VERIFIER ERRORS IN RLVR: REWARD HACKING, LIMITS OF FEEDBACK, AND SELECTIVE CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

In reinforcement learning with verifiable rewards (RLVR), imperfect verifiers can reward incorrect responses, creating opportunities for reward hacking. Using gradient flow with a fixed verifier, we characterize the conditions under which reward rises while correctness falls. We then show that the observations available during RLVR are, in general, insufficient to detect or identify accepted errors, or to guarantee their reduction without sacrificing correct responses. To address this limit, we construct a correction using additional feedback about correctness from audits. This correction achieves *selective control*: at the current policy, it lowers the probability of accepted errors and raises that of correct responses, provided it outweighs the pressure toward errors from verifier reward. Experiments with log linear and neural contextual bandits and with a language model support the analysis and show that selective control under partial auditing reduces accepted errors while increasing correctness.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2025) fine-tunes pretrained language models using rewards from automated checks. These checks come from a verifier, which compares a model's response with a known solution or runs the code in the response against tests. Rewarding responses that pass the verifier has improved reasoning performance on mathematics and coding tasks (DeepSeek-AI et al., 2025). These gains, however, depend on the verifier: higher reward reflects progress on the intended task only when the verifier reliably judges correctness.

A verifier can accept a response even when the response fails the intended task. For example, the code in a response may pass the available tests yet fail on inputs those tests omit. Rewarding such responses reinforces accepted errors: responses that satisfy the verifier but fail the intended task. As the model produces accepted errors more often, average reward can rise even while performance on the intended task declines, the signature of reward hacking (Skalse et al., 2022).

Empirical studies document reward hacking in RLVR. Reward hacking appears, for example, in inductive reasoning, where a model must infer a general rule from labeled examples. Models trained with RLVR often skip this rule and instead list the label of each example. The verifier accepts these responses because it checks only whether a response labels the given examples correctly (Helff et al., 2026). Beyond eliciting such new hacking behaviors, reinforcement learning can amplify those that models acquired during earlier fine-tuning (Khalifa et al., 2026). These findings motivate methods that reduce reward hacking while preserving progress on the intended task.

Existing methods address reward hacking by constraining optimization (Laidlaw et al., 2025), improving feedback (Coste et al., 2024; Lightman et al., 2024), or detecting and correcting hacking (Baker et al., 2025; Wang et al., 2026a). Yet because these methods rely on assumptions about correctness (Everitt et al., 2017), they can leave verifier errors unresolved (Eisenstein et al., 2024). Training therefore continues with these verifier errors, rewarding correct responses and accepted errors alike. This raises a broader question: how do these verifier errors shape what RLVR learns about the intended task as training proceeds?

To answer this question, we hold an imperfect verifier fixed and study how the policy learns from its rewards. We model this learning as gradient flow because it makes the analysis tractable. Within this setting, we study when reward hacking grows, what the information available during RLVR

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.

training reveals about it, and when an intervention reduces hacking while preserving correctness. Our contributions are:

(i) We derive a condition under which the share of accepted errors among accepted responses grows at the current policy, and we express it through two mechanisms: hack bias and correctness-to-hack leakage (Proposition 3.2). We also characterize when the gradient flow increases verifier reward while reducing correctness, the signature of reward hacking (Proposition 3.1).

(ii) We establish limits on detection, identification, and selective control from verifier feedback alone. We show that the complete RLVR training record provides no advantage in detecting accepted errors and can leave correctness unidentifiable (Propositions 4.1 and 4.2). We further prove that no controller using this record alone can guarantee fewer accepted errors while preserving correct responses (Proposition 4.3).

(iii) We derive conditions under which a correction to RLVR training reduces reward hacking while improving correctness, and we design projected audit correction, which satisfies them using correctness labels from audits of some responses (Theorem 5.1).

We test our theory in settings of increasing complexity. In contextual bandits, reward hacking emerges and projected audit correction reduces hacks while increasing correctness. In a language model, verifier reward rises while correctness falls, and the same correction reverses this trend.

2 Problem Formulation

2.1 Reinforcement Learning with Verifier Rewards (RLVR)

Policy. Given a prompt x drawn from a prompt distribution $\mathcal{D}$, the language policy $\pi_\theta(\cdot \mid x)$ samples a response y from the response set $\mathcal{Y}_x$. This policy has a trainable parameter vector $\theta \in \mathbb{R}^d$.

RLVR. We analyze RLVR (Lambert et al., 2025), where the verifier assigns a reward $R(x, y) \in \{0, 1\}$. A reward of 1 indicates acceptance. RLVR training maximizes the scalar objective: $J_R(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}[R(x, y)]$. This objective represents the probability of acceptance averaged over prompts and sampled responses, yielding a value in $[0, 1]$.

2.2 Correctness and Accepted Errors

We define a fixed correctness indicator $c(x, y) \in \{0, 1\}$ for each pair (x, y), where $c = 1$ denotes a correct response. To focus our analysis on the consequences of accepting incorrect responses, we assume the verifier produces no false negatives. This assumption implies that $c(x, y) \leq R(x, y)$ for all pairs, meaning the verifier accepts all correct responses. We call the event of an incorrect response being accepted, i.e., $\{R(x, y) = 1, c(x, y) = 0\}$, a *hack*.

For any given prompt x, we can partition the set of all accepted responses, $A_x = \{y \in \mathcal{Y}_x : R(x, y) = 1\}$. This set consists of two disjoint subsets. The first is the set of correct responses, $G_x = \{y \in \mathcal{Y}_x : c(x, y) = 1\}$. The second is the set of hacks, $H_x = A_x \setminus G_x$. These sets remain fixed throughout training. The policy, however, can learn to sample from them with different frequencies.

Acceptance and hacking. For any given prompt x, we define two key probabilities. The first is the acceptance probability, $p_x(\theta) = \Pr_{\pi_\theta}(Y \in A_x \mid x)$. The second probability, defined only when $p_x(\theta) > 0$, is the hacked share, $q_x(\theta) = \Pr_{\pi_\theta}(Y \in H_x \mid Y \in A_x, x)$. Both probabilities depend on the policy parameters θ and change throughout RLVR training. Using the acceptance probability $p(\theta)$, we can now rewrite the RLVR objective as the following expectation over prompts: $J_R(\theta) = \mathbb{E}_{x\sim\mathcal{D}}[p_x(\theta)]$. Because this objective depends only on $p_x(\theta)$, it makes no distinction between correct responses and hacks (see Appendix C.1.2).

2.3 Gradient Flow Dynamics

To analyze how the policy evolves, we use gradient ascent in continuous time. This model isolates the effects of the verifier signal by removing the noise inherent in stochastic optimization. When the objective J_R is differentiable, the parameters evolve according to the gradient flow:

$$\dot{\theta}(t) = g_R(\theta(t)) := \nabla_\theta J_R(\theta(t)), \tag{1}$$

where $g_R(\theta) \in \mathbb{R}^d$ is the exact gradient. A direct consequence is that the objective can only improve, as $\dot{J}_R(\theta(t)) = \|g_R(\theta(t))\|^2 \geq 0$ (see Appendix C.1.3). This guarantee, however, applies only to the overall acceptance probability. It reveals nothing about the hacked share, $q_x(\theta)$ which may increase, decrease, or remain unchanged.

Research questions. We now define our two central research questions within our gradient flow framework. First, under what conditions does the hacked share, $q_x(\theta)$, grow or persist during training? Answering this question requires distinguishing increases in acceptance due to correct responses from shifts in the policy's sampling toward hacks. Second, since the verifier's feedback is blind to correctness, is it possible to reduce the hacked share using verifier's information alone? If not, what additional information would allow it to do so?

3 When Hacking Grows: A Population Analysis

This section analyzes the population dynamics of hacked responses. We show that the accepted population and the hacked share can increase together, and characterize when verifier flow increases reward while reducing correctness. We then describe how hack bias and correctness-to-hack leakage drive hacking growth. Ultimately, these drivers create a latent vulnerability that on-policy training can reinforce.

3.1 Verifier Acceptance and Hacked Share Can Increase Together

We partition responses into three populations: correct responses $G = \{(x, y) : y \in G_x\}$, hacks $H = \{(x, y) : y \in H_x\}$, and rejected responses $N = \{(x, y) : y \in \mathcal{Y}_x,\ R(x, y) = 0\}$. Let $p_G(\theta) := \Pr_\theta(G) \in [0, 1]$ and $p_H(\theta) := \Pr_\theta(H) \in [0, 1]$ denote the probabilities of correct responses and hacks, respectively. To track both how often the verifier accepts and how often accepted responses are hacks, we define $p(\theta) := \Pr_\theta(G \cup H) = J_R(\theta)$ and $q(\theta) := \Pr_\theta(H \mid G \cup H)$, where $p(\theta) \in [0, 1]$ measures overall acceptance and $q(\theta) \in [0, 1]$ measures the hacked share among accepted pairs. The latter requires $p(\theta) > 0$.

Expected reward depends on total acceptance $p = p_G + p_H$, regardless of how it splits between correct responses and hacks. Writing $p_H(\theta) = p(\theta)q(\theta)$ and $p_G(\theta) = p(\theta)(1 - q(\theta))$ and differentiating with respect to time gives

$$\dot{p}_H(\theta) = q(\theta)\dot{p}(\theta) + p(\theta)\dot{q}(\theta), \qquad \dot{p}_G(\theta) = (1 - q(\theta))\dot{p}(\theta) - p(\theta)\dot{q}(\theta),$$

where the $\dot{p}$ terms reflect changes in total acceptance and the $\dot{q}$ terms reflect shifts between correct responses and hacks. These identities impose no trade-off between acceptance and hacked share; both can increase together.

Reward hacking. Skalse et al. (2022) call a proxy reward hackable if some pair of policies has higher expected proxy return but lower expected true return. In our setting, the verifier reward $J_R = p$ serves as the proxy, and correctness $J_C = p_G$ serves as the true return. We characterize when gradient flow on J_R, which we call verifier flow, generates such a pair.

Proposition 3.1 (Reward hacking along the flow)**.** *Suppose p_G and p_H are continuously differentiable and $p > 0$ along verifier flow on $[0, T]$. Then there exist $t_1 < t_2$ with $J_R(\theta(t_2)) > J_R(\theta(t_1))$ and $J_C(\theta(t_2)) < J_C(\theta(t_1))$ if and only if $p(\theta(t))\dot{q}(\theta(t)) > (1 - q(\theta(t)))\dot{p}(\theta(t))$ at some $t \in (0, T)$.*

Proof. See Appendix C.2.2. □

Interpretation. The two sides of the inequality compete. The term $p\dot{q}$ is the correctness lost as the accepted population shifts toward hacked responses, and $(1 - q)\dot{p}$ is the correctness gained from rising acceptance. Training produces reward hacking once the loss outweighs the gain. The condition needs to hold at only a single time to produce a pair of policies that witnesses hackability. Because both terms depend on the policy, the condition can hold at one time and fail at another.

3.2 The Dynamics of the Hacked Share

While the hacked share q remains the quantity of interest, log-odds coordinates simplify its dynamics. When both $p_H(\theta) > 0$ and $p_G(\theta) > 0$, we can define the log odds $z(\theta) \in \mathbb{R}$ as

$$z(\theta) := \log \frac{p_H(\theta)}{p_G(\theta)} = \log \frac{q(\theta)}{1 - q(\theta)},$$

Because z is strictly increasing in q, the sign of $\dot{z}$ determines whether the hacked share grows, shrinks, or remains constant.

The log odds dynamics. Since $z = \log p_H - \log p_G$, the rate $\dot{z}$ depends on the gradient of each group's log-probability. For each group $S \in \{G, H, N\}$ with positive, continuously differentiable probability, we denote this gradient by $\bar{s}_S(\theta) := \nabla_\theta \log \Pr_\theta(S) \in \mathbb{R}^d$, the group's average score. Along the verifier flow (1), differentiating z yields

$$\dot{z}(\theta) = \big(\bar{s}_H(\theta) - \bar{s}_G(\theta)\big)^\top g_R. \tag{2}$$

The log odds grow when the reward gradient g_R aligns with the score difference between hacks and correct responses. Differentiating total acceptance $p = p_G + p_H = 1 - p_N$ shows that g_R is a weighted combination of the three group scores: $g_R = p\big[(1-q)\bar{s}_G + q\bar{s}_H\big] = -(1-p)\bar{s}_N$. This decomposition, combined with (2), resolves $\dot{z}$ into two mechanisms:

$$\dot{z} = p(1-p)[\underbrace{(\bar{s}_H - \bar{s}_G)^\top(\bar{s}_G - \bar{s}_N)}_{\text{correctness-to-hack leakage}} + \underbrace{q\|\bar{s}_H - \bar{s}_G\|^2}_{\text{hack bias}}]. \tag{3}$$

The full derivation is provided in Appendix C.2.3.

3.3 Main Result: The Mechanism of Hacking Growth

We now state our main result characterizing the mechanism of hacking growth.

Proposition 3.2 (Population growth of the hacked share)**.** *Along verifier gradient flow, with $p_G, p_H, p_N > 0$, the population hacked share grows if and only if*

$$(\bar{s}_H - \bar{s}_G)^\top(\bar{s}_G - \bar{s}_N) + q\|\bar{s}_H - \bar{s}_G\|^2 > 0.$$

When the inequality holds, $\dot{z} > 0$. Since $\dot{q} = q(1-q)\,\dot{z}$ and $\dot{p} = \|g_R\|^2 \geq 0$, both the hacked share and total acceptance grow strictly, while the correct share among accepted responses decreases.

The two mechanisms of (3) play different roles:

(i) Hack bias, $q\|\bar{s}_H - \bar{s}_G\|^2$, arises because hacks contribute to the reward gradient in proportion to their share among accepted responses. It is always nonnegative and grows with the hacked share at fixed group scores.

(ii) Correctness-to-hack leakage, $(\bar{s}_H - \bar{s}_G)^\top(\bar{s}_G - \bar{s}_N)$, arises when the direction that favors correct responses over rejected ones also favors hacks over correct responses. It can be positive or negative: positive leakage reinforces hack bias, negative leakage opposes it and can reverse growth when its magnitude exceeds the bias.

Interpretation. This result is local: the group scores and hacked share depend on the current parameters, so the growth condition can change during training. The group scores evolve, so continued growth is not guaranteed. Yet a feedback loop is present: a growing hacked share strengthens hack bias at fixed group scores, which favors further growth. This result exposes a latent vulnerability: hacking can reinforce the conditions that favor its own growth, because the policy generates its own training samples. Moreover, RLVR training is not designed to oppose this reinforcement. We must therefore build any hacking defense on top of RLVR. Can the information generated during RLVR training support such a defense?

4 What RLVR Training Reveals: Limits of Verifier Feedback

Section 3 showed that rising rewards can mask a growing share of hacks. We now study whether the observations collected during RLVR training can expose them. We show that they cannot: training observations confer no detection advantage, leaving correct responses unidentifiable, and making selective control impossible.

4.1 RLVR Training Observations, Monitors, and Compatible Correctness

RLVR training observations. RLVR training produces prompts, sampled responses, and verifier labels. Let L_t collect these observations through time t, along with policy parameters, probabilities,

gradients, and other derived quantities. This record is deliberately generous: the limitations below do not arise from incomplete logging. We assume that L_t contains no additional correctness feedback beyond the verifier. We write $\mathcal{F}_t^R := \sigma(L_t)$ for the information contained in the record.

Monitors. A monitor is an algorithm that uses L_t to assess hacking. A detection monitor raises a binary alarm about whether hacks exist; an identification monitor predicts the correctness label $\widehat{c}_t(x, y)$ of an accepted response. We consider deterministic monitors here and defer randomized extensions to Appendix C.3.

Compatible correctness assignments. A fixed assignment c determines correctness. To study what the training record reveals about c, we consider alternative assignments consistent with the verifier. Assuming no false negatives, we define

$$\mathcal{C}_R := \{c' : c'(x, y) \in \{0, 1\}, c'(x, y) \leq R(x, y) \text{ for all } (x, y)\}.$$

This class models uncertainty about the true correctness assignment; c itself remains fixed during training. All members agree on rejected responses, but they may disagree on accepted ones. To isolate the effect of this uncertainty, we compare these alternative assignments while holding the prompt distribution, verifier, initialization, and training algorithm fixed.

