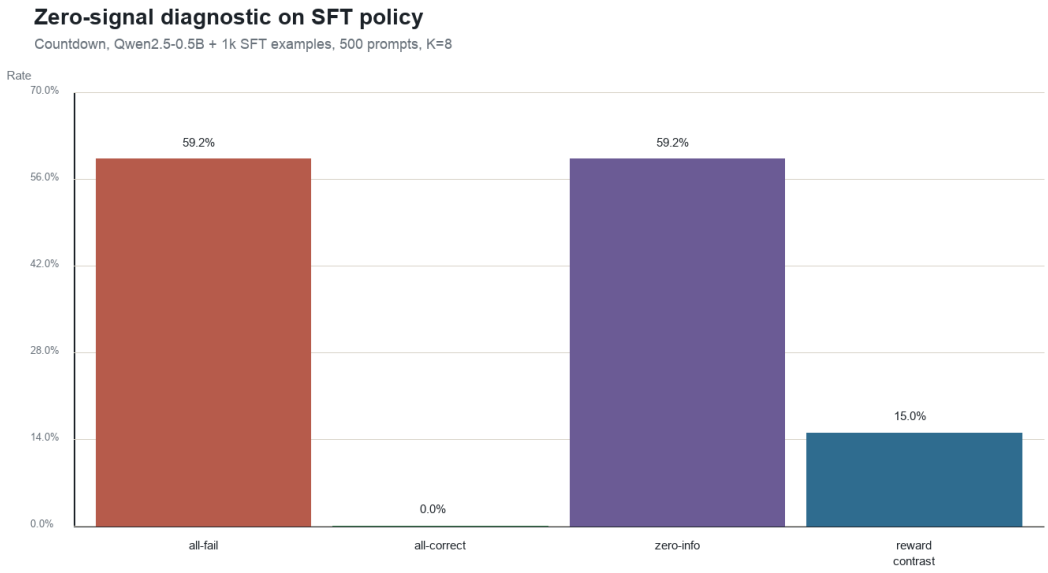

Figure 1: Zero-signal diagnostic for the SFT warm-start policy. Most groups are all-fail, even though pass@8 is nonzero.

This supports the proposal's main motivation. Binary RLVR can fail not because the verifier is wrong, but because the sampled group contains no reward contrast. In such batches, group-relative advantages cannot distinguish better and worse failed attempts.

5.2 Feedback Repair Requires Strong Controls

Table 2 and Figure 2 summarize the expanded repair run. The strongest immediate repair condition is same-budget resampling, with a success rate of 0.0688. F3 localized feedback is the strongest textual feedback condition at 0.0475, but it is only modestly above F1 generic feedback and shuffled feedback. F4 targeted feedback underperforms all major controls despite being much longer.

Table 2: Expanded repair v2 over 800 failed attempts per condition.

Condition	Success	Count	New error	Avg. feedback tokens
F1 generic	0.0425	34	0.0338	6.00
F2 error-class	0.0288	23	0.0312	9.92
F3 localized	0.0475	38	0.0325	10.03
F4 targeted	0.0250	20	0.0888	46.20
F-shuffled	0.0375	30	0.0550	10.02
R-resample	0.0688	55	0.2388	0.00

