

Environment-Grounded Self-Reward for Dense Feedback in RLVR

Research Proposal — NeurIPS 2026

Rohan Sikand · Stanford University

March 2026

Problem: Sparse Rewards in RLVR

RLVR (GRPO) trains LLMs on **binary pass/fail**.

- A correct algorithm that fails one edge case → same reward as total nonsense
- All-pass or all-fail groups → **zero gradient signal** despite compute spent

The environment produces **rich feedback** (stack traces, errors, partial results) — all **discarded** after computing pass/fail.

The ingredients for dense reward already exist inside the training loop.

Prior Approaches and Their Problems

Approach	Signal	Problem
Process reward models	Dense, step-level	Expensive labels, static
External LLM judge	Dense, flexible	Cost + distributional gap
Self-rewarding LMs	Dense, on-policy	Reward collapse (no grounding)
EGSR (ours)	Dense, on-policy	<i>Grounded in env feedback</i>

Key Observation

At every GRPO step, the environment already produces:

1. The student's **solution** a_i and **reasoning trace** r_i
2. **Execution feedback** f_i — stack traces, compiler errors, partial test outputs
3. **Binary reward** $R_{\text{env}} \in \{0, 1\}$

Items 1 and 2 are thrown away after computing item 3.

*A model that reads its own code alongside `IndexError` at line 4 understands **why** it failed — and can assign partial credit.*

EGSR: The Policy as Its Own Reward Model

Same model, two roles, differing only in context:

$$\underbrace{\pi_{\theta}(\cdot \mid s)}_{\text{Student: generates rollout}} \qquad \underbrace{\pi_{\theta}(\cdot \mid s, c_i, p_{\text{judge}})}_{\text{Teacher: judges rollout}}$$

where $c_i = (a_i, f_i, r_i)$ is the **privileged judgment context**.

Three properties:

- **Parameter-aligned**: same weights $\rightarrow$ zero distributional gap
- **Grounded**: teacher conditioned on environment feedback
- **Cheap**: one additional forward pass per rollout

Dense Self-Reward Extraction

Teacher outputs judgment over structured classes $\mathcal{K}$:

$$R_{\text{self}}(s, a_i) = \sum_{k \in \mathcal{K}} w_k \cdot P_{\theta}(q = k \mid s, c_i, p_{\text{judge}})$$

Class	Weight
correct	+1.0
minor_bug	+0.6
wrong_approach	+0.2
critical_error	0.0
exploit	-0.5

Continuous reward in $[-0.5, 1.0]$ from a single forward pass.

Why Dense Reward Matters for GRPO

GRPO advantages: $\hat{A}_i = (r_i - \mu)/\sigma$

Binary rewards create **two failure modes**:

1. **All pass/all fail** $\rightarrow \sigma = 0 \rightarrow$ no gradient signal
2. **No credit assignment** $\rightarrow$ 9/10 correct treated same as random

Dense self-reward addresses both:

- Different failure modes $\rightarrow$ different rewards $\rightarrow$ GRPO **always has signal**
- Partial credit: nearly-correct solutions get higher advantage

Why Environment Grounding Matters

1. Prevents reward collapse

- Without f_i : model can hallucinate success $\rightarrow$ sycophantic self-reward
- With f_i : `IndexError` at line 4 is an **inescapable constraint**

2. Recovers discarded information

$$I(Q^*; f \mid R_{\text{env}}, a) = H(Q^* \mid R_{\text{env}}, a) - H(Q^* \mid R_{\text{env}}, a, f) > 0$$

Binary reward leaves ambiguity about *why* a rollout failed. Feedback resolves it.

3. Parameter-aligned by construction

Same weights $\theta \rightarrow$ when θ updates, both student and teacher update.

The Judge Maintenance Problem

Risk: RL optimizes θ for generation, not judgment. Teacher forward pass under stop-gradient — nothing explicitly trains it.

Prior work uses a **labeled replay buffer** $\mathcal{B}_{\text{judge}}$. But:

- Introduces label dependency
- Goes **stale** as policy evolves (same distributional gap problem!)

Two signals are free at every step: $R_{\text{env}} \in \{0, 1\}$ and execution feedback f_i

Label-Free Judge Maintenance

Feedback prediction — predict f_i from (s, a_i) :

$$\mathcal{L}_{\text{pred}}(\theta) = -\mathbb{E} \left[\frac{1}{|f_i|} \sum_k \log \pi_{\theta}(f_{i,k} \mid s, a_i, f_{i,<k}) \right]$$

Forces internalization of code $\rightarrow$ execution causal relationship. Self-supervised, always on-distribution.

Verifier calibration — anchor R_{self} to pass/fail:

$$\mathcal{L}_{\text{cal}}(\theta) = \mathbb{E} \left[(R_{\text{self}}(s, a_i) - R_{\text{env}}(s, a_i))^2 \right]$$

Joint objective:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} [\mathcal{L}_{\text{GRPO}} + \lambda_1 \mathcal{L}_{\text{pred}} + \lambda_2 \mathcal{L}_{\text{cal}}]$$

Theoretical Motivation

1. Gradient variance reduction

Binary: $\text{Var}[\hat{A}] \propto p(1-p)/G \rightarrow$ vanishes at extreme p

Dense: $\text{Var}[\hat{A}] \geq \sigma_R^2/G > 0$ regardless of p

2. Information-theoretic value of feedback

$I(Q^*; f \mid R_{\text{env}}, a) > 0$ — upper bounds improvement from leveraging f_i

3. Calibration preserves optimal policy

If $\mathcal{L}_{\text{cal}} = 0$, optimal policies under $R_{\text{self}} \subseteq$ optimal policies under R_{env}