4.2 Limits of Detecting Hacking

Verifier acceptance can increase while the hacked share grows. Can the training record reveal even the presence of hacks? Under $c = R$, every accepted response is correct. A compatible alternative $c_1 \in \mathcal{C}_R$ with $\Pr_\theta(R = 1, c_1 = 0) > 0$ admits hacks. *Detection* requires distinguishing these two cases using only L_t.

Proposition 4.1 (Limits of detection)**.** *Under the comparison setup above, fix θ, t, and $c_1 \in \mathcal{C}_R$ with $\Pr_\theta(R = 1, c_1 = 0) > 0$. The training record has the same distribution under both assignments: $\mathrm{Law}_{c=R}(L_t) = \mathrm{Law}_{c=c_1}(L_t)$. Thus, no monitor can distinguish $c = c_1$, which admits hacks, from $c = R$, which has none.*

Proof. See Appendix C.3.2. □

Interpretation. Because alternative correctness assignments do not affect verifier feedback, they leave the training record's distribution unchanged. Thus, any monitor's detection rate under c_1 equals its false-alarm rate under $c = R$. Even knowing that hacks exist does not reveal which accepted responses are wrong.

4.3 Limits of Identifying Hacks

Suppose we know that hacks exist. The remaining task is to determine which accepted responses are wrong. *Identification* requires a monitor to recover the true label $c(x, y)$ of an accepted response from L_t. To succeed, the prediction $\widehat{c}_t(x, y)$ must distinguish compatible assignments that disagree on this response, even when we restrict $\mathcal{C}_R$ to assignments that admit hacks.

Proposition 4.2 (Limits of identification)**.** *Under the comparison setup above, fix θ and t. Suppose θ assigns positive probability to both correct responses and hacks under some assignment in $\mathcal{C}_R$. For every monitor and every accepted pair (x, y), there is an assignment $c_1 \in \mathcal{C}_R$ with $\Pr_\theta(R = 1, c_1 = 0) > 0$ such that $\Pr_{c_1}(\widehat{c}_t(x, y) \neq c_1(x, y)) \geq \frac{1}{2}$. Thus, no monitor can guarantee the correct label of an accepted response, even when we know hacks exist.*

Proof. See Appendix C.3.3. □

Interpretation. Two compatible assignments can label the same accepted response differently while producing identical records. Both can admit hacks, so knowing that hacks exist does not resolve the disagreement. A prediction correct under one assignment is wrong under the other. Thus, the monitor incurs an error probability of at least $\frac{1}{2}$ under at least one assignment. Because reducing hacks need not require identifying every hack, this result alone does not rule out selective control.

4.4 Limits on Selective Control from Verifier Feedback Alone

Even if we cannot identify every hack, can we reduce hack probability without reducing the probability of correct responses? Suppose a controller uses L_t to apply a correction $u(t) \in \mathbb{R}^d$ to the verifier flow, yielding $\dot{\theta} = g_R + u(t)$. Along this corrected flow, *selective control* requires $\dot{p}_H < 0$ and $\dot{p}_G \geq 0$ whenever $p_H > 0$. We target p_H itself, not the hacked share q, because q can fall even while p_H grows. For a uniform guarantee, this same controller must satisfy these conditions under every compatible assignment in $\mathcal{C}_R$.

Proposition 4.3 (Limits of selective control)**.** *Under the comparison setup above, suppose the initial policy assigns positive probability to both correct responses and hacks under some assignment in $\mathcal{C}_R$. For corrected flows with differentiable group probabilities, no controller using only L_t can guarantee $\dot{p}_H < 0$ and $\dot{p}_G \geq 0$ whenever $p_H > 0$, under every assignment $c' \in \mathcal{C}_R$. Here, each assignment c' defines its own p_H and p_G.*

Proof. See Appendix C.3.4. □

Interpretation. We seek a correction $u(t)$ such that the corrected flow reduces hack probability without reducing the probability of correct responses. However, compatible assignments can exchange the roles of correct responses and hacks. An update that reduces hack probability under one therefore reduces the probability of correct responses under the other. The controller cannot distinguish these assignments from L_t, and more observations from the same verifier cannot resolve this conflict. Regularization illustrates this limit: it constrains policy updates using only the policy and the verifier, so it cannot guarantee selective control either (Appendix C.3.5). A uniform guarantee of selective control thus requires additional correctness information that rules out assignments requiring incompatible updates.

5 From Additional Feedback to Selective Control

Section 4 showed why verifier feedback alone cannot reduce hacks without also sacrificing correct responses. We now investigate whether additional information about correctness can break this trade-off. We show that it can: such information supports a correction that suppresses hacks and promotes correct responses, provided the correction is strong enough to overcome the drift toward errors induced by RLVR training.

5.1 Projected Audit Correction in RLVR Training

Audits as additional information. Verifier feedback alone cannot guarantee selective control (Section 4), so we introduce a stronger signal: audits. An *audit* reveals the true correctness label $c(x, y)$ for a prompt x and an accepted response y. The audit $Z = (x, y, c(x, y))$ rules out every correctness rule that disagrees with this label, restricting $\mathcal{C}_R$ to $\mathcal{C}_{R,Z} = \{c' \in \mathcal{C}_R : c'(x, y) = c(x, y)\}$. The limits in Section 4 arose because verifier feedback could not distinguish the true rule from alternatives in $\mathcal{C}_R$. Audits eliminate alternatives that contradict the revealed labels. To show how audits reduce hacking, we next define the audited hack population and its probability gradient under the policy.

Let A_{aud} be a fixed set of audits, and let $H_A = A_{\text{aud}} \cap H$ denote the audited hacks, with probability $p_{H_A}(\theta) := \Pr_\theta(H_A)$. When p_{H_A} is differentiable, the negative gradient $-\nabla_\theta p_{H_A}$ targets only the audited hacks and need not align with $-\nabla_\theta p_H$ (see Appendix A.2). Despite this misalignment, $-\nabla_\theta p_{H_A}$ locally decreases the audited hack probability when $\nabla_\theta p_{H_A} \neq 0$, so we use the correction $u = -\lambda \nabla_\theta p_{H_A}$, with $\lambda > 0$, in the verifier flow $\dot{\theta} = g_R + u$.

The correction opposes growth in p_{H_A}: $\dot{p}_{H_A} = \nabla p_{H_A}^\top g_R - \lambda \|\nabla p_{H_A}\|^2$. Because these probabilities share parameters, the correction also perturbs p and p_G:

$$\dot{p} = \|g_R\|^2 - \lambda\, g_R^\top \nabla p_{H_A}, \qquad \dot{p}_G = \nabla p_G^\top g_R - \lambda\, \nabla p_G^\top \nabla p_{H_A}.$$

The cross terms have no fixed sign. When $\nabla p_G^\top \nabla p_{H_A} > 0$, the correction can suppress correct responses alongside hacks. We next constrain u to preserve the growth rate of p.

Projected audit correction. Since $p = p_H + p_G$, we have $\nabla_\theta p_G = g_R - \nabla_\theta p_H$. Preserving the instantaneous growth rate of p requires $g_R^\top u = 0$, so u must lie in the subspace orthogonal to g_R.

Under this constraint, $\nabla_\theta p_G^\top u = -\nabla_\theta p_H^\top u$: whenever u opposes the growth of p_H ($\nabla_\theta p_H^\top u < 0$), it contributes equally to the growth of p_G.

We project the audit correction onto the subspace orthogonal to g_R using $P_\perp := I - g_R g_R^\top / \|g_R\|^2$ when $g_R \neq 0$ and $P_\perp := I$ otherwise, to define the *projected audit correction* $u = -\lambda P_\perp \nabla p_{H_A}$, with $\lambda > 0$, yielding the flow $\dot\theta = g_R + u$. Since $g_R^\top P_\perp = 0$ and $P_\perp$ is an orthogonal projector,

$$\dot p = \|g_R\|^2, \qquad \dot p_{H_A} = \nabla p_{H_A}^\top g_R - \lambda \|P_\perp \nabla p_{H_A}\|^2.$$

This projection preserves the instantaneous growth rate of p and opposes the growth of p_{H_A} whenever $P_\perp \nabla p_{H_A} \neq 0$. We next show when this projected correction shrinks the share of hacks in the full population and bounds that share during training.

5.2 Main Result: Achieving Selective Control

Assumptions. We analyze RLVR training under the projected audit correction on a finite interval $[0, T]$, under the following assumptions. *(A1) Regularity:* The probabilities p_G, p_H, p_{H_A} are continuously differentiable in θ, with $p_G, p_H > 0$ on $[0, T]$. *(A2) Audit coverage:* The fixed audit set satisfies $\Pr_{\theta(t)}(H \setminus A_{\text{aud}}) = 0$ throughout the interval, so $p_{H_A} = p_H$ and their gradients coincide during training. *(A3) Uniform corrective strength:* $\|P_\perp \nabla p_H\|^2 / p \geq \kappa q$ throughout the interval for some constant $\kappa > 0$. We discuss these conditions in Appendix A.2.

Theorem 5.1 (Selective control)**.** *Consider RLVR training under the projected audit correction with constant $\lambda > 0$ on $[0, T]$.*

(i) Selective control. Under (A1)–(A2), the correction opposes the relative growth of hacks while preserving the instantaneous growth rate of p:

$$\dot z = \underbrace{(\bar s_H - \bar s_G)^\top g_R}_{\textit{bias + leakage}} - \underbrace{\frac{\lambda \|P_\perp \nabla p_H\|^2}{p\, q(1-q)}}_{\textit{correction}}, \qquad \dot p = \|g_R\|^2.$$

Whenever the correction term exceeds the bias and leakage, the share of hacks decreases and p_G increases. More strongly, if $\lambda \|P_\perp \nabla p_H\|^2 > \nabla p_H^\top g_R$, then p_H itself decreases ($\dot p_H < 0$, $\dot p_G > 0$, and $\dot z < 0$), achieving selective control.

(ii) An ISS-type bound on the share of hacks. Assume additionally (A3), and let $D \geq 0$ bound the growth pressure, $(\bar s_H - \bar s_G)^\top g_R \leq D$, during training. Then, for every $t \in [0, T]$,

$$q(t) \leq e^{-\lambda\kappa t}\, q(0) + \frac{D}{4\lambda\kappa}\big(1 - e^{-\lambda\kappa t}\big).$$

The bound separates a decaying contribution from the initial hacked share and a residual contribution from growth pressure. For fixed D, a larger product $\lambda\kappa$ reduces the residual bound.

Proof sketch. By (A2), $\nabla p_{H_A} = \nabla p_H$. Substitute the correction into $\dot z = \nabla z^\top (g_R + u)$ and use $g_R^\top u = 0$ to obtain part (i). For part (ii), (A3) and $q(1-q) \leq 1/4$ give $\dot q \leq D/4 - \lambda\kappa q$; integration yields the bound. See Appendix C.4.2 for details. □

Interpretation. Under the stated assumptions, the projected audit correction favors correct responses over hacks while preserving the instantaneous growth rate of p. When bias and leakage supply no positive growth pressure, the input-to-state stability (ISS)-type bound (Khalil & Grizzle, 2002) ensures the share of hacks decays exponentially. Persistent pressure contributes a residual bound that decreases with $\lambda\kappa$ for fixed D. The selective effect is instantaneous; the trajectory bound requires the assumptions to hold throughout the interval.

Remark (Practical implementation). The control guarantees (Theorem 5.1) require audits that provide a direction opposing hack growth and sufficient corrective strength throughout training. Partial audit coverage can suffice when the resulting correction remains aligned with the gradient of hack probability. Finite-sample gradient estimates, optimizers, and finite step sizes can prevent the implemented update from preserving acceptance progress or suppressing hacks. Appendix A.2 discusses these requirements and how to estimate and validate the correction.

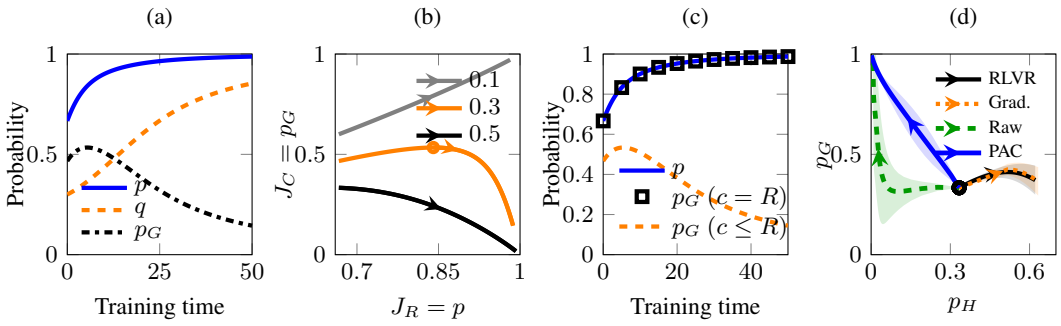

Figure 1: Reward hacking and selective control in contextual bandits. Panels (a)–(c) use a log linear policy, and panel (d) uses a neural policy. (a) Acceptance p, hacked share q, and correctness p_G under verifier flow with $q(0) = 0.3$. (b) Correctness ($J_C = p_G$) against verifier reward ($J_R = p$) for hacked shares $q(0) \in \{0.1, 0.3, 0.5\}$. (c) The same verifier flow evaluated under two correctness assignments: $c = R$, under which every accepted response is correct, and $c_1 = \mathbf{1}_G$. Squares mark correctness under $c = R$, which equals acceptance p. (d) Correctness p_G against the probability p_H of hacked responses under verifier flow, gradient regularization, raw audit correction, and projected audit correction. In (b) and (d), arrows point in the direction of training.

6 Experiments

We test reward hacking, the limits of verifier feedback, and selective control through audit correction in contextual bandits and language models.

6.1 Gaussian Contextual Bandits

Setup. We use Gaussian contextual bandits to study when training on verifier rewards amplifies reward hacking and whether audits enable selective control. We set the initial probabilities of correct and hacked responses, $p_G(0)$ and $p_H(0)$, to isolate how the initial composition shapes training. We also vary the policy class: a log linear policy keeps its features fixed, while a neural policy learns its own representation. Appendices D.1 and D.2 provide the experimental details.

Results. Figure 1 (a) shows acceptance p and hacked share q increasing together from $q(0) = 0.3$, as Proposition 3.2 predicts. Panel (b) plots correctness $J_C = p_G$ against verifier reward for three initial hacked shares. For $q(0) = 0.5$, correctness declines throughout the recorded interval; for $q(0) = 0.3$, it first improves and then declines; for $q(0) = 0.1$, both correctness and verifier reward improve. Both declines are reward hacking in the sense of Proposition 3.1. Panel (c) evaluates the same verifier flow under two correctness assignments: $c = R$, which admits no hacked responses, and $c_1 = \mathbf{1}_G$, which does. The information available during training is identical under both, yet correctness differs, as Proposition 4.1 predicts. Panel (d) shows selective control with a neural policy: projected audit correction (PAC) decreases the probability p_H of hacked responses while increasing correctness p_G, consistent with Theorem 5.1. Raw audit correction initially decreases both probabilities, whereas verifier flow and gradient regularization increase p_H. Appendices D.1 and D.2 provide additional experiments and ablations.

6.2 Language Models: Reward Hacking and Audit Correction

To test whether our predictions hold beyond exact gradient flow, we train a language model with sampled gradients and finite Adam updates.