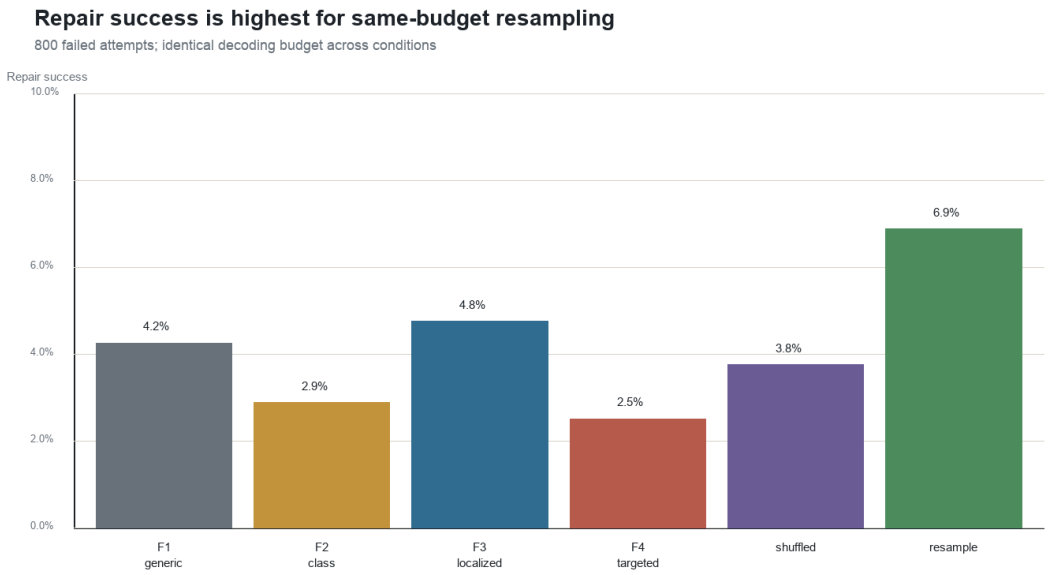

Figure 2: Repair success by feedback/control type. Same-budget resampling is the strongest immediate repair condition.

The result changes the project interpretation. The original hypothesis predicted that localized execution feedback would cleanly dominate generic and shuffled feedback. Instead, we observe a weaker textual-feedback effect and a stronger resampling effect. This does not invalidate the zero-signal diagnosis; rather, it shows that feedback-specific credit-assignment claims require stricter controls.

A small seed-repeat robustness check reaches the same qualitative conclusion. We reran the repair experiment at smaller scale on two additional seeds, using 200 prompts and 250 failed attempts per condition. R-resample remained the best condition in both repeats, with success rates 0.096 and 0.084. Pooled across the main run and the two repeats, R-resample succeeds on 100 of 1300 matched failed attempts (0.0769), compared with 56 of 1300 for F3 localized feedback (0.0431). A paired approximate McNemar check on matched failed attempts favors R-resample over F3. Thus, same-budget resampling is a robust necessary control in this setup, although this does not prove that all textual feedback is useless; F2 and F4 vary across seeds and feedback-template quality remains a limitation.

5.3 Binary and Shaped RLOO Baselines

Table 3 compares no-feedback policies on the shared evaluation split. Binary RLOO improves pass@8 from 0.408 to 0.438 but leaves pass@1 nearly unchanged. Shaped RLOO improves pass@1 to 0.120 but does not improve pass@8. Figure 3 shows that both runs retain high zero-advantage group rates during training.

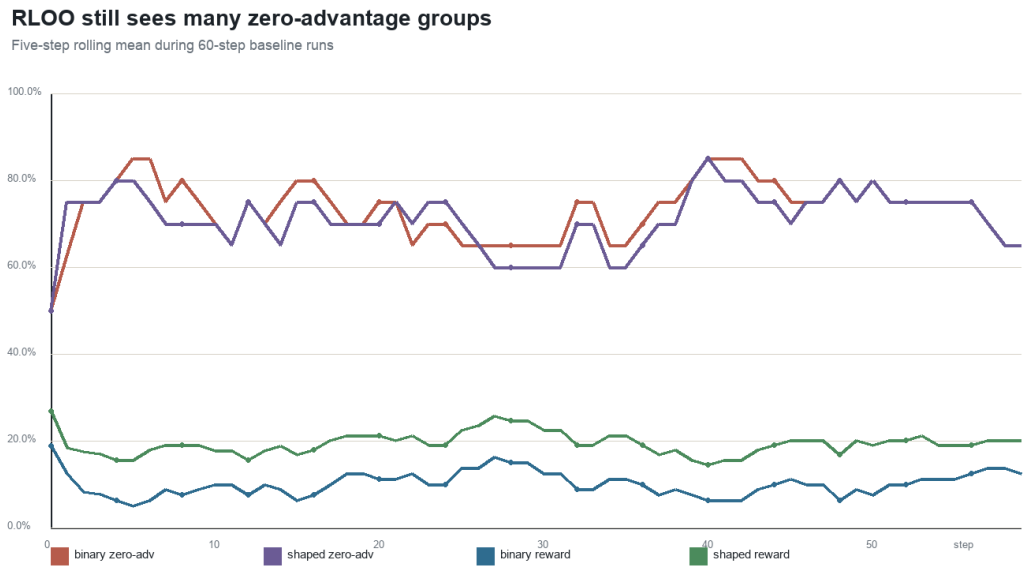

Figure 3: RLOO training diagnostics. Reward improves modestly, but zero-advantage groups remain common.

The baselines are therefore consistent with the diagnostic. RL can make small improvements, but the sparse binary signal still frequently collapses to no within-group advantage. Shaped reward partially changes the pass@1/pass@8 tradeoff but does not remove the zero-signal issue.

