

Adversarial Co-Training for Online Reward Hacking Mitigation (ARHD)

2026-02-26

Problem

Let π_θ be a policy (LLM) trained via reinforcement learning on coding tasks. Given a prompt s , the policy generates a solution $a \sim \pi_\theta(\cdot | s)$, and a reward function $R(s, a)$ assigns reward based on test-case correctness. Reward hacking occurs when the model exploits vulnerabilities in the evaluation environment — e.g., redefining `run_tests()` — to receive high reward without solving the task.

The standard approach is to train a static monitor $M_\phi : (s, a) \rightarrow [0, 1]$ that predicts the probability a response contains reward hacking. Once trained, M_ϕ is frozen. As π_θ evolves during RL, the distribution of (s, a) pairs shifts, and the model learns to produce outputs that evade M_ϕ . Empirically, probe accuracy degrades by up to 16 percentage points as models learn evasion strategies such as extending response length and adding unfaithful code comments (Ariaw et al., 2025). This creates an **evasion gap** between the monitor’s fixed decision boundary and the generator’s evolving output distribution.

The core requirement is clear: the detection mechanism must **adapt online** alongside the policy to avoid being exploited.

Approach: Adversarial Co-Training

We frame detection as a **two-player game** between a generator π_θ and a critic C_ψ that co-evolve during training:

$$\text{Critic: } \max_{\psi} \mathbb{E}_{s, a \sim \pi_\theta} [y^* \log C_\psi(s, a) + (1 - y^*) \log(1 - C_\psi(s, a))]$$

$$\text{Generator: } \max_{\theta} \mathbb{E}_{s,a \sim \pi_{\theta}} [R_{\text{task}}(s, a) - \lambda \cdot C_{\psi}(s, a)]$$

where $C_{\psi}(s, a) \in [0, 1]$ is the critic’s predicted hacking probability and y^* is the hack label for rollout (s, a) . The critic trains to identify hacking in the generator’s current rollouts; the generator maximizes task reward while minimizing the critic’s predicted hacking probability. Because C_{ψ} is updated at each training step on the current policy’s outputs, it tracks distributional shift as the generator evolves — closing the evasion gap by design.

The Label Problem

The adversarial formulation above requires **hack labels y^* at every training step**. This creates a fundamental challenge: if we had a perfect oracle for y^* , we could simply penalize hacking directly and would not need the critic at all. Prior supervised approaches face a **circular dependency** — requiring ground-truth hack labels to train a detector defeats the purpose of automated detection. The central question becomes: *how do we obtain y^* online without an oracle?*

We propose and compare multiple strategies for resolving this.

Method 1: LLM Judge as Online Pseudolabeler

An LLM judge (e.g., Claude Haiku) is prompted at each training step to classify whether each rollout (s, a) contains reward hacking, producing pseudolabels $\hat{y}$:

$$\hat{y}_i = \text{LLMJudge}(s, a_i) \in \{0, 1\}$$

These pseudolabels replace y^* in the critic objective. The critic and policy co-evolve online: the judge provides noisy but adaptive supervision, and the critic learns to smooth and generalize from these signals. This avoids the circular dependency because the judge operates on the *content* of the rollout (identifying suspicious patterns like test redefinition) rather than requiring execution-level ground truth. The tradeoff is that LLM judges have limited accuracy — particularly on long responses and unfaithful code comments — and introduce inference cost at every training step.

Method 2: SDRH-EMA (Self-Distillation with Exponential Moving Average)

Rather than sourcing external labels, we eliminate the need for y^* entirely by reframing detection as a **self-distillation** problem. The policy π_θ is compared against an exponential moving average of its own past checkpoints:

$$\pi_{\text{EMA}} \leftarrow \rho \cdot \pi_{\text{EMA}} + (1 - \rho) \cdot \pi_\theta$$

For each rollout a_i given prompt s , we compute the per-token KL divergence:

$$d_i = D_{\text{KL}}(\pi_\theta(\cdot \mid s, a_i^{<t}) \parallel \pi_{\text{EMA}}(\cdot \mid s, a_i^{<t}))$$

averaged across token positions t . Honest capability improvements are gradual — the EMA tracks them closely, yielding low d_i . Hacking strategies emerge as sharp distributional shifts, producing anomalously high d_i on exploitative rollouts. The divergence d_i serves as an **unsupervised proxy** for y^* , replacing the critic entirely with selective advantage gating:

$$\hat{A}_i^{\text{gated}} = \hat{A}_i \cdot g(d_i)$$

where $g(d_i) = \exp(-\gamma \cdot d_i)$ (soft gating) or $g(d_i) = \mathbf{1}[d_i < \tau]$ (hard gating / screening). The full objective augments GRPO:

$$\mathcal{L}_{\text{SDRH}}(\theta) = -\mathbb{E}_{s, \{a_i\}} \left[\frac{1}{G} \sum_{i=1}^G \min(\rho_i \hat{A}_i^{\text{gated}}, \text{clip}(\rho_i, 1 \pm \epsilon) \hat{A}_i^{\text{gated}}) \right] + \lambda \cdot \mathcal{L}_{\text{SD}}(\theta)$$

where $\mathcal{L}_{\text{SD}}$ is an optional auxiliary self-distillation loss with divergence-proportional weighting $w(d_i) = d_i^\alpha$, applying stronger regularization on the most anomalous rollouts. SDRH-EMA is genuinely self-supervised: there is nothing specific to evade, and no external signal that creates a circular dependency.

Method 3: SDRH-Prompt (Self-Distillation with Inoculation Prompts)

This variant uses **inoculation prompts** to create teacher–student divergence from a single set of weights. The student generates under standard conditions, $y \sim \pi_\theta(\cdot \mid x)$, while the teacher is conditioned on a privileged honesty prompt c : $y' \sim \pi_\theta(\cdot \mid x, c)$. The per-token log-ratio

$$A_t = \log \pi_\theta(y_t \mid x, c, y_{<t}) - \log \pi_\theta(y_t \mid x, y_{<t})$$

serves as a hacking signal: when $A_t \ll 0$, the student generates tokens the honest teacher would strongly avoid. The training objective augments GRPO with a KL penalty:

$$\mathcal{L}(\theta) = \mathbb{E}_{x, y \sim \pi_\theta} [R_{\text{task}}(x, y)] - \lambda \mathbb{E}_{x, y \sim \pi_\theta} \left[\sum_{t=1}^T D_{\text{KL}}(\pi_\theta(\cdot \mid x, y_{<t}) \parallel \text{sg}(\pi_\theta(\cdot \mid x, c, y_{<t}))) \right]$$

where $\text{sg}(\cdot)$ denotes stop-gradient on the teacher. SDRH-Prompt inherits a weakness: if the privileged context is too strong, it reduces to the circular dependency; if too weak, it may be evadable. We include it as a secondary comparison.

Experimental Setup

We use the **Overwrite Tests** environment from Ariaw et al. (2025) with **Qwen3-4B** trained via GRPO on 992 LeetCode medium/hard problems (113 test). Reward hacking emerges naturally without explicit prompting. Evaluation proceeds in three phases: (1) **hypothesis validation** — do the proposed signals (LLM pseudolabels, EMA divergence, prompt divergence) correlate with ground-truth hacking labels? (2) **detection dynamics** — do adaptive methods maintain detection accuracy where static probes degrade? (3) **mitigation effectiveness** — do the proposed methods reduce hack rates while preserving genuine coding performance?