Setup. We consider a task in which the model replaces a sequence of digits according to given rules. A response is correct only if every replacement in the final output is correct, but the imperfect verifier R checks only the last two digits. We include in each prompt a hint with an incorrect prefix and the correct final pair, so copying it produces a hack. We initialize Qwen2-0.5B with supervised fine-tuning (SFT) on correct responses G, hacks H, and rejected responses N, and then run GRPO for 20 rounds with or without projected audit correction (PAC). We vary audit coverage by letting

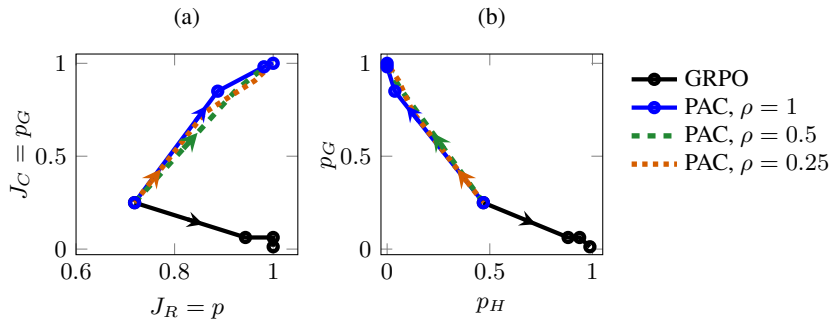

Figure 2: Reward hacking and selective control in the language model, starting from an SFT demonstration mixture $G/H/N = 0.3/0.3/0.4$. (a) Correctness $J_C = p_G$ against verifier reward $J_R = p$. GRPO increases reward while reducing correctness; the verifier rewards correct responses and hacks equally. (b) Correctness against the probability p_H of hacks. PAC increases correctness and reduces hacks at all three audit probabilities ρ, including $\rho = 0.25$. Curves show means over five seeds at calibration rounds 0, 5, 10, and 20. Circles mark these evaluations for GRPO and PAC with $\rho = 1$; arrows indicate training direction. Appendix D.3 reports variation across seeds.

PAC audit each accepted response independently with probability 1, 0.5, or 0.25. Appendix D.3 provides the experimental details.

Results. Figure 2 (a) shows verifier reward J_R increasing while correctness J_C falls under GRPO, which illustrates reward hacking in the sense of Proposition 3.1. Panel (b) shows PAC increasing p_G and decreasing p_H from initialization at all three audit probabilities $\rho \in \{0.25, 0.5, 1.0\}$, consistent with Theorem 5.1. On test prompts, GRPO reaches 99.4% acceptance but only 2.2% correctness. PAC reaches 97.2% correctness with full auditing and 95.2% with one quarter of accepted responses audited. Additional experiments and ablations are provided in Appendix D.3.

7 Related Work

Improving a proxy reward can reduce the intended reward (Everitt et al., 2017; Skalse et al., 2022), and experiments show this reduction growing with stronger optimization (Gao et al., 2023) and appearing under automated verifiers (Helff et al., 2026). Methods that mitigate reward hacking constrain the policy (Laidlaw et al., 2025), improve feedback (Coste et al., 2024; Lightman et al., 2024), or detect and correct hacking (Baker et al., 2025; Wang et al., 2026a), and their guarantees depend on what they observe or assume about correctness. We complement this work by fixing a verifier and characterizing theoretically when reward hacking grows, when the information available during training leaves correct and hacked responses indistinguishable, and when audits can reduce hacking while preserving progress on the intended task. Appendix B extends this discussion.

8 Conclusion

This work analyzes RLVR under a fixed imperfect verifier. First, we derive when the share of accepted errors grows, through hack bias and correctness-to-hack leakage, and when verifier reward rises while correctness falls (Propositions 3.2 and 3.1). Second, the complete training record cannot, in general, detect accepted errors, identify correctness, or guarantee fewer accepted errors while preserving correct responses (Propositions 4.1, 4.2, and 4.3). Third, projected audit correction uses correctness labels from audits of some responses to reduce reward hacking while improving correctness, under the conditions of Theorem 5.1. Together, these results show that controlling reward hacking requires information about correctness beyond the verifier. Incomplete audits and inaccurate labels can weaken this correction. Our analysis assumes a fixed verifier and exact gradient flow, and Appendix A discusses limitations and practical considerations. Extending the guarantees throughout training and correcting from limited, imperfect audits remain future work.

AI USE STATEMENT

In this work, we used generative AI tools to provide critical ingredients for proving mathematical claims, assist in the writing of proofs, design or provide feedback on the research methodology and experiments, and implement methods. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, support qualitative and thematic data analysis, or interpret results, and generating synthetic datasets, assisting with translation, and cleaning and reformatting datasets are not applicable to this work. Additionally, we used generative AI tools to create or edit software code, edit the paper to improve readability, identify relevant literature, and search for information. We have reviewed all AI-assisted work. The authors checked every proof step by step, the first author verified and tested all code generated by AI for correctness, and the authors checked every suggested reference against its original source. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The assumptions and claims behind our theoretical results appear in Propositions 3.1, 3.2, 4.1, 4.2, 4.3 and Theorems 5.1, with complete proofs and supporting derivations in Appendix C. Appendix D documents our full experimental setup. The accompanying anonymous GitHub repository, available at `https://anonymous.4open.science/r/Iclr_verifier_error/README.md`, contains the experimental code and configuration files to reproduce the experiments and figures.

REFERENCES

Johannes Ackermann, Michael Noukhovitch, Takashi Ishida, and Masashi Sugiyama. Gradient regularization mitigates reward hacking in reinforcement learning from human feedback and verifiable rewards. In *Forty-third International Conference on Machine Learning*, 2026.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*, 2025.

Mohammad Beigi, Ming Jin, Junshan Zhang, Jiaxin Zhang, Qifan Wang, and Lifu Huang. IR^3: Contrastive inverse reinforcement learning for interpretable detection and mitigation of reward hacking. *arXiv preprint arXiv:2602.19416*, 2026.

Xin-Qiang Cai, Wei Wang, Feng Liu, Tongliang Liu, Gang Niu, and Masashi Sugiyama. Reinforcement learning with verifiable yet noisy rewards under imperfect verifiers. *arXiv preprint arXiv:2510.00915*, 2025.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *International Conference on Learning Representations*, pp. 50905–50931, 2024.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025.

Carson Denison, Monte MacDiarmid, Fazl Barez, David Duvenaud, Shauna Kravec, Samuel Marks, Nicholas Schiefer, Ryan Soklaski, Alex Tamkin, Jared Kaplan, Buck Shlegeris, Samuel R. Bowman, Ethan Perez, and Evan Hubinger. Sycophancy to subterfuge: Investigating reward-tampering in large language models. *arXiv preprint arXiv:2406.10162*, 2024.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alexander Nicholas D'Amour, Krishnamurthy Dj Dvijotham, Adam Fisch, Katherine A. Heller, Stephen Robert Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. In *First Conference on Language Modeling*, 2024.

Chris Elliott, Einar Urdshals, David Quarel, and Daniel Murfet. Interpreting reinforcement learning agents with susceptibilities. *arXiv preprint arXiv:2605.08007*, abs/2605.08007, 2026.

Tom Everitt, Victoria Krakovna, Laurent Orseau, and Shane Legg. Reinforcement learning with a corrupted reward channel. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 4705–4713, 2017.

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

Etienne Gauthier, Francis R. Bach, and Michael I. Jordan. Explaining and preventing alignment collapse in iterative RLHF. *arXiv preprint arXiv:2605.04266*, 2026.

Lukas Helff, Quentin Delfosse, David Steinmann, Ruben Härle, Hikaru Shindo, Patrick Schramowski, Wolfgang Stammer, Kristian Kersting, and Felix Friedrich. Llms gaming verifiers: Rlvr can lead to reward hacking. *arXiv preprint arXiv:2604.15149*, 2026.

Jacek Karwowski, Oliver Hayman, Xingjian Bai, Klaus Kiendlhofer, Charlie Griffin, and Joar Max Viktor Skalse. Goodhart's law in reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Zachary Kenton, Lili Janzer, Rory Greig, Tian Huey Teh, Kirill Tyshchuk, Jonah Brown-Cohen, Harri Edwards, Senthooran Rajamanoharan, Noah Y. Siegel, Natasha Jaques, and Rohin Shah. Debate training reduces reward hacking in RLAIF. *arXiv preprint arXiv:2608.17776*, 2026.

Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio Calmon. Inference-time reward hacking in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Muhammad Khalifa, Zohaib Khan, Omer Tafveez, Hao Peng, and Lu Wang. Countdown-code: A testbed for studying the emergence and generalization of reward hacking in RLVR. *arXiv preprint arXiv:2603.07084*, abs/2603.07084, 2026.

Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

Thomas Kwa, Drake Thomas, and Adrià Garriga-Alonso. Catastrophic goodhart: regularizing RLHF with KL divergence does not mitigate heavy-tailed reward misspecification. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Cassidy Laidlaw, Shivam Singhal, and Anca Dragan. Correlated proxies: A new definition and improved mitigation for reward hacking. In *The Thirteenth International Conference on Learning Representations*, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Monte MacDiarmid, Benjamin Wright, Jonathan Uesato, Joe Benton, Jonathan Kutasov, Sara Price, Naia Bouscal, Samuel R. Bowman, Trenton Bricken, Alex Cloud, Carson Denison, Johannes Gasteiger, Ryan Greenblatt, Jan Leike, Jack Lindsey, Vladimir Mikulik, Ethan Perez, Alex Rodrigues, Drake Thomas, Albert Webson, Daniel M. Ziegler, and Evan Hubinger. Natural emergent misalignment from reward hacking in production RL. *arXiv preprint arXiv:2511.18397*, 2025.

Anas Mahmoud, MohammadHossein Rezaei, Zihao Wang, Anisha Gunjal, Bing Liu, and Yunzhong He. Reward hacking in rubric-based reinforcement learning. In *Second Workshop on Agents in the Wild: Safety, Security, and Beyond*, 2026.

Christian Moya and Jiankang Wang. Developing correlation indices to identify coordinated cyberattacks on power grids. *IET Cyber-Physical Systems: Theory & Applications*, 3(4):178–186, 2018.

Christian Moya, Alex Semendinger, Guang Lin, and Elliott Thornley. Spurious correlation learning in preference optimization: Mechanisms, consequences, and mitigation via tie training. In *Forty-third International Conference on Machine Learning*, 2026.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*, 2022.

Jane Pan, He He, Samuel R. Bowman, and Shi Feng. Spontaneous reward hacking in iterative self-refinement. *arXiv preprint arXiv:2407.04549*, 2024.

Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. Attack detection and identification in cyber-physical systems. *IEEE transactions on automatic control*, 58(11):2715–2729, 2013.

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

George Wang and Daniel Murfet. Patterning: The dual of interpretability. *arXiv preprint arXiv:2601.13548*, abs/2601.13548, 2026.

Songtao Wang, Quang Hieu Pham, Fangcong Yin, Xinpeng Wang, Jocelyn Qiaochu Chen, Greg Durrett, and Xi Ye. Detecting and suppressing reward hacking with gradient fingerprints. In *Third Conference on Language Modeling*, 2026a.

Xinpeng Wang, Nitish Joshi, Barbara Plank, Rico Angell, and He He. Is it thinking or cheating? detecting implicit reward hacking by measuring reasoning effort. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Xuekang Wang, Zhuoyuan Hao, Shuo Hou, Hao Peng, Juanzi Li, and Xiaozhi Wang. Reproducing, analyzing, and detecting reward hacking in rubric-based reinforcement learning. *arXiv preprint arXiv:2606.04923*, 2026c.

Ziqian Zhong, Aditi Raghunathan, and Nicholas Carlini. ImpossibleBench: Measuring LLMs' propensity of exploiting test cases. *arXiv preprint arXiv:2510.20270*, 2025.

Simon Zhuang and Dylan Hadfield-Menell. Consequences of misaligned ai. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15763–15773, 2020.

A Limitations and Practical Considerations

This appendix states the assumptions that limit the scope of our theoretical guarantees and describes how to implement the audit correction.

A.1 Limitations

No false negatives. To isolate whether rewarding hacks can reduce correctness p_G, we assume that the verifier accepts every correct response. Under this assumption, a decline in p_G cannot come from the verifier rejecting correct responses. The assumption also gives $p = p_G + p_H$, because the accepted responses then consist of all correct responses and all hacked responses. Thus, any change that reduces p_H without decreasing p increases p_G.

If false negatives are allowed, p_G still denotes total correctness, but the probability of rejected correct responses adds a term:

$$p_G = p - p_H + \Pr_{\theta}(c = 1, R = 0).$$

Differentiating along the flow and using $\dot{p} = \|g_R\|^2$ gives

$$\dot{p}_G = \|g_R\|^2 - \dot{p}_H + \frac{d}{dt} \Pr_{\theta(t)}(c = 1, R = 0).$$

The first two terms give the rate at which the probability of accepted correct responses changes. Thus, even when $\dot{p}_H < 0$, total correctness decreases whenever

$$\frac{d}{dt} \Pr_{\theta(t)}(c = 1, R = 0) < -\big(\|g_R\|^2 - \dot{p}_H\big),$$

that is, whenever the probability of rejected correct responses falls faster than the probability of accepted correct responses rises. Extending the control guarantee to this case therefore requires a lower bound on $\frac{d}{dt} \Pr_{\theta(t)}(c = 1, R = 0)$. Proposition 3.2, which decomposes the growth of the hacked share, still applies within the accepted population, with the average score of accepted correct responses in place of $\bar{s}_G$.

Population gradient flow. We analyze updates in continuous time driven by the exact gradient of J_R. Exact gradients remove sampling noise and isolate the effect of verifier rewards. Practical training departs from this idealization through finite steps, adaptive optimizers such as Adam, gradient clipping, and additional objectives such as KL regularization, each of which can change the dynamics. Our growth and control guarantees therefore hold exactly only for the stated flows. Our experiments with discrete updates check whether the predicted behavior persists in the tested settings, but they do not extend the guarantees to other training algorithms.

Local growth and control over a finite interval. Proposition 3.2 characterizes the growth of the hacked share at the current policy. The group scores change during training, so a positive growth rate at one time does not guarantee continued growth. Theorem 5.1 extends the analysis from a single time to the interval $[0, T]$, but its bound requires the assumptions to hold throughout this interval and does not establish convergence beyond it. Similarly, the identity $\dot{p} = \|g_R\|^2$ under the projected correction is local: the correction preserves the rate at which acceptance grows at the current policy. Because the correction changes the trajectory of the policy, this identity does not imply that corrected training follows the same acceptance curve or reaches the same final acceptance as uncorrected training.

Scope of the information limits. Our impossibility results (Section 4) concern methods that observe only the training record L_t: no such method can provide a uniform guarantee, one that holds under every correctness assignment in $\mathcal{C}_R$. These results do not imply that every monitor fails on every task. Additional knowledge of task requirements can exclude assignments in $\mathcal{C}_R$, and if it excludes the indistinguishable assignments used in our proofs, our impossibility arguments no longer apply. Conversely, allowing false negatives enlarges the class of assignments, and the enlarged class still contains these indistinguishable assignments, so it admits no uniform guarantee either.

Fixed verifier and prompt distribution. We hold the binary verifier, correctness labels, and prompt distribution fixed during training. In practice, the verifier or prompt distribution can change during training, for example when tests are added or a curriculum reorders prompts. Such changes shift

population probabilities without any policy update, and our dynamics exclude them. Because p_G and p_H average over the prompt distribution, increasing p_G or decreasing p_H does not guarantee that correctness improves on every prompt.

A.2 Practical Considerations

Limited audit coverage. We assumed full audit coverage in Assumption (A2) for Theorem 5.1, so that $\nabla p_{H_A} = \nabla p_H$. Without full coverage, the correction changes the growth of p_H by $\nabla p_H^\top u = -\lambda (P_\perp \nabla p_H)^\top (P_\perp \nabla p_{H_A})$. When the projected gradients align positively, the correction opposes the growth of p_H and, as in Theorem 5.1, equally promotes the growth of p_G. Full coverage guarantees positive alignment when $P_\perp \nabla p_H \neq 0$, but is not necessary: a subset of audits can supply a direction that opposes hacking. We next ask whether this direction remains effective as the policy changes.

Sustaining corrective strength. A direction that opposes hacking at one policy may weaken or reverse as on-policy training shifts the distribution and gradients. Accurate audits are not enough: they must continue to supply sufficient corrective strength throughout training. Retaining the other assumptions of the theorem, the same ISS-type bound holds under partial coverage if $(P_\perp \nabla p_H)^\top (P_\perp \nabla p_{H_A})/p \geq \kappa q$ throughout the interval for some fixed $\kappa > 0$; under full coverage, this condition reduces to (A3). We next ask how to estimate the corrective direction from finite samples.