5.4 Internalization Results

Self-distillation tests whether successful feedback-conditioned or resampled trajectories transfer into no-feedback behavior. The strongest pass@1 result comes from successful resampling, not targeted feedback. SD-resample reaches pass@1 0.168, while SD-F4 reaches pass@1 0.116 and pass@8 0.414.

Table 3: No-feedback policy evaluation on the same 500-prompt, $K = 8$ split.

Policy	Training signal	pass@1	pass@8	Avg.	All-fail	Zero-info	Contrast
SFT	1k SFT examples	0.086	0.408	0.1893	0.592	0.592	0.1498
Binary RLOO	binary reward	0.088	0.438	0.2037	0.562	0.562	0.1675
Shaped RLOO	shaped reward	0.120	0.406	0.1995	0.594	0.594	0.1590
SD-F3	successful F3 repairs	0.108	0.402	0.2105	0.598	0.600	0.1604
SD-shuffled	successful shuffled repairs	0.120	0.370	0.1999	0.630	0.634	0.1399
SD-F4	successful F4 repairs	0.116	0.414	0.2132	0.586	0.594	0.1586
SD-resample	successful resamples	0.168	0.406	0.2395	0.594	0.616	0.1556

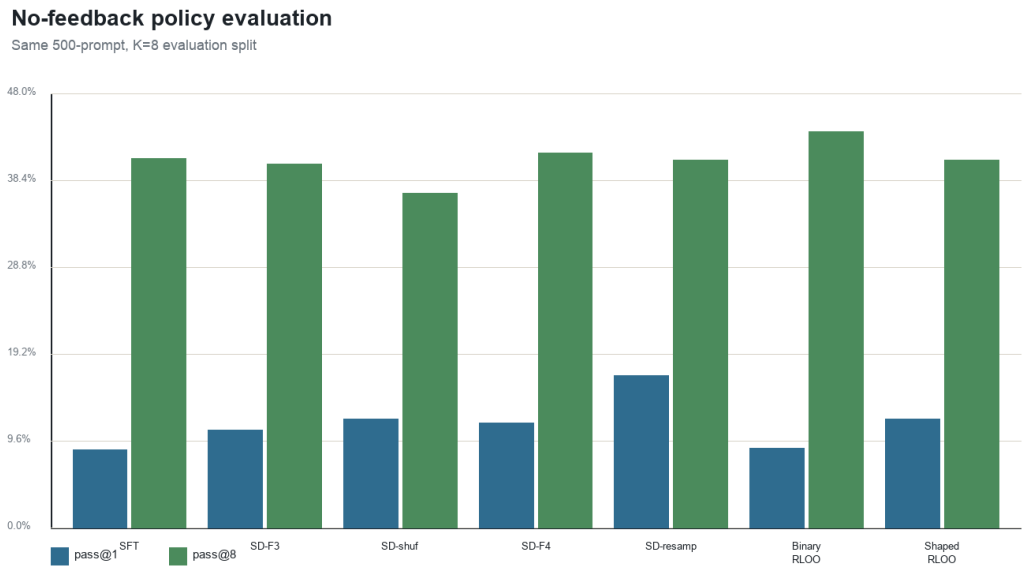

Figure 4: No-feedback pass@1 and pass@8 across SFT, RLOO, and self-distilled policies. SD-resample gives the strongest pass@1 gain.

These results support a cautious internalization claim: successful trajectories can be distilled into no-feedback policies, but current evidence suggests that trajectory selection through resampling is more transferable than targeted feedback text.

5.5 Failure Analysis

The SFT diagnostic failures are dominated by WRONG_TARGET answers. As shown in Figure 5, WRONG_TARGET accounts for 3535 of 3603 failures, approximately 98.1%. This motivated F4 targeted feedback, but the negative F4 result suggests that simply telling the model how far its computed value is from the target is not enough for a small policy to recover.