$\rightarrow$ Dense reward doesn't introduce false optima

Implementation Plan

Framework: verl + FSDP2 + vLLM

- Extend in two places: (1) teacher forward pass, (2) $\mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{cal}}$ in backward

Models:

- Qwen3-4B-Instruct — primary (matches existing baselines)
- Qwen3-14B-Instruct — scale experiment
- OLMo3-7B-Instruct — secondary model family

Domains: Code generation (LeetCode, LiveCodeBench) + Math (MATH, AIME)

Experimental Plan: Baselines

Method	Reward	Adapts?	Labels?
GRPO (binary)	$R_{\text{env}} \in \{0, 1\}$	N/A	No
Test-case fraction	$R = k/n$	N/A	No
External judge (Haiku)	Dense, frozen	Static	No
Ungrounded self-reward	$\pi_{\theta}(\cdot s, a_i)$	On-policy	No
EGSR + $\mathcal{L}_{\text{CE}}$	R_{self}	On-policy	Yes
EGSR (label-free)	R_{self}	On-policy	No

E1: Signal Validation (Go/No-Go)

Frozen Qwen3-4B, 5K rollouts, no RL training.

Measure: AUC of R_{self} vs ground-truth correctness under context ablations:

Condition	Context	Expected
(a)	Solution only (ungrounded)	Low
(b)	+ execution feedback f_i	High
(c)	+ reasoning trace r_i	Moderate
(d)	Full ($a + f + r$)	Highest

Gate: If (b) $\approx$ (a) at 4B $\rightarrow$ pivot to 14B or stop.

E2: Dense Reward Effectiveness (Core Result)

Full GRPO training, 200–400 steps.

Hypotheses:

1. EGSR $>$ binary GRPO on Pass@1 and convergence speed
2. EGSR $\approx$ external judge at fraction of cost
3. EGSR $>$ ungrounded self-reward $\rightarrow$ grounding is key
4. EGSR $>$ test-case fraction within same pass-fraction bucket

Additional:

- Stratify by problem difficulty
- Gradient norm analysis (validate theory)
- Group size ablation: EGSR $G=4$ vs binary GRPO $G=16$?

E3: Judge Maintenance (Key Novel Experiment)

Track teacher AUC every 20 steps throughout RL.

Condition	Maintenance	Labels?
(a) None	$\lambda_1 = \lambda_2 = 0$	No
(b) $\mathcal{L}_{\text{pred}}$ only	Feedback prediction	No
(c) $\mathcal{L}_{\text{cal}}$ only	Verifier calibration	No
(d) $\mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{cal}}$	Full label-free	No
(e) $\mathcal{L}_{\text{CE}}$	Static labeled buffer	Yes

Hypothesis: (a) degrades; (e) degrades slowly (stale buffer); (d) stays flat.

E4–E6: Supporting Experiments

E4 — Reward hacking dynamics

- Track hack rate over training (external labeler)
- Does exploit class ($w = -0.5$) suppress hacking?

E5 — Scale and domain transfer

- 4B vs 14B self-judgment quality
- Math domain (MATH/AIME) — different feedback modality

E6 — Mechanistic analysis

- What does the teacher attend to in f_i ?
- Empirically estimate $I(Q^*; f \mid R_{\text{env}}, a)$

Key Ablations

Ablation	Experiment	Values
Judgment granularity	E1, E2	Binary vs 5-class
λ_1 (feedback pred.)	E3	$\{0.01, 0.1, 0.5, 1.0\}$
λ_2 (calibration)	E3	$\{0.01, 0.1, 0.5, 1.0\}$
Warm-start SFT	E1, E3	With vs without
Group size G	E2	$\{4, 8, 16\}$
w_{exploit}	E4	$\{-1.0, -0.5, -0.25, 0\}$

Risks and Mitigations

Risk	Mitigation
4B judgment capacity	Warm-start + $\mathcal{L}_{\text{pred}}$; 14B fallback
$\mathcal{L}_{\text{pred}}$ doesn't $\rightarrow$ judgment	$\mathcal{L}_{\text{cal}}$ bridges; ablate each
Reward collapse	$\mathcal{L}_{\text{pred}}$ forces f_i engagement
Self-judge hacking	$\mathcal{L}_{\text{cal}}$ + $\mathcal{L}_{\text{pred}}$ + consistency checks
Fraction baseline matches	Within-bucket analysis; AIME

Timeline

Phase	Weeks	Milestone
Infrastructure (E0)	1	verl pipeline confirmed
Signal validation (E1)	2–3	Go/no-go decision
Core experiments (E2+E3)	4–8	Main results
Supporting (E4+E5)	9–11	Scale + hacking
Mechanistic + writing (E6)	11–13	Paper complete

Target: **NeurIPS 2026**

Summary

Method:

- Policy as its own reward model
- Dense reward from env-grounded self-judgment
- Label-free maintenance via $\mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{cal}}$

Properties:

- On-policy (parameter-aligned)
- Grounded (no reward collapse)
- Cheap (one forward pass)
- Label-free (no replay buffer)

Implementation:

- verl + vLLM
- Qwen3-4B/14B, OLMo3-7B
- Code + math domains

Theory:

- Gradient variance reduction
- Information-theoretic bound
- Calibration $\rightarrow$ no false optima

1. **Framing:** Lead with *dense reward shaping for RLVR* (clean problem) — self-distillation connection in discussion section
2. **Go/no-go:** E1 gates everything — 3 weeks to know if thesis holds
3. **Key novel experiment:** E3 (judge maintenance) — standalone result if label-free $\geq$ labeled
4. **Biggest empirical risk:** Test-case fraction baseline — need within-bucket wins
5. **Theoretical contributions:** Variance reduction (most tractable) + MI bound (motivational)