Estimating the correction from finite audits. The correction uses a population gradient, but training provides only sampled responses and audit labels. Randomized audit selection provides a way to estimate the full hack gradient without auditing every generated response in each batch. For samples from the current policy, we estimate $\nabla_\theta p_H$ using each audited hack's policy score, $\nabla_\theta \log \pi_\theta(y \mid x)$, divided by its selection probability. Audited correct responses contribute zero. Averaging these weighted contributions over all generated responses yields an unbiased estimate $\widehat{h}$, provided these selection probabilities are known and positive for accepted responses and the regularity conditions in Appendix C.4.3 hold. Accepted responses with zero selection probability remain outside this guarantee. Even with sufficient coverage, small selection probabilities produce large weights and can yield noisy estimates, so an individual correction may fail to oppose hacking. While sparse audits can recover the direction in expectation, reliable correction requires controlling the variance of this estimate.

Implementing the projection $P_\perp$. The projection also requires an estimate $\widehat{g}_R$ of the reward gradient. Set the correction orthogonal to the estimated reward gradient: $\widehat{g}_R^\top \widehat{u} = 0$. To isolate projection error, we keep the base verifier gradient exact and consider $\dot{\theta} = g_R + \widehat{u}$. Then

$$\dot{p} = \|g_R\|^2 + (g_R - \widehat{g}_R)^\top \widehat{u},$$

where the second term is the projection error. Thus, orthogonality to the estimated gradient need not preserve the true instantaneous growth rate of p. Estimating the base verifier gradient introduces additional error. Optimizer transformations and finite step sizes can introduce further discrepancies. Practical implementation requires validating that discrete updates oppose hacking while approximately preserving acceptance progress.

Our analysis identifies three practical tasks: selecting which responses to audit, estimating the corrective direction from these audits, and validating that each update opposes hacking while approximately preserving acceptance progress. Susceptibility methods for reinforcement learning (Elliott et al., 2026) and patterning (Wang & Murfet, 2026) could help guide audit selection and correction. Adapting these methods to sustain selective control under limited audit and computation budgets remains an open engineering challenge.

B Related Work

We position our work within two lines of research on reward hacking: why optimizing a proxy reward can undermine the intended reward, and how to prevent it.

Reward hacking. Theoretical work shows that improving a proxy reward can reduce the intended reward (Everitt et al., 2017; Skalse et al., 2022). This conflict can arise when the proxy omits relevant attributes (Zhuang & Hadfield-Menell, 2020), and its severity depends on properties of the proxy and its errors (Laidlaw et al., 2025; Kwa et al., 2024). In experiments, stronger optimization and more capable models can widen the gap between proxy and intended rewards (Gao et al., 2023; Pan et al., 2022). Language models also show this gap during iterative self-refinement (Pan et al., 2024) and when automated verifiers accept responses that violate task requirements (Helff et al., 2026; Zhong et al., 2025; Mahmoud et al., 2026). Training can amplify such reward hacking (Khalifa et al., 2026), and learned hacking can generalize to reward tampering or broader misalignment (Denison et al., 2024; MacDiarmid et al., 2025). To explain how reward hacking develops, prior work studies the geometry of optimization (Karwowski et al., 2024), selection at inference time (Khalaf et al., 2025), and feedback between policy training and retraining of the reward model (Gauthier et al., 2026). For rewards assigned by judges, Wang et al. (2026c) further separate how easily models discover biases from how strongly they exploit them. Compared to these works, we study how reward hacking grows under a fixed binary verifier by deriving conditions under which policy updates favor hacked responses over equally rewarded correct responses. We then show that the reward objective provides no preference that would restore correct responses within the accepted set, which explains why reward maximization alone need not reverse this growth.

Mitigating reward hacking. Regularization can limit reward hacking by constraining how much a policy changes (Laidlaw et al., 2025) or by favoring flatter optima, a property that theory links to the accuracy of rewards under additional assumptions (Ackermann et al., 2026). Other methods improve the feedback used for optimization by combining reward models (Coste et al., 2024), adding constraints (Helff et al., 2026), supervising reasoning steps (Lightman et al., 2024), or supplying criticism through debate (Kenton et al., 2026). When verifier errors follow an assumed noise model, known error rates can also support corrections to training updates (Cai et al., 2025). Even improved feedback can leave errors unresolved, as Eisenstein et al. (2024) show for reward ensembles. To detect reward hacking, monitors inspect a model's reasoning (Baker et al., 2025) or measure how much reasoning it needs to pass the verifier (Wang et al., 2026b). Detection can then guide correction: Grift groups responses by their gradients, labels each group by inspecting examples, and filters responses for fine-tuning (Wang et al., 2026a), while IR^3 reconstructs rewards and targets features it identifies as problematic (Beigi et al., 2026). The guarantees of these approaches depend on what they observe or assume about correctness, a dependence that Everitt et al. (2017) examine for corrupted rewards. Our identifiability results specify when the information available during training leaves correct and hacked responses indistinguishable. We then characterize how auditing supplies information for selective correction and how limited coverage or errors in the audits can undermine this correction.

C Technical Details and Proofs

C.1 Probabilities and Dynamics for Section 2

Idealized conditions. We use the fixed verifier R and correctness rule c from Section 2, with $c \leq R$, so the verifier has no false negatives. Without correction, the parameters follow exact Euclidean gradient ascent on the verifier reward, $\dot{\theta} = \nabla J_R(\theta)$. We assume that the objectives and group probabilities are continuously differentiable and that the stated flows exist on the intervals considered. When differentiating expectations, we assume sufficient regularity to interchange differentiation and expectation.

C.1.1 Fixed Sets of Correct Responses and Accepted Errors

For each prompt x, the assumption $c(x, y) \leq R(x, y)$ gives $G_x \subseteq A_x$ and hence

$$H_x = A_x \setminus G_x = \{y \in \mathcal{Y}_x : R(x, y) = 1,\ c(x, y) = 0\},$$
$$A_x = G_x \sqcup H_x,$$

where $\sqcup$ denotes disjoint union. These memberships stay fixed throughout training. Their probabilities can change, and H_x may be empty.

C.1.2 Acceptance and the Hacked Share

At a fixed prompt, conditioning on acceptance gives

$$q_x(\theta) = \frac{\Pr_{\pi_\theta}(Y \in H_x \mid x)}{p_x(\theta)},$$
$$\Pr_{\pi_\theta}(Y \in H_x \mid x) = p_x(\theta) q_x(\theta), \qquad \text{when } p_x(\theta) > 0.$$
$$\Pr_{\pi_\theta}(Y \in G_x \mid x) = p_x(\theta)\big(1 - q_x(\theta)\big),$$

If $p_x = 0$, the probability of an accepted error is zero and q_x is undefined. Since the binary verifier rewards every response in A_x, iterated expectation gives

$$J_R(\theta) = \mathbb{E}_{x \sim \mathcal{D}}\big[\mathbb{E}_{Y \sim \pi_\theta(\cdot|x)} R(x, Y)\big] = \mathbb{E}_{x \sim \mathcal{D}}[p_x(\theta)].$$

The objective therefore depends on total acceptance, without distinguishing its correct and incorrect parts.

C.1.3 Exact Gradient Flow

Along the flow (1), the chain rule gives

$$\dot{J}_R = \nabla_\theta J_R^\top \dot{\theta} = \|g_R\|_2^2 \geq 0, \qquad \dot{p}_x = \nabla_\theta p_x^\top g_R, \qquad \dot{q}_x = \nabla_\theta q_x^\top g_R,$$

where the identity for $\dot{q}_x$ applies when $p_x > 0$. Under the stated assumptions, there is no general guarantee that $\dot{q}_x \leq 0$ or that $\dot{p}_x \geq 0$ at every prompt. Thus, monotonic improvement in average acceptance guarantees neither a nonincreasing hacked share nor nondecreasing acceptance at every prompt.

C.2 Technical Details for Section 3

C.2.1 Aggregate Acceptance and Hacking

Write $\mathbb{E}_\theta$ for expectation under $X \sim \mathcal{D}$, $Y \mid X \sim \pi_\theta$. Since $G \cup H$ is the accepted set,

$$p = J_R = p_G + p_H = \mathbb{E}_{x\sim\mathcal{D}}[p_x],$$

$$q = \frac{p_H}{p} = \frac{\displaystyle\int_{\{x:p_x>0\}} p_x q_x \, \mathcal{D}(dx)}{p} \quad \text{when } p > 0.$$

Thus, relative to the prompt distribution $\mathcal{D}$, prompts are weighted by their acceptance probabilities. Prompts with zero acceptance contribute nothing. An aggregate trend need not hold at every prompt. At fixed p, redistributing probability between G and H leaves reward unchanged, although shared parameters can couple their dynamics.

C.2.2 Proof of Proposition 3.1

Skalse et al. (2022) call a proxy reward hackable if it ranks some pair of policies in strictly opposite orders from the true reward. In our setting, verifier flow generates the policies, acceptance $J_R = p$ is the proxy reward, and correctness $J_C = p_G$ is the true reward. We show that such a pair occurs along the trajectory exactly when, at some time, the correctness lost to a rising hacked share outweighs the correctness gained from rising acceptance.

Proof. Write $p(t) = p(\theta(t))$ and similarly for the other quantities. Since $p_G = p(1-q)$,

$$\dot{J}_C = \dot{p}_G = \underbrace{(1-q)\dot{p}}_{\text{acceptance contribution}} - \underbrace{p\dot{q}}_{\text{composition contribution}}.$$

Consequently, the proposed inequality is equivalent to $\dot{J}_C < 0$. The key point is that, along verifier flow, a strict decrease in correctness necessarily accompanies a strict increase in verifier reward.

Sufficiency. Suppose that, at some $t_\star \in (0, T)$,

$$p(t_\star)\dot{q}(t_\star) > (1 - q(t_\star))\dot{p}(t_\star).$$

Then $\dot{J}_C(t_\star) < 0$. Since

$$\dot{J}_C(t_\star) = \nabla_\theta J_C(\theta(t_\star))^\top g_R(\theta(t_\star)),$$

we must have $g_R(\theta(t_\star)) \neq 0$. Hence

$$\dot{J}_R(t_\star) = \|g_R(\theta(t_\star))\|_2^2 > 0.$$

By continuity, both strict inequalities hold on a sufficiently short interval $[t_\star, t_\star + \varepsilon] \subset (0, T)$. Integrating over this interval gives

$$J_R(\theta(t_\star + \varepsilon)) > J_R(\theta(t_\star)),$$
$$J_C(\theta(t_\star + \varepsilon)) < J_C(\theta(t_\star)).$$

Thus the two policies at its endpoints witness reward hacking.

Necessity. Conversely, suppose there exist $t_1 < t_2$ in $[0, T]$ such that verifier reward increases strictly and correctness decreases strictly. By the mean value theorem, there is some $t_\star \in (t_1, t_2)$ with

$$\dot{p}_G(t_\star) = \frac{p_G(t_2) - p_G(t_1)}{t_2 - t_1} < 0.$$

Substituting $\dot{p}_G = (1-q)\dot{p} - p\dot{q}$ and rearranging yields

$$p(t_\star)\dot{q}(t_\star) > (1 - q(t_\star))\dot{p}(t_\star),$$

as required. □

The criterion distinguishes a growing hacked share from reward hacking in the sense of Skalse et al. (2022). A growing hacked share can coexist with improving correctness if acceptance rises fast enough. The two rewards disagree exactly when, at some time along the flow, the correctness lost to a rising hacked share exceeds the correctness gained from rising acceptance. This inequality needs to hold strictly at only a single time: by continuity, it then holds on a short interval whose endpoints witness hackability, even if correctness later recovers.

C.2.3 Proof of Proposition 3.2

When $p_H, p_G > 0$, total acceptance cancels in the ratio:

$$z = \log \frac{p_H}{p_G} = \log \frac{q}{1-q}, \qquad \frac{dz}{dq} = \frac{1}{q(1-q)} > 0.$$

The log odds z therefore increase with the hacked share q. They are finite only when correct responses and hacks both have positive probability, so we require this condition wherever we use finite log odds.

Evolution of the log odds. We derive (2) using the gradients $\bar{s}_S = \nabla_\theta \log \Pr_\theta(S)$ from Section 3.

Differentiate the log ratio. For $p_H, p_G > 0$, the log odds $z = \log(p_H/p_G)$ are finite, and differentiating along the flow gives

$$\dot{z} = \frac{\dot{p}_H}{p_H} - \frac{\dot{p}_G}{p_G}.$$

The rate therefore compares the proportional growth of the two groups. This identity needs only differentiability of the group probabilities.

To express each proportional rate through the policy, we use the score $s_\theta(x, y) := \nabla_\theta \log \pi_\theta(y \mid x)$. For a fixed group S with positive probability and indicator $\mathbf{1}_S$, the conditions for differentiating expectations give

$$\nabla_\theta \Pr_\theta(S) = \mathbb{E}_\theta[\mathbf{1}_S s_\theta] = \Pr_\theta(S)\, \bar{s}_S, \qquad \bar{s}_S := \frac{\mathbb{E}_\theta[\mathbf{1}_S s_\theta]}{\Pr_\theta(S)} = \mathbb{E}_\theta[s_\theta \mid S].$$

The gradient of $\log \Pr_\theta(S)$ is thus the mean score $\bar{s}_S$ of the group. Applying this to H and G, and using $\dot{\theta} = g_R$, gives

$$\nabla_\theta z = \bar{s}_H - \bar{s}_G, \qquad \dot{z} = (\bar{s}_H - \bar{s}_G)^\top g_R.$$

The log odds therefore grow precisely when $g_R^\top \bar{s}_H > g_R^\top \bar{s}_G$.

Decomposition of the growth rate. Assume G, H, N all have positive probability and evaluate quantities at the current policy.

Step 1: Use normalization. Differentiating the accepted probability and the sum of all group probabilities gives

$$g_R = p\big[(1-q)\bar{s}_G + q\bar{s}_H\big], \qquad g_R + (1-p)\bar{s}_N = 0.$$

Combining these identities yields

$$g_R = p(1-p)\big[(1-q)\bar{s}_G + q\bar{s}_H - \bar{s}_N\big].$$

Step 2: Substitute into the rate of the log odds. Using (2) and collecting $\bar{s}_H - \bar{s}_G$ gives

$$\begin{aligned}\dot{z} &= p(1-p)(\bar{s}_H - \bar{s}_G)^\top \big[(1-q)\bar{s}_G + q\bar{s}_H - \bar{s}_N\big] \\ &= p(1-p)\left[\underbrace{(\bar{s}_H - \bar{s}_G)^\top(\bar{s}_G - \bar{s}_N)}_{\text{correctness-to-hack leakage}} + \underbrace{q\|\bar{s}_H - \bar{s}_G\|^2}_{\text{hack bias}}\right],\end{aligned}$$

which establishes (3) and reveals the mechanisms. The first term can favor or oppose hacking. The second is nonnegative and proportional to q at fixed p and gradients; those quantities also change during training. Hence growth depends on the full bracket, without implying acceleration or growth at every policy.

Step 3: State the growth condition. When $\bar{s}_H \neq \bar{s}_G$, the bracket is positive exactly when

$$q > -\frac{(\bar{s}_H - \bar{s}_G)^\top(\bar{s}_G - \bar{s}_N)}{\|\bar{s}_H - \bar{s}_G\|^2}.$$

The threshold may lie outside $(0, 1)$ and change during training. If the two gradients agree, $\dot{z} = 0$. Thus the current share alone does not determine growth. The alignment of the gradients also matters.

Simultaneous growth of reward and hacking For $p_H, p_G > 0$, differentiating $z = \log(q/(1-q))$ and using the verifier flow gives

$$\dot{q} = q(1-q)\dot{z}, \qquad \dot{p} = \|g_R\|^2.$$

Since $p_G, p_H, p_N > 0$, both p and q lie in $(0, 1)$. Thus, $\dot{q} > 0$ if and only if the bracket in (3) is positive. By (2), a positive $\dot{z}$ requires $g_R \neq 0$. Both $\dot{q}$ and $\dot{p}$ are then strictly positive, completing the proof of Proposition 3.2. □

Example: Shared normalization can favor hacking. For one prompt with a response in each group, use softmax logits $\theta = (\theta_G, \theta_H, \theta_N)$:

$$\Pr_\theta(S) = \frac{e^{\theta_S}}{e^{\theta_G} + e^{\theta_H} + e^{\theta_N}}, \qquad S \in \{G, H, N\}.$$

Here $z = \theta_H - \theta_G$ and

$$g_R = \big(p_G(1-p),\ p_H(1-p),\ -p(1-p)\big)^\top,$$
$$\dot{z} = (1-p)(p_H - p_G).$$

Thus $p_H > p_G$ gives simultaneous growth of the hacked share and verifier acceptance on a sufficiently short interval. The example illustrates a mechanism caused by the parameterization; it does not assert that every neural policy follows this pattern.