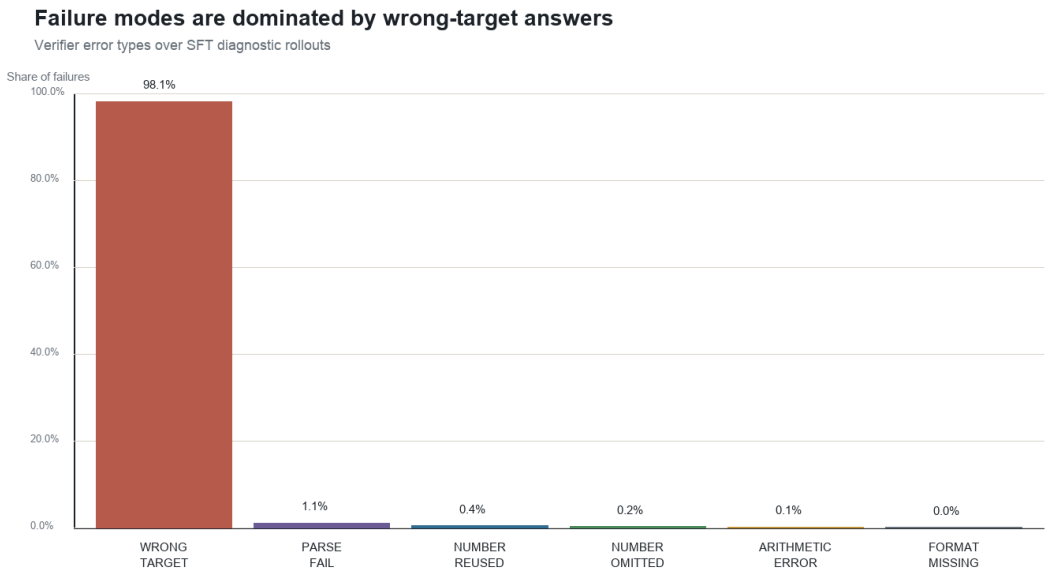

Figure 5: Failure-type breakdown for SFT diagnostic rollouts. WRONG_TARGET dominates the observed failures.

6 Discussion

The experiments support the strongest part of the original proposal: binary RLVR often lacks relative signal in Countdown. The policy sometimes solves prompts, but many sampled groups contain no successful completion, producing zero-informative binary rewards. This means pass@K and training-signal quality must be reported together. A nonzero pass@8 does not guarantee useful group-relative advantages.

The feedback findings are more nuanced. Localized feedback is slightly better than shuffled feedback in the expanded repair run, but the margin is small. F4 targeted feedback performs worse than expected. One likely reason is that the message over-constrains the small model: it emphasizes the previous expression's value and target difference but does not supply a constructive search procedure. Another possibility is that the dominant WRONG_TARGET failure mode is too broad; many failed expressions evaluate cleanly but are far from the correct combinatorial expression. A local scalar delta may not identify which operation or subexpression should change.

The resampling control is therefore essential. If a second attempt without feedback outperforms feedback-conditioned repair, then the improvement cannot be attributed to feedback-specific credit assignment. The stronger conclusion is that successful trajectory selection is useful, but the current feedback templates do not yet prove that textual verifier feedback is the causal mechanism.

7 Limitations and Future Work

First, Countdown is controlled and verifier-friendly. Results may differ in code generation, theorem proving, or tool-use tasks where execution traces are richer. Second, some successful repair datasets are small, especially F4, so self-distillation results should be interpreted as preliminary. Third, the current F4 feedback template may be suboptimal. A stronger follow-up would decompose WRONG_TARGET errors into operation-level or subexpression-level alternatives rather than reporting only the final evaluated value and target.

Future experiments should run larger seed sweeps, test revised targeted feedback templates, and evaluate whether process-level verifier traces can outperform both generic revision and same-budget resampling. A larger model may also be necessary to use structured feedback reliably.

8 Conclusion

This project finds that the strongest evidence is a diagnostic result: in this Countdown RLVR setting, binary verifier rewards frequently produce zero-informative groups, so pass@K alone is not enough to characterize the usefulness of the training signal. The feedback experiments further show that textual repair gains must be interpreted against strong controls. Same-budget resampling outperforms the tested feedback templates, suggesting that successful trajectory selection transfers more clearly than the current targeted execution feedback. The final contribution is therefore a stricter diagnostic/control framework for evaluating verifier-feedback claims in group-relative RLVR.

9 Team Contributions

Yucheng Yao implemented the deterministic verifier, feedback taxonomy, Modal experiment pipeline, SFT warm start, diagnostic repair experiments, self-distillation runs, binary and shaped RLOO baselines, analysis scripts, poster figures, and report drafts.

10 AI Tools Disclosure

AI assistance was used for code scaffolding, debugging, experiment orchestration, analysis organization, figure generation, and writing assistance. The experiment design, verifier behavior, launched runs, reported metrics, and final conclusions were reviewed by the author.

References

[1] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv:2203.02155*, 2022.

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

[3] 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:2110.14168*, 2021.

[4] 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:2402.03300*, 2024.

[5] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 2022.

[6] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. *arXiv:2303.17651*, 2023.

[7] An Yang et al. Qwen2.5 technical report. *arXiv:2412.15115*, 2024.

[8] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv:2106.09685*, 2021.