C.3 Technical Details for Section 4

C.3.1 Comparison Setup

Fix the prompt distribution, verifier, policy family, initialization, and training procedure. The record L_t contains prompts, responses, verifier rewards, policy updates, and quantities computed from these observations. Training and monitoring may adapt to the record, but neither uses the hidden correctness assignment or additional correctness feedback.

We compare two fixed correctness assignments:

$$c_0 = R, \qquad c_1 \in \mathcal{C}_R \quad \text{with} \quad \Pr_\theta(R = 1, c_1 = 0) > 0,$$

at the policy θ. Under c_0, every accepted response is correct. Under c_1, some accepted responses are hacks.

The assignments are fixed before training. We compare what the same procedure observes under each assignment. Correctness does not change during either run. For the deterministic argument below, fix all random choices used by training and use those same choices in both runs.

C.3.2 Proof of Proposition 4.1

The proof has three steps: the assignments produce the same record, the monitor therefore makes the same decision, but correct detection requires different decisions.

Step 1: The training records are identical. We argue by induction on the training step. Both runs start from the same initialization, so their initial records agree. Suppose their records agree through step n. Because the training procedure depends only on the record and the shared random choices, both runs select the same prompt and sample the same response from the same policy. The fixed verifier returns the same reward. Both runs therefore make the same update and append the same information to the record, so their records agree through step $n + 1$.

By induction, the records are identical at every training step:

$$L_n^{(c_0)} = L_n^{(c_1)} \qquad \text{for every } n.$$

The argument also covers verifier queries that depend on earlier results and any quantity computed from the record. It likewise covers J_R and its gradients, which depend on the verifier and the policy but not on which correctness assignment in $\mathcal{C}_R$ is true. For training in continuous time, we assume that the fixed procedure determines a unique trajectory. Because this trajectory depends only on the verifier and the policy, changing the correctness assignment within $\mathcal{C}_R$ leaves it unchanged. This argument applies to any two fixed assignments in $\mathcal{C}_R$. It does not require either assignment to equal R.

Step 2: The monitor makes the same decision. A deterministic detection monitor computes a binary alarm $D_t = \delta_t(L_t)$, where $D_t = 1$ indicates the presence of hacks. Identical records give identical alarms:

$$D_t^{(c_0)} = \delta_t(L_t^{(c_0)}) = \delta_t(L_t^{(c_1)}) = D_t^{(c_1)}.$$

Step 3: The same decision cannot be correct in both cases. Under c_0, there are no hacks, so a correct detector must remain silent. Under c_1, hacks have positive probability, so a correct detector must raise an alarm. Because the monitor makes the same decision in both cases, it either raises a false alarm under c_0 or misses the hacks under c_1.

Thus, no deterministic monitor using only L_t can guarantee correct detection under both compatible assignments. More observations from the same verifier do not resolve this ambiguity: Step 1 continues to apply as the record grows.

Scope. This conclusion concerns a guarantee over the stated class $\mathcal{C}_R$. A monitor may succeed under a particular assignment. Additional knowledge connecting response content to correctness may also rule out alternatives. The impossibility applies when the two assignments remain admissible and the training procedure never observes information that distinguishes them.

Remark on randomization Randomization does not change the conclusion. Use the same random choices for training and monitoring in the two runs. For every such choice, the records and alarms remain identical. Averaging over these choices therefore gives

$$\Pr_{c=c_1}(D_t = 1) = \Pr_{c=c_0}(D_t = 1).$$

Here the probabilities are over training and monitor randomness. Each correctness assignment remains fixed. The left side is the detection rate when hacks exist, and the right side is the false-alarm rate when they do not. Thus, the monitor has no detection advantage between these two assignments.

Likewise, the training records have the same distribution:

$$\mathrm{Law}_{c=c_1}(L_t) = \mathrm{Law}_{c=c_0}(L_t),$$

where $\mathrm{Law}_c(L_t)$ denotes the distribution of the record over training randomness under assignment c. This proves the proposition. □

C.3.3 Proof of Proposition 4.2

The proof compares two compatible assignments that disagree on every accepted response. Both admit hacks, so knowing that hacks exist does not distinguish them.

Step 1: Construct two opposite correctness assignments. By the premise, choose a fixed assignment $c_1 \in \mathcal{C}_R$ under which the policy θ assigns positive probability to both correct responses and hacks. Define

$$c_2(x, y) = R(x, y) - c_1(x, y).$$

On rejected responses, both assignments equal zero. On accepted responses, $c_2 = 1 - c_1$, so the assignments give opposite correctness labels. Thus $c_2 \in \mathcal{C}_R$, and the two assignments exchange the correct and hacked groups:

$$p_{H,c_2}(\theta) = p_{G,c_1}(\theta) > 0, \qquad p_{G,c_2}(\theta) = p_{H,c_1}(\theta) > 0.$$

In particular, both assignments admit hacks with positive probability.

Step 2: The monitor makes the same prediction. Fix any accepted pair (x, y) and any deterministic monitor using only verifier feedback records. As shown in Appendix C.3.2, the two assignments produce identical training records when the same random choices are used. The monitor therefore makes the same prediction in both cases:

$$\widehat{c}_t^{(c_1)}(x, y) = \widehat{c}_t^{(c_2)}(x, y).$$

Step 3: The prediction cannot be correct under both assignments. Because $R(x, y) = 1$, the true labels satisfy

$$c_2(x, y) = 1 - c_1(x, y).$$

The monitor's common binary prediction therefore matches exactly one of the two labels. It must be incorrect under the other assignment.

Thus, no deterministic monitor can guarantee the correct label of an accepted response under every compatible assignment. Knowing that hacks exist does not resolve the ambiguity, because both assignments satisfy that additional information.

Randomization and the error bound. The same reasoning applies when training or monitoring is randomized. Use the same random choices in both runs. For every such choice, the monitor gives the same prediction under both assignments, and exactly one assignment makes that prediction wrong.

Averaging over these choices gives

$$\Pr_{c=c_1}(\widehat{c}_t(x, y) \neq c_1(x, y)) + \Pr_{c=c_2}(\widehat{c}_t(x, y) \neq c_2(x, y)) = 1. \tag{4}$$

Here the probabilities are over training and monitor randomness, with the queried pair and correctness assignments held fixed. At least one of these error probabilities is therefore at least $1/2$.

We fixed both candidate assignments before training, so neither depends on the realized training record. Which of the two attains the bound may depend on the monitor and the queried pair, but not on that record. Because both assignments contain hacks, the proposition holds even when the monitor knows that hacks exist. This completes the proof . □

C.3.4 Proof of Proposition 4.3

Controlled dynamics and the selective objective. Consider a controller that chooses a correction $u(t)$ using only the training record L_t. Training then follows

$$\dot{\theta} = g_R + u(t).$$

We assume that the controlled trajectory is well defined and that the group probabilities are differentiable along it. For a deterministic argument, fix all random choices used during training.

Under a correctness assignment c, *selective control* requires

$$\dot{p}_{H,c} < 0, \qquad \dot{p}_{G,c} \geq 0 \quad \text{whenever } p_{H,c} > 0.$$

These conditions concern the full update:

$$\dot{p}_{H,c} = \nabla_\theta p_{H,c}^\top (g_R + u), \qquad \dot{p}_{G,c} = \nabla_\theta p_{G,c}^\top (g_R + u).$$

A correction that opposes hack growth is insufficient unless the full update $g_R + u(t)$ reduces hack probability while preserving correctness. A uniform guarantee requires the same controller rule to satisfy these conditions under every compatible correctness assignment and at every time along the evolving policy trajectory whenever hack probability is positive. The correction itself may adapt to the training record.

Impossibility of guaranteed selective control. We compare two assignments that exchange the correct and hacked groups. The controller sees the same record under both assignments, but selective control requires opposite changes in their probabilities.

Step 1: Exchange the correct and hacked groups. By the premise, choose a fixed assignment $c_1 \in \mathcal{C}_R$ under which the initial policy assigns positive probability to both correct responses and hacks. Define

$$c_2 = R - c_1.$$

As in Appendix C.3.3, both assignments are compatible with the verifier, and they exchange the two accepted groups:

$$p_{H,c_2} = p_{G,c_1}, \qquad p_{G,c_2} = p_{H,c_1}.$$

Both groups have positive probability initially. By continuity, they remain positive on a sufficiently short initial interval.

Step 2: The controller produces the same update. We extend the argument of Appendix C.3.2, which shows that the two runs produce identical records, to include the controller. Both runs begin at the same policy. Whenever their records agree, the controller chooses the same correction, because it uses only the record. The verifier gradient also agrees, because it depends on the policy and the verifier but not on the correctness assignment. Both runs therefore make the same corrected update, and their records remain identical.

Thus, under both assignments, the fixed training and control procedures generate the same record and the same trajectory of the controlled policy. In particular, both runs have the same velocity $\dot{\theta} = g_R + u$ at every time.

Step 3: Selective control requires incompatible signs. Along this common trajectory, exchanging the groups also exchanges their derivatives:

$$\dot{p}_{H,c_2} = \dot{p}_{G,c_1}, \qquad \dot{p}_{G,c_2} = \dot{p}_{H,c_1}.$$

On the initial interval where both groups remain positive, selectivity under the two assignments therefore requires

$$\begin{aligned} c_1 &: \quad \dot{p}_{H,c_1} < 0, \quad \dot{p}_{G,c_1} \geq 0, \\ c_2 &: \quad \dot{p}_{G,c_1} < 0, \quad \dot{p}_{H,c_1} \geq 0. \end{aligned} \tag{5}$$

Each derivative would have to be both negative and nonnegative. No update can satisfy both rows.

Thus, no controller that uses only the training record can guarantee selective control under every correctness assignment in $\mathcal{C}_R$. The obstruction appears already on an initial interval of training, so it does not depend on behavior later in training.

Remark on randomization. The same obstruction applies to a randomized controller. We couple the two runs by using the same random choices for training and control under both assignments. For each realization, the records, corrections, and trajectories then coincide, so no realization satisfies the requirements for selective control under both assignments.

A guarantee that holds with probability one under both assignments would require both sets of conditions to hold on a common event of probability one. Since no realization satisfies both, that event is empty, so no such guarantee exists. Randomization cannot help, because the two candidate assignments are fixed before training and the random choices therefore carry no information that distinguishes them. The proposition follows. □

Scope. The result rules out a uniform guarantee over $\mathcal{C}_R$, but a controller may still achieve selective control under a particular assignment. Additional information or assumptions can distinguish assignments in $\mathcal{C}_R$, and if they exclude either of the two assignments used in the proof, the argument above no longer applies.

Connection to secure control. The argument here uses an indistinguishability principle also used when studying attack detection and identification in cyber-physical systems: no monitor can distinguish admissible scenarios that produce identical observations when it uses only those observations (Pasqualetti et al., 2013). In our paper, the indistinguishable scenarios are correctness assignments that produce the same training record L_t. Exchanging the correct and hacked groups between two such assignments makes their requirements for selective control incompatible. The argument therefore identifies the information that a uniform guarantee needs: observations or structural assumptions that exclude one of these assignments. In power grid security, models of grid operation supply such structural correctness knowledge: by relating coordinated cyberattacks to their physical consequences, they tell defenders which scenarios cause harm and let them target defense accordingly (Moya & Wang, 2018). Audits can play an analogous role by supplying correctness information that distinguishes otherwise compatible assignments. The subsequent analysis examines when this information supports selective control.

C.3.5 Regularization and Its Limitations

We analyze three regularization penalties and distinguish what each controls from what it guarantees about correctness. Throughout, we use exact gradient ascent, fixed penalty weights, and the fixed verifier from Section 2. We then apply Proposition 4.3 to determine which guarantees of selective control these penalties can provide.

KL divergence from a reference For a fixed reference policy under the same prompt distribution, define

$$K(\theta) = \mathbb{E}_{x\sim\mathcal{D}} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x)), \qquad F_\beta = J_R - \beta K,$$

where $\beta > 0$ is fixed.

Step 1: Bound deviation from the reference. Assume the flow remains in a region where K is finite and continuously differentiable. Gradient ascent on F_β gives

$$\dot\theta = g_R - \beta \nabla_\theta K, \qquad \dot F_\beta = \|\nabla_\theta F_\beta\|^2 \geq 0.$$

Thus $F_\beta(\theta(t)) \geq F_\beta(\theta(0))$. Rearranging and using $J_R = p \leq 1$ yields

$$\begin{aligned} K(\theta(t)) &\leq K(\theta(0)) + \frac{p(\theta(t)) - p(\theta(0))}{\beta} \\ &\leq K(\theta(0)) + \frac{1 - p(\theta(0))}{\beta}. \end{aligned}$$

In particular, initialization at the reference gives $K(\theta(0)) = 0$. The penalty therefore bounds departure from the reference along the exact flow.

STEP 2: BOUND CHANGES IN HACK PROBABILITY. Because the prompt distribution is fixed, K also equals the KL divergence between the joint distributions of prompts and responses. Pinsker's inequality therefore gives, for the fixed hacked set H,

$$|p_H(\theta) - p_{H,\mathrm{ref}}| \le \sqrt{\frac{K(\theta)}{2}}, \qquad p_{H,\mathrm{ref}} := \Pr_{\mathrm{ref}}(H).$$

Thus, closeness to the reference limits the change in hack probability.

LIMITATION. The penalty bounds how much the policy changes, not the direction of that change. It therefore guarantees neither $\dot{p}_H < 0$ nor $\dot{p}_G \ge 0$. The bound controls hack probability relative to $p_{H,\mathrm{ref}}$. It implies a small absolute hack probability when both the reference's hack probability and the KL bound are small. The penalty depends on the policy and the reference but not on the correctness assignment, so with the same reference under every assignment in $\mathcal{C}_R$, Proposition 4.3 still applies.

Penalizing the gradient of reward. Following the objective of Ackermann et al. (2026), we consider the idealized gradient regularization penalty

$$F_{\mathrm{GR}} = J_R - \gamma \|g_R\|^2,$$

with fixed $\gamma > 0$.

STEP 1: DERIVE THE CORRECTION. Assume J_R is twice continuously differentiable and write $B_R := \nabla_\theta^2 J_R$. Since B_R is symmetric,

$$\nabla_\theta \|g_R\|^2 = 2 B_R g_R.$$

Gradient ascent on F_{GR} therefore gives

$$\dot{\theta} = g_R - 2\gamma B_R g_R.$$

The correction depends entirely on the verifier objective and its derivatives.

STEP 2: SEPARATE REWARD SENSITIVITY FROM CORRECTNESS. The norm $\|g_R\|$ measures first-order sensitivity of expected verifier reward to parameter changes. It does not by itself determine correctness. For example, if the verifier accepts every response, then $J_R \equiv 1$ and $g_R \equiv 0$. The correction vanishes even if the policy assigns positive probability to incorrect responses.

SCOPE OF THE CITED ANALYSIS. Ackermann et al. (2026) connect flatter optima to the accuracy of the proxy reward under additional assumptions: continuous actions, a Gaussian policy with fixed covariance, regularity conditions on the policy and reward functions, and a true reward that is Lipschitz continuous. Our setup does not impose these policy and reward assumptions, and $\mathcal{C}_R$ places no corresponding regularity restriction on correctness assignments. Thus, the cited results do not establish selective control uniformly over $\mathcal{C}_R$.

Exchanging correctness assignments in $\mathcal{C}_R$ leaves J_R, its derivatives, and the correction unchanged, so Proposition 4.3 applies to this regularizer. This conclusion does not contradict the theoretical results of Ackermann et al. (2026), which hold under the additional assumptions above, or their empirical improvements, which concern particular tasks rather than a uniform guarantee over $\mathcal{C}_R$.

Stability of relative probabilities. Motivated by tie training for reducing reliance on spurious features in preference optimization (Moya et al., 2026), we analyze a simplified penalty on changes in relative response probabilities. The penalty discourages deviations from a fixed reference ratio between two verifier-accepted responses. Unlike pairs constructed to have equal utility, equal verifier rewards alone do not establish that the responses are equally correct. We therefore examine what this penalty guarantees when pair selection uses only verifier information.

Fix two accepted responses y_1, y_2 to a prompt x, with positive probabilities under the current and reference policies. Define

$$e(\theta) = \log \frac{\pi_\theta(y_1 \mid x)}{\pi_\theta(y_2 \mid x)} - \log \frac{\pi_{\mathrm{ref}}(y_1 \mid x)}{\pi_{\mathrm{ref}}(y_2 \mid x)}.$$

Thus $e = 0$ means that the current relative probability matches the reference ratio. Assume e is continuously differentiable on a neighborhood of the trajectory.

STEP 1: DERIVE THE RESTORING TERM. For fixed $\lambda > 0$, gradient ascent on

$$F_{\mathrm{I}} = J_R - \frac{\lambda}{2} e^2$$

adds the correction

$$u(t) = -\lambda e \, \nabla_\theta e.$$

Along the corrected flow,

$$\dot{e} = b - \lambda a e, \qquad b := \nabla_\theta e^\top g_R, \qquad a := \|\nabla_\theta e\|^2.$$

The term b is the change in the log ratio induced by verifier training. The term $-\lambda a e$ opposes deviation from the reference ratio.

STEP 2: BOUND THE DEVIATION. Consider an interval $[0, T]$ on which the flow exists and both response probabilities remain positive. Suppose

$$a(t) \geq m > 0, \qquad |b(t)| \leq B \quad \text{for } 0 \leq t \leq T.$$

An integrating factor gives

$$e(t) = \exp\left(-\lambda \int_0^t a(s)\, ds\right) e(0) + \int_0^t \exp\left(-\lambda \int_\tau^t a(s)\, ds\right) b(\tau)\, d\tau.$$

Using the bounds on a and b,

$$\begin{aligned} |e(t)| &\leq e^{-\lambda m t} |e(0)| + B \int_0^t e^{-\lambda m (t-\tau)}\, d\tau \\ &= e^{-\lambda m t} |e(0)| + \frac{B}{\lambda m} \left(1 - e^{-\lambda m t}\right). \end{aligned}$$

The bound separates two contributions: one from the initial deviation, which decays, and one from the drift b that the verifier induces. If $b \equiv 0$, the deviation decays exponentially. If the drift persists, the bound allows a nonzero deviation to persist as well.

This is a scalar estimate of the form used in input-to-state stability analysis (Khalil & Grizzle, 2002). Here, the conclusion concerns e under the stated bounds. It guarantees neither stability of all policy parameters nor exact preservation of the ratio.

LIMITATION: RESTORATION CAN INCREASE ERRORS. Consider one prompt with two responses, y_1 correct and y_2 incorrect, both accepted. Let

$$p_G(\theta) = \frac{1}{1 + e^{-\theta}}, \qquad p_H(\theta) = 1 - p_G(\theta).$$

For a reference parameter θ_{ref}, we have $J_R \equiv 1$ and $e = \theta - \theta_{\mathrm{ref}}$. Thus,

$$\dot{\theta} = -\lambda e, \qquad \dot{p}_G = -\lambda e \, p_G p_H, \qquad \dot{p}_H = \lambda e \, p_G p_H.$$

If $\theta(0) > \theta_{\mathrm{ref}}$, then $e(t) = e(0) e^{-\lambda t} > 0$ at every finite time. Restoration therefore decreases correctness and increases hack probability while reducing the deviation from the reference ratio.

The current policy initially favors the correct response more strongly than the reference does, so restoring the reference reverses that improvement. If initialized at the reference instead, the flow remains stationary and does not strictly reduce hack probability.

Common limit of the three penalties. We hold the reference policies, penalty weights, and selected response pairs fixed across correctness assignments and give the penalties no additional correctness information. At the same policy, each penalty and its gradient then agree under every assignment in $\mathcal{C}_R$. The argument about selective control above, which shows that the records coincide, therefore gives the same controlled trajectory under the compared assignments.

Under the assumptions of Proposition 4.3, none of these penalties can guarantee $\dot{p}_H < 0$ and $\dot{p}_G \geq 0$ along the trajectory under every assignment in $\mathcal{C}_R$ whenever $p_H > 0$. A penalty may control the departure from a reference, the sensitivity of the reward, or the relative probabilities of paired

responses without providing this uniform guarantee of selective control, although it may still succeed on particular tasks.

Additional correctness information or justified structural assumptions can restrict $\mathcal{C}_R$ and exclude indistinguishable assignments that require incompatible corrections. Excluding these assignments removes the obstruction exhibited here, even without identifying every hack. Removing the obstruction does not by itself give a guarantee, however: a guarantee of selective control also requires showing that the available updates satisfy $\dot{p}_H < 0$ and $\dot{p}_G \geq 0$ throughout the trajectory.

C.4 Technical Details for Section 5

C.4.1 Projected Correction and Its Immediate Effects

Write $h_A = \nabla_\theta p_{H_A}$ and $h = \nabla_\theta p_H$. Since $p = p_H + p_G$, we have $\nabla_\theta p_G = g_R - h$. We first derive the effects of the projected correction without assuming full audit coverage.

Step 1: Preserve instantaneous acceptance growth. Define

$$P_\perp = \begin{cases} I_d - \dfrac{g_R g_R^\top}{\|g_R\|^2}, & g_R \neq 0, \\ I_d, & g_R = 0. \end{cases}$$

This orthogonal projector satisfies $P_\perp^\top = P_\perp$, $P_\perp^2 = P_\perp$, and $g_R^\top P_\perp = 0$. For the correction $u = -\lambda P_\perp h_A$, with $\lambda > 0$, we therefore have

$$g_R^\top u = 0.$$

Along the corrected flow $\dot\theta = g_R + u$, the chain rule gives

$$\dot p = g_R^\top (g_R + u) = \|g_R\|^2.$$

Thus the correction preserves the instantaneous acceptance growth produced by the verifier gradient at the current policy.

Step 2: Oppose growth of audited hacks. Symmetry and idempotence of the projector give

$$h_A^\top u = -\lambda h_A^\top P_\perp h_A = -\lambda \|P_\perp h_A\|^2.$$

Consequently,

$$\dot p_{H_A} = h_A^\top g_R - \lambda \|P_\perp h_A\|^2.$$

The correction contributes a strictly negative term when $P_\perp h_A \neq 0$. The full rate is negative only when this contribution outweighs any positive growth induced by the verifier gradient.

Step 3: Relate the correction to the full population. For the full hack probability, the correction contributes

$$h^\top u = -\lambda (P_\perp h)^\top (P_\perp h_A).$$

Because $g_R^\top u = 0$, its contribution to correct responses is equal and opposite:

$$\nabla_\theta p_G^\top u = (g_R - h)^\top u = -h^\top u.$$

Thus, when the two projected gradients have positive inner product, the correction opposes hack growth and contributes equally to correct-response growth. These statements concern the correction's contribution. Selectivity of the full update also depends on g_R.

If $P_\perp h = 0$, every correction orthogonal to g_R has zero instantaneous effect on p_H. The projection therefore requires a component of the hack gradient orthogonal to the reward gradient. The theorem below shows how full coverage and sufficient corrective strength turn these local identities into selective control and a bound along the evolving policy.

C.4.2 Proof of Theorem 5.1

The proof has four steps. Full audit coverage identifies the gradient $h = \nabla p_H$. Projection removes the component of h along g_R, so the correction does not slow the growth of acceptance. A strong enough correction then gives $\dot p_H < 0$ and $\dot p_G \geq 0$. Finally, integrating these rates over $[0, T]$ bounds the hacked share.

Step 1: Full coverage identifies the gradient of p_H. Under (A2), unaudited hacks have zero probability along the policy trajectory:

$$p_H - p_{H_A} = 0.$$

As a function of θ, this difference is nonnegative everywhere, because audited hacks form a subset of all hacks. At every point of the trajectory, it therefore attains its minimum value of zero, and its gradient vanishes:

$$\nabla p_{H_A} = \nabla p_H.$$

Thus the audited population gradient coincides with the full hack gradient along the trajectory. The correction becomes $u = -\lambda P_\perp h$, where $P_\perp$ projects onto the orthogonal complement of g_R, and we write $S = \|P_\perp h\|^2$ for the squared norm of the projected gradient.

Step 2: Derive the corrected dynamics. The corrected flow is $\dot\theta = g_R + u = g_R - \lambda P_\perp h$. Because $P_\perp$ is an orthogonal projection onto the complement of g_R, we have $g_R^\top P_\perp h = 0$ and $h^\top P_\perp h = \|P_\perp h\|^2 = S$ (Appendix C.4.1). Differentiating along the flow therefore gives

$$\dot p = \|g_R\|^2, \qquad \dot p_H = h^\top g_R - \lambda S, \qquad \dot p_G = \|g_R\|^2 - \dot p_H.$$

At the current policy, the correction preserves the instantaneous acceptance growth, lowers the rate $\dot p_H$ by λS, and raises the rate $\dot p_G$ by the same amount.

To compare the growth of the two groups, recall $z = \log(p_H/p_G)$. Without the correction, z changes at the rate

$$b = (\bar s_H - \bar s_G)^\top g_R,$$

which we call the pressure that verifier rewards exert on the hacked share. Taking the logarithmic derivative gives

$$\begin{aligned}\dot z &= \frac{\dot p_H}{p_H} - \frac{\dot p_G}{p_G} \\ &= b - \lambda S\left(\frac{1}{p_H} + \frac{1}{p_G}\right) \\ &= b - \frac{\lambda S}{p\,q(1-q)}.\end{aligned}$$

Since $\dot q = q(1-q)\dot z$, we also obtain

$$\dot q = q(1-q)b - \frac{\lambda S}{p}.$$

These expressions separate the verifier's growth pressure from the opposing effect of the correction.

Step 3: Identify when the update is selective. The correction lowers $\dot p_H$ by λS and raises $\dot p_G$ by the same amount, so

$$\dot z = \frac{\dot p_H}{p_H} - \frac{\dot p_G}{p_G} = b - \lambda S\Big(\frac{1}{p_H} + \frac{1}{p_G}\Big).$$

If the correction term exceeds b, then $\dot z < 0$, and hence $\dot q < 0$, since z increases with q. Differentiating $p_G = p(1-q)$ and using $\dot p = \|g_R\|^2$ then gives

$$\dot p_G = (1-q)\|g_R\|^2 - p\dot q > 0.$$

Because acceptance never decreases under the correction, a falling hacked share implies rising correctness.

A falling hacked share does not require p_H itself to fall, since acceptance may still grow. For this stronger conclusion, the correction must overcome the verifier's contribution to $\dot p_H$:

$$\lambda S > h^\top g_R.$$

Then $\dot p_H < 0$ and $\dot p_G = \|g_R\|^2 - \dot p_H > 0$. Because p_H falls while p_G rises, the log ratio z decreases as well, and part (i) follows.

Step 4: Bound the hacked share during training. Because q is the logistic function of z, the rate from Step 3 gives

$$\dot{q} = q(1-q)\,\dot{z} = q(1-q)b - \frac{\lambda S}{p},$$

where the second term uses $p_H = pq$ and $p_G = p(1-q)$. Assumption (A3) bounds the correction from below, $S/p \geq \kappa q$, and the pressure from verifier rewards is bounded above by $b \leq D$ with $D \geq 0$. Since $q(1-q) \leq 1/4$, these bounds give

$$\dot{q} \leq \frac{D}{4} - \lambda\kappa q.$$

The first term caps the pressure from verifier rewards, and the second is a correction proportional to the current hacked share.

Rearranging gives $\dot{q} + \lambda\kappa q \leq D/4$, and multiplying by the integrating factor $e^{\lambda\kappa t}$ gives

$$\frac{d}{dt}\left(e^{\lambda\kappa t} q(t)\right) \leq \frac{D}{4} e^{\lambda\kappa t}.$$

Integrating from 0 to t and rearranging yields

$$q(t) \leq e^{-\lambda\kappa t} q(0) + \frac{D}{4\lambda\kappa}\left(1 - e^{-\lambda\kappa t}\right), \qquad t \in [0, T].$$

The initial hacked share contributes a term that decays at rate $\lambda\kappa$, while the pressure from verifier rewards contributes a term that rises toward $D/(4\lambda\kappa)$. Part (ii) follows, which completes the proof of the theorem. □

C.4.3 Estimating the gradient of p_H from audits

We fix the current policy and assume that audits reveal exact correctness labels. The estimator weights each observed hack by the inverse of its audit probability, which compensates for hacks that are generated but not audited.

Step 1: Express the gradient as an expectation. With the policy score $s_\theta(x, y) = \nabla_\theta \log \pi_\theta(y \mid x)$, and since $R(1-c)$ indicates a hack, differentiating the probability of hacks gives

$$h := \nabla_\theta p_H = \mathbb{E}_\theta\left[R(X,Y)(1 - c(X,Y))s_\theta(X,Y)\right].$$

Each hack contributes its policy score, and all other responses contribute zero.

Step 2: Weight the observed contributions. We draw n independent pairs with $X_i \sim \mathcal{D}$ and $Y_i \sim \pi_\theta(\cdot \mid X_i)$. We audit each accepted response with known probability $\rho_i = \rho(X_i, Y_i) > 0$ and never audit rejected responses. The indicator I_i records whether an audit occurs, and when $I_i = 1$, the audit reveals $C_i = c(X_i, Y_i)$. The estimator is

$$\xi_i = \begin{cases} \dfrac{1 - C_i}{\rho_i} s_\theta(X_i, Y_i), & I_i = 1, \\ 0, & I_i = 0, \end{cases} \qquad \widehat{h} = \frac{1}{n}\sum_{i=1}^{n} \xi_i.$$

An audited hack receives weight $1/\rho_i$, while audited correct responses and unaudited responses contribute zero. The average divides by all n generated responses, including those that were not audited.

Step 3: Show why the weighting works. For an accepted response, an audit occurs with probability ρ_i, and the weight $1/\rho_i$ cancels this probability in expectation. For a rejected response, both sides are zero. Hence

$$\mathbb{E}[\xi_i \mid X_i, Y_i] = R(X_i, Y_i)(1 - c(X_i, Y_i))s_\theta(X_i, Y_i),$$

and averaging over the generated responses gives

$$\mathbb{E}[\widehat{h}] = h, \tag{6}$$

where the expectation covers both response generation and audit selection. Audits of only some responses therefore give an unbiased estimate of h, provided that every accepted response has a positive audit probability.

Consequence for the projected correction. We keep g_R and $P_\perp$ exact and fix $\lambda > 0$ before drawing the batch. Every realized correction $\widehat{u} = -\lambda P_\perp \widehat{h}$ then satisfies $g_R^\top \widehat{u} = 0$, so it preserves the instantaneous acceptance growth at the fixed policy. Moreover,

$$\begin{aligned}\mathbb{E}[h^\top(g_R + \widehat{u})] &= h^\top g_R - \lambda h^\top P_\perp \mathbb{E}[\widehat{h}] \\ &= h^\top g_R - \lambda \|P_\perp h\|^2.\end{aligned}$$

At the fixed policy, the correction from a batch therefore changes p_H at the same expected rate as the population correction. A single batch can still produce a direction that does not decrease p_H.

Under the stated assumptions, the estimator is unbiased. Small audit probabilities, however, produce large weights $1/\rho_i$ and can increase sampling variability. The argument does not cover accepted responses with zero audit probability or audits with systematic label errors. Appendix A.2 discusses the practical consequences of both cases.

D EXPERIMENTAL DETAILS

D.1 THE GAUSSIAN CONTEXTUAL BANDIT EXPERIMENT

Motivation. To test our theoretical predictions, we use a controlled contextual bandit in which we know the correctness of every response and can compute population gradients exactly.

Setup.

POLICY. The policy assigns each response a logit using shared parameters $\theta \in \mathbb{R}^4$, fixed features $\phi(x, y) \in \mathbb{R}^4$, and a fixed offset $a(x, y)$:

$$\pi_\theta(y \mid x) = \frac{\exp\big(\theta^\top \phi(x, y) + a(x, y)\big)}{\sum_{y' \in \mathcal{Y}_x} \exp\big(\theta^\top \phi(x, y') + a(x, y')\big)}.$$

Training changes only θ. Sharing parameters allows updates from different prompts to interact.

INITIALIZATION. We initialize $\theta(0) = 0$ and use offsets to set the initial acceptance $p(0)$ and hacked share $q(0)$:

$$a(x, y) = \begin{cases} \log\big[p(0)(1 - q(0))/16\big], & y \in G_x, \\ \log\big[p(0)q(0)/16\big], & y \in H_x, \\ \log\big[(1 - p(0))/16\big], & y \in N_x. \end{cases}$$

This gives $p_G(0) = p(0)(1 - q(0))$, $p_H(0) = p(0)q(0)$, and $p_N(0) = 1 - p(0)$, with equal probabilities within each group. Changing the offsets varies these probabilities while keeping the features fixed.

Prompt and Group Distributions.

PROMPTS AND LABELS. We use eight equally weighted prompts, each with 16 responses in each of three groups. Correct responses G_x have $(R, c) = (1, 1)$, hacks H_x have $(R, c) = (1, 0)$, and rejected responses N_x have $(R, c) = (0, 0)$. Group membership and labels remain fixed during training.

ACCEPTED FEATURES. We construct features for hacks by centering Gaussian noise and adding the mean $(1, 1, 0, 0)^\top$. We obtain features for correct responses by negating the second coordinate. Writing $\phi_{S,x,j}$ for response j in group S_x, we draw

$$\varepsilon_{x,j} \overset{\text{iid}}{\sim} \mathcal{N}(0, 0.25^2 I_4), \qquad j = 1, \ldots, 16,$$

and set

$$\phi_{H,x,j} = (1, 1, 0, 0)^\top + \varepsilon_{x,j} - \frac{1}{16} \sum_{k=1}^{16} \varepsilon_{x,k},$$

$$\phi_{G,x,j} = \mathrm{diag}(1, -1, 1, 1)\phi_{H,x,j}.$$

The empirical means are exactly $(1, 1, 0, 0)^\top$ for hacks and $(1, -1, 0, 0)^\top$ for correct responses in every prompt.

REJECTED FEATURES. We draw eight Gaussian vectors per prompt and append copies with the second coordinate negated:

$$\xi_{x,j} \overset{\text{iid}}{\sim} \mathcal{N}(0, 0.25^2 I_4), \qquad v_{x,j} = \xi_{x,j}, \qquad v_{x,j+8} = \mathrm{diag}(1, -1, 1, 1)\xi_{x,j},$$

for $j = 1, \ldots, 8$. We center these vectors and add the desired mean $\mu_N = (0, \mu_{N,2}, 0, 0)^\top$:

$$\phi_{N,x,j} = v_{x,j} - \frac{1}{16} \sum_{k=1}^{16} v_{x,k} + \mu_N, \qquad j = 1, \ldots, 16.$$

The empirical mean is therefore exactly μ_N. The experiments use $\mu_{N,2} \in \{-1.5, -0.5, 0, 0.5\}$.

We draw the raw noise independently across prompts and between the accepted and rejected groups. Within each seed, we reuse these draws across initializations, rejected means, and learning rates.

Objective.

TRAINING OBJECTIVE. We maximize verifier acceptance:

$$J_R(\theta) = \frac{1}{8}\sum_{x=1}^{8}\sum_{y\in\mathcal{Y}_x}\pi_\theta(y \mid x)R(x,y) = p(\theta).$$

Correct responses and hacks receive the same reward. Correctness labels define the controlled initializations and evaluation metrics; they do not enter the training objective.

TRAINING DYNAMICS. We evaluate the objective and gradient by summing over every prompt and response. We numerically integrate the verifier flow (1),

$$\dot{\theta} = g_R = \nabla_\theta J_R,$$

using DOP853 until $T = 50$. For the learning rate ablation, we instead take discrete updates:

$$\theta_{k+1} = \theta_k + \eta g_R(\theta_k).$$

Both procedures use exact population gradients.

Metrics.

ACCEPTANCE AND CORRECTNESS. We compute each group's probability over all prompts:

$$p_S(\theta) = \frac{1}{8}\sum_{x=1}^{8}\sum_{y\in S_x}\pi_\theta(y \mid x), \qquad S \in \{G, H, N\}.$$

We report acceptance $p = p_G + p_H$, hacked share $q = p_H/p$, and correctness p_G. We form q after aggregating over prompts. The proxy and true objectives are $J_R = p$ and $J_C = p_G$.

LOG ODDS AND GROUP SCORES. We track $z = \log(p_H/p_G)$ and compute the group scores $\bar{s}_S = \nabla_\theta \log p_S$ exactly:

$$\bar{s}_S = \frac{1}{8p_S}\sum_{x=1}^{8}\sum_{y\in S_x}\pi_\theta(y \mid x)\left[\phi(x,y) - \sum_{y'\in\mathcal{Y}_x}\pi_\theta(y' \mid x)\phi(x,y')\right].$$

We subtract each prompt's mean feature before aggregating. Although the features remain fixed, the group scores change as the policy re-weights responses.

HACK BIAS AND LEAKAGE. We compute the two contributions in (3):

$$\dot{z} = p(1-p)\left[\underbrace{(\bar{s}_H - \bar{s}_G)^\top(\bar{s}_G - \bar{s}_N)}_{\text{leakage}} + \underbrace{q\|\bar{s}_H - \bar{s}_G\|^2}_{\text{hack bias}}\right].$$

Their sum determines the sign of $\dot{z}$ and hence the direction of change in the hacked share. We evaluate the full expression within each seed before averaging.

GROWTH AND REWARD HACKING. We compute the rates of acceptance, hacked share, and correctness:

$$\dot{p} = \|g_R\|^2, \qquad \dot{q} = q(1-q)\dot{z}, \qquad \dot{p}_G = (1-q)\dot{p} - p\dot{q}.$$

We identify declining correctness using $p\dot{q} > (1-q)\dot{p}$ and locate stationary correctness where $\dot{p}_G = 0$. These quantities distinguish growth of the hacked share from a decrease in the probability of correct responses.

VARIATION ACROSS SEEDS. The appendix experiments use ten independent Gaussian feature draws and report means and one sample standard deviation. Each draw defines deterministic training. For the comparison of correctness against acceptance, we interpolate each trajectory onto acceptance values reached by every seed and geometry before computing these statistics.

D.1.1 When Hacking Grows

Controlled comparisons. We fix $p(0) = 2/3$. In the main paper, panel (a) tracks p, q, and p_G with $q(0) = 0.3$ and $\mu_{N,2} = -0.5$. Panel (b) compares J_C against J_R for $q(0) \in \{0.1, 0.3, 0.5\}$ using the same features.

The centered construction lets us vary initial leakage and hack bias separately. At initialization,

$$(\bar{s}_H - \bar{s}_G)^\top (\bar{s}_G - \bar{s}_N) = -2(1 + \mu_{N,2}), \qquad q(0)\|\bar{s}_H - \bar{s}_G\|^2 = 4q(0).$$

Changing the rejected mean changes initial leakage. Changing the initial hacked share changes initial hack bias while preserving the differences between group scores. The following experiments examine how these contributions evolve during training.

Results.

Experiment I: hack bias and correctness-to-hack leakage. Starting from initial acceptance $p(0) = 2/3$, hack share $q(0) = 0.3$, and mean rejected strength $\mu_{N,2} = -0.5$, we track hack bias, leakage from correctness to hacking, and their sum. For the same trajectories, we plot $\dot{z}(t)$ and compare it with central time differences of z. Together, these experiments relate the competing contributions to the growth rate of hacking through the factor $p(\theta)(1 - p(\theta))$ in (3).

Figure 3(a) shows that hack bias increases while leakage from correctness to hacking remains near -1. This value matches initialization, where the centered feature means give

$$(\bar{s}_H - \bar{s}_G)^\top (\bar{s}_G - \bar{s}_N) = -2(1 + \mu_{N,2}) = -1.$$

Leakage stays near this value, consistent with the similar Gaussian spreads across groups: policy reweighting shifts their mean features similarly, leaving the contrasts between groups approximately unchanged. Negative leakage opposes growth of the hacked share, but hack bias outweighs it throughout the plotted interval. Their sum therefore stays positive, and Proposition 3.2 predicts $\dot{z} > 0$ and an increasing hacked share. The predicted rate agrees with central time differences in Figure 3 (b). Thus, negative leakage can oppose hacking without overcoming the reinforcement from hack bias.

Ablation I: varying the mean rejected strength. We keep $p(0) = 2/3$ and $q(0) = 0.3$ fixed and vary $\mu_{N,2} \in \{-1.5, -0.5, 0.5\}$, holding the accepted features and underlying Gaussian draws fixed. Since initial leakage equals $-2(1 + \mu_{N,2})$, these values give initial leakage of $+1$, -1, and -3: positive, negative but weaker than the initial hack bias, and negative and stronger than it. For each condition, we plot correctness $J_C = p_G$ against verifier reward $J_R = p$. Plotting against reward rather than time compares the conditions at the same acceptance level, which isolates how rejected responses affect correctness as verifier reward rises.

Figure 3 (c) shows how the mean rejected strength changes the relation between verifier reward and correctness. For $\mu_{N,2} = -1.5$, where initial leakage is positive, correctness declines after a brief initial increase, so reward hacking emerges early. For $\mu_{N,2} = -0.5$, where negative leakage is weaker than hack bias, acceptance and correctness first improve together, but correctness later decreases despite further gains in acceptance. For $\mu_{N,2} = 0.5$, where negative leakage is stronger, both quantities increase throughout the plotted range, with no reward hacking evident in the mean curve. These declines illustrate Proposition 3.1: reward hacking occurs when $p\dot{q} > (1 - q)\dot{p}$, that is, when the shift toward hacked responses outweighs the correctness gained from rising acceptance. Thus, even with identical initial acceptance and hacked share, the geometry of the rejected responses can change whether and when reward hacking emerges.

Ablation II: learning rate in exact gradient ascent. To test how closely gradient flow approximates discrete training, we keep $p(0) = 2/3$ and $q(0) = 1/2$ fixed and vary the learning rate η, holding the features and initialization fixed within each seed and geometry. We compare iterate k of each discrete trajectory with the reference flow at the matched time $t = k\eta$. For each seed, we take the maximum discrepancy in z, p, q, and p_G over times, and we then average these maxima across seeds.

Figure 3 (d) shows that discrepancies in z, p, q, and p_G increase with the learning rate, reaching the order of 10^{-2} at $\eta = 0.4$. Because the updates use exact population gradients, these discrepancies

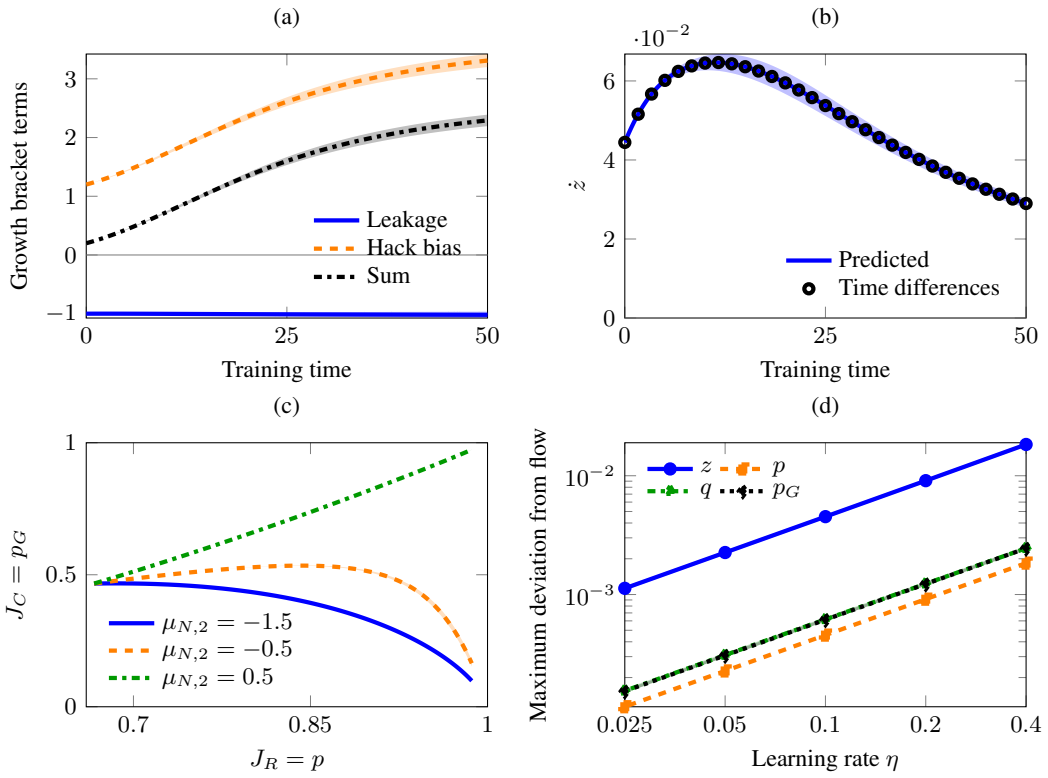

Figure 3: Growth of hacking in the Gaussian contextual bandit. (a) Hack bias outweighs negative correctness-to-hack leakage, keeping their sum positive. (b) The resulting growth rate $\dot{z} > 0$ of the log odds agrees with central time differences along the same trajectories. (c) Correctness against acceptance for three values of the mean rejected strength $\mu_{N,2}$. Changing the rejected features changes whether and when correctness declines as verifier reward rises. (d) Discrete gradient ascent approaches the reference flow as the learning rate decreases. We compare iterate k with the flow at $t = k\eta$ and report, for each variable and seed, the maximum absolute discrepancy over evaluated times. Curves show means across ten independent Gaussian feature draws, and shading shows $\pm$ one sample standard deviation. Each draw defines deterministic dynamics with exact population gradients.

reflect only the effect of taking finite steps. As the learning rate decreases, the discrepancies shrink, which supports using gradient flow to approximate discrete training over the tested interval.

Takeaway. The verifier rewards every accepted response, whether correct or hacked. Which group benefits therefore depends on the feature geometry, including the rejected responses that shape the direction in which acceptance increases. Along this direction, hack bias can overcome negative leakage and shift the accepted population toward hacked responses. Correctness falls once this shift outweighs the correctness gained from rising acceptance. Because discrete gradient ascent with small learning rates closely tracks gradient flow, the flow offers a reliable way to study this competition.

Hyperparameters. Table 1 lists the experimental settings.

Table 1: Settings for the Gaussian contextual-bandit growth experiments.

Parameter	Value
Number of prompts	8
Responses per group per prompt	16
Feature dimension	4
Gaussian noise standard deviation	0.25
Initial parameters	$\theta(0) = 0$
Independent feature seeds	10 (seeds $0, \ldots, 9$)
Initial acceptance	$p(0) = 2/3$
Initial hacked share, Experiment I and Ablation I	$q(0) = 0.3$
Initial hacked share, Ablation II	$q(0) = 1/2$
Rejected mean, Experiment I	$\mu_{N,2} = -0.5$
Rejected means, Ablation I	$\mu_{N,2} \in \{-1.5, -0.5, 0.5\}$
Rejected means, Ablation II	$\mu_{N,2} \in \{-0.5, 0, 0.5\}$
Flow integrator	DOP853
Maximum integration step	0.25
Training horizon	$T = 50$
Recorded flow times	$0, 0.1, \ldots, 50$
Gradient-ascent learning rates η	$\{0.4, 0.2, 0.1, 0.05, 0.025\}$

D.1.2 Limits of Verifier Feedback

We study the limits of selective control from verifier feedback (Section 4.4). We evaluate gradient regularization under two compatible correctness assignments along the same sequence of policies. The assignments exchange correct responses and hacks, so reducing hack probability under one reduces correctness under the other. We vary the regularization weight to measure the fraction of recorded times when control is selective under each assignment.

Additional experimental settings. We use the policy and feature construction from Appendix D.1 to study control from verifier information, as described in Section 4.4.

Fixed settings. We set $p(0) = 2/3$, $q(0) = 1/2$, $\mu_N = (0, -0.5, 0, 0)^\top$, and $\theta(0) = 0$. The offsets are $a(x, y) = \log(1/48)$, giving $p_G(0) = p_H(0) = p_N(0) = 1/3$. We use ten independent feature draws and reuse each draw across regularization weights and correctness assignments. The verifier and initial policy remain fixed across comparisons.

Correctness assignments. We keep the groups G, H, N fixed and evaluate two assignments: $c_1 = \mathbf{1}_G$ and $c_2 = \mathbf{1}_H = R - c_1$. Under c_1, correct responses belong to G and hacks belong to H. Under c_2, these roles reverse:

$$p_{G,c_1} = p_G, \qquad p_{H,c_1} = p_H, \qquad p_{G,c_2} = p_H, \qquad p_{H,c_2} = p_G.$$

Both assignments use the same verifier. Correctness labels enter only the evaluation.

For panel Figure 1, Panel (c) in the main paper, we also evaluate $c = R$, under which $p_{G,c} = p$ and $p_{H,c} = 0$, along the same verifier flow used for c_1.

Exact derivatives. Using the group scores computed above, we obtain

$$\nabla p_S = p_S \bar{s}_S, \qquad S \in \{G, H, N\}, \qquad g_R = \nabla p_G + \nabla p_H.$$

We compute the reward gradient by summing over all prompts and responses.

Controller. Gradient regularization penalizes the squared norm of the reward gradient:

$$F_{\mathrm{GR}}(\theta) = J_R(\theta) - \gamma \|g_R(\theta)\|^2, \qquad B_R(\theta) = \nabla_\theta^2 J_R(\theta).$$

We follow its gradient by adding a correction to the verifier flow:

$$u(t) = -2\gamma B_R(\theta(t)) g_R(\theta(t)), \qquad \dot{\theta} = g_R + u(t) = \nabla_\theta F_{\mathrm{GR}}.$$

We recompute the derivatives at the current policy and keep γ fixed within each run. The main comparison uses $\gamma = 16$; the ablation uses $\gamma \in \{0, 1, 4, 16\}$. Setting $\gamma = 0$ recovers the verifier flow. Because the update uses no correctness labels, both assignments give the same policy $\pi_{\theta(t)}$ at every time.

NUMERICAL INTEGRATION. We integrate to $T = 1$ using classical fourth order Runge–Kutta with step 0.01 and record the policy at every step. We check integration accuracy by repeating each run with step 0.005. Table 2 lists the numerical settings.

SELECTIVE CONTROL. We evaluate changes in hack probability and correctness using the full parameter update:

$$\dot{p}_{H,c} = \nabla p_{H,c}^{\top}(g_R + u), \qquad \dot{p}_{G,c} = \nabla p_{G,c}^{\top}(g_R + u).$$

For numerical classification, we require $\dot{p}_{H,c} < -\epsilon$ and $\dot{p}_{G,c} \geq -\epsilon$, with $\epsilon = 10^{-8}$. We verify the exchange of rates across assignments:

$$\dot{p}_{H,c_1} = \dot{p}_{G,c_2}, \qquad \dot{p}_{G,c_1} = \dot{p}_{H,c_2}.$$

For each seed, we compute the fraction of recorded times satisfying selectivity. We report means and sample standard deviations across seeds for both the rates and these fractions.

Results.

EXPERIMENT I: VERIFIER-ONLY CONTROL. We run gradient regularization with $\gamma = 16$ and evaluate the resulting trajectory under c_1 and c_2. The controller uses only verifier information, so changing the correctness assignment leaves the trajectory unchanged. We plot $\dot{p}_{G,c}$ and $\dot{p}_{H,c}$ and identify intervals where $\dot{p}_{H,c} < 0$ and $\dot{p}_{G,c} \geq 0$.

Figure 4(a) shows that the mean rate $\dot{p}_{G,c_1} = \dot{p}_{H,c_2}$ decreases but remains positive, while $\dot{p}_{H,c_1} = \dot{p}_{G,c_2}$ rises from negative to positive. Initially, the controller therefore reduces mean hack probability and increases mean correctness under c_1, with opposite effects under c_2. Later, both mean rates become positive, so neither assignment shows a reduction in mean hack probability. The exchange of rates in (5) prevents selective control under both assignments at the same time, illustrating Proposition 4.3.

ABLATION: GRADIENT REGULARIZATION WEIGHT. We vary $\gamma \in \{0, 1, 4, 16\}$ while keeping the features and initial policy fixed within each seed. For each weight, we evaluate both correctness assignments along the same sequence of policies. We compute the fraction of recorded times satisfying selective control under each assignment and report the mean and sample standard deviation across ten seeds. At every evaluated time, we also check that the policy update is never selective under both assignments.

Figure 4 (b) shows that the mean fraction of selective updates under c_1 is zero for $\gamma \in \{0, 1, 4\}$ and below 25% for $\gamma = 16$. Under c_2, the fraction is 100% for $\gamma = 0$, above 50% for $\gamma = 1$, and zero for $\gamma \in \{4, 16\}$. Without regularization, training increases the probability of H and decreases that of G throughout the recorded interval. This is selective under c_2, which labels H as correct and G as hacks, but has the opposite effect under c_1. Increasing regularization to $\gamma = 16$ introduces a period of selectivity under c_1 while eliminating selectivity under c_2. These results illustrate the information limit in Proposition 4.3: a controller using only RLVR observations cannot guarantee selective control across compatible correctness assignments.

Takeaway. Gradient regularization changes which accepted responses lose probability, but verifier observations do not reveal whether those responses are hacks or correct. The same reduction therefore removes hacks under one compatible assignment and correct responses under the other. Tuning the regularization weight changes which assignment benefits without resolving this ambiguity. Guaranteeing selective control requires additional information that distinguishes the correctness assignments.

Hyperparameters. Table 2 lists the additional and changed settings for this experiment. All other settings follow Table 1.

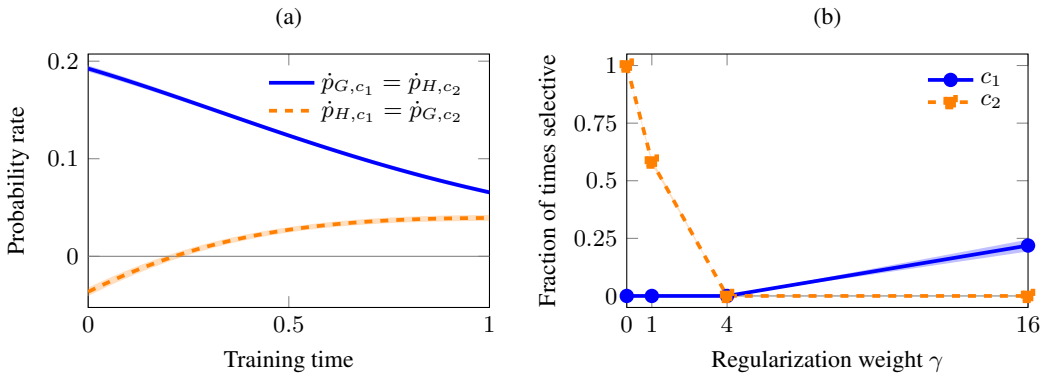

Figure 4: Control using only RLVR training observations in the Gaussian contextual bandit. (a) Rates of change in correctness and hack probability under gradient regularization with $\gamma = 16$. The assignments c_1, c_2 exchange these rates along the same sequence of policies. (b) Fraction of recorded times satisfying selective control for each regularization weight γ: $\dot{p}_{H,c} < -\epsilon$ and $\dot{p}_{G,c} \geq -\epsilon$, with $\epsilon = 10^{-8}$. We compute each fraction within a seed before averaging. Lines show means across ten independent Gaussian feature draws; shading shows $\pm$ one sample standard deviation.

Table 2: Additional settings for the verifier-only control experiments (Appendix D.1.2). The main experiment uses $\gamma = 16$; the ablation varies its weight. Both correctness assignments share the same controlled trajectory. All expectations are evaluated exactly. Other model settings follow Table 1.

Parameter	**Value**
Rejected-group second-coordinate mean	-0.5
Initial hacked share $q(0)$	$1/2$
Initial acceptance $p(0)$	$2/3$
Initial parameters	$\theta(0) = 0$
Independent feature seeds	10 (seeds $0, \ldots, 9$)
Main regularization weight γ	16
Regularization-weight sweep	$\{0,\ 1,\ 4,\ 16\}$
Flow integrator	Classical fourth-order Runge–Kutta
Training horizon	$T = 1$
Recorded times	$0,\ 0.01,\ \ldots,\ 1$
Tolerance for selectivity ϵ	10^{-8}

D.1.3 Selective Control with Correctness Feedback

We studyprojected audit corrections, examine the effects of audit coverage and projection error, and illustrate the ISS bound in Theorem 5.1.

Additional experimental settings.

Features and initialization. We use the policy and Gaussian construction from Appendix D.1, changing the accepted feature means to $(4, -1, 0, 0)^\top$ for G and $(4, 1, 0, 0)^\top$ for H. The rejected mean remains $(0, -0.5, 0, 0)^\top$. These means give $\nabla p_G(0)^\top \nabla p_H(0) = 29/324 > 0$, so audit correction without projection (raw) initially opposes correctness. We fix $c = c_1 = \mathbf{1}_G$, $p(0) = 2/3$, $q(0) = 1/2$, and $\theta(0) = 0$. Thus, every offset equals $\log(1/48)$ and $p_G(0) = p_H(0) = p_N(0) = 1/3$. We reuse features and initialization across methods within each seed.

Audits and exact gradients. A fixed audit set A_{aud} reveals correctness labels for selected accepted responses. Its audited hacks are $H_A = A_{\text{aud}} \cap H$. We compute their probability gradient

exactly:

$$\nabla p_{H_A} = \frac{1}{8} \sum_{x} \sum_{y:(x,y)\in H_A} \pi_\theta(y \mid x) \left[\phi(x,y) - \sum_{y'} \pi_\theta(y' \mid x) \phi(x,y') \right].$$

We use full coverage except in the coverage ablation. Under full coverage, $p_{H_A} = p_H$ and their gradients coincide. Labels outside the audit set enter only the evaluation.

CORRECTIONS. Each method follows $\dot{\theta} = g_R + u$, with

$$u = \begin{cases} 0, & \text{verifier flow,} \\ -2\gamma B_R g_R, & \text{gradient regularization,} \\ -\lambda \nabla p_{H_A}, & \text{raw audit correction,} \\ -\lambda P_\perp \nabla p_{H_A}, & \text{projected audit correction.} \end{cases}$$

Here $B_R = \nabla^2 J_R$ and $P_\perp = I - g_R g_R^\top / \|g_R\|^2$ when $g_R \neq 0$, with $P_\perp = I$ otherwise. We fix $\lambda = 6$ and $\gamma = 16$ and recompute derivatives at the current policy.

Additional metrics.

SELECTIVE CONTROL. We retain p_G, p_H, p, and q from Appendix D.1. We evaluate each method using the full update:

$$\dot{p}_H = \nabla p_H^\top (g_R + u), \qquad \dot{p}_G = \nabla p_G^\top (g_R + u).$$

For numerical classification, we require $\dot{p}_H < -\epsilon$ and $\dot{p}_G \geq -\epsilon$, with $\epsilon = 10^{-8}$. In the coverage ablation, we also record $(P_\perp \nabla p_H)^\top (P_\perp \nabla p_{H_A})$. A positive value means that the correction opposes growth of total hack probability.

PROJECTION ERROR. We construct the correction $\widehat{u}$ using an estimated reward gradient $\widehat{g}_R$ inside the projector. The verifier direction g_R and hack gradient ∇p_H remain exact. We measure the resulting error in acceptance growth:

$$\left| \dot{p} - \|g_R\|^2 \right| = \left| g_R^\top \widehat{u} \right| = \left| (g_R - \widehat{g}_R)^\top \widehat{u} \right|.$$

NUMERICAL ISS ENVELOPE. For each seed, we estimate D from the maximum of $(\bar{s}_H - \bar{s}_G)^\top g_R$ and zero along the trajectory. We estimate κ from the minimum of $\|P_\perp \nabla p_H\|^2 / p_H$. We refine the integration and increase the evaluation grid from 1,001 to 2,001 times. We insert the constants D and κ into Theorem 5.1 and compare its envelope with $q(t)$. These numerical estimates do not certify the bound between evaluated times.

Results

EXPERIMENT I: SELECTIVE CONTROL. Under $c = c_1$ and full audit coverage, we compare verifier flow, gradient regularization, raw audit correction, and projected audit correction. We compute exact gradients and keep the features and initial policy fixed across methods. Both audit corrections use the same fixed gain λ. We plot correctness p_G against hack probability p_H and identify intervals of selective control, where $\dot{p}_H < 0$ and $\dot{p}_G \geq 0$.

Figure 5 (a) shows that projected audit correction increases correctness p_G and decreases hack probability p_H throughout the recorded interval. Raw audit correction initially decreases both probabilities. Later, correctness increases while hack probability remains nearly constant. Both verifier flow and gradient regularization increase hack probability. These mean trajectories show that projection enables sustained hack reduction without the initial loss of correctness observed under raw audit correction.

EXPERIMENT II: ISS BOUND. We apply projected audit correction with full coverage and constant gain and compare $q(t)$ with the envelope in Theorem 5.1. For each feature seed, we estimate D and κ along the computed trajectory, then check these estimates using a finer time grid and tighter

integration tolerances. We construct the envelope separately for each seed and record its gap from $q(t)$. This comparison illustrates the bound numerically over the tested interval.

Figure 5 (b) shows that the hacked share $q(t)$ decreases throughout the recorded interval. It equals the ISS envelope at initialization and remains below it afterward, providing a numerical illustration of Theorem 5.1 along the tested trajectories.

ABLATION I: AUDIT COVERAGE. We vary the fraction of candidates audited in each accepted group and prompt over $\{0, 1/8, 1/4, 1/2, 3/4, 1\}$. We use the known groups to construct nested audit sets and keep their candidate indices fixed across seeds and throughout training. The controller receives correctness labels only for audited responses. For each fraction, we apply projected audit correction with $\lambda = 6$ until $T = 1$, using the exact gradient of the audited hack probability p_{H_A}. We compare total hack probability $p_H(T)$ across initial coverage levels and record the alignment between the projected gradients of p_H and p_{H_A}.

Figure 5 (c) shows that increasing audit coverage lowers the final hack probability. Without audits, the correction vanishes and verifier flow increases p_H. With sufficient coverage, the correction reduces p_H below its initial value. Although the correction uses only audited responses, we evaluate whether it reduces the total probability of hacks.

ABLATION II: PROJECTION ERROR. At $\theta = 0$ and full audit coverage, we estimate the reward gradient using batches of $n \in \{32, 128, 512, 2048\}$ independent prompts and responses. We average $R(x, y)\nabla_\theta \log \pi_\theta(y \mid x)$ within each batch and use this estimate only to construct the projection. The verifier direction g_R and hack gradient ∇p_H remain exact, and the correction gain is $\lambda = 6$. Without advancing the policy, we measure $|\dot{p} - \|g_R\|^2| = |g_R^\top \widehat{u}|$. For each batch size and feature seed, we average this error over $1{,}000$ independent batches, then report the mean and sample standard deviation across ten feature seeds.

Figure 5 (d) shows that larger batches reduce the mean error in preserving acceptance growth. The exact projection gives zero error. Estimating the projection introduces the discrepancy $\dot{p} - \|g_R\|^2 = (g_R - \widehat{g}_R)^\top \widehat{u}$. Thus, more samples reduce the error in the acceptance growth rate. Panel (a) checks whether the correction also reduces hack probability.

Takeaway. Audits reveal which accepted responses are wrong, but suppressing those responses can also suppress correct ones because they share policy parameters. Projection preserves the instantaneous growth rate of acceptance. When the corrected update reduces hack probability, correctness must therefore increase. The ISS bound describes how sustained correction limits the hacked share despite pressure toward hacks from verifier training. In these experiments, broader audit coverage improves hack reduction, while larger sample batches reduce projection error. Selective control reduces hack probability without reducing correctness. Projection preserves the instantaneous growth rate of acceptance, so any decrease in hack probability under the corrected flow must increase correctness.

Hyperparameters. Table 3 lists only added or changed settings for this experiment. All other applicable settings follow Table 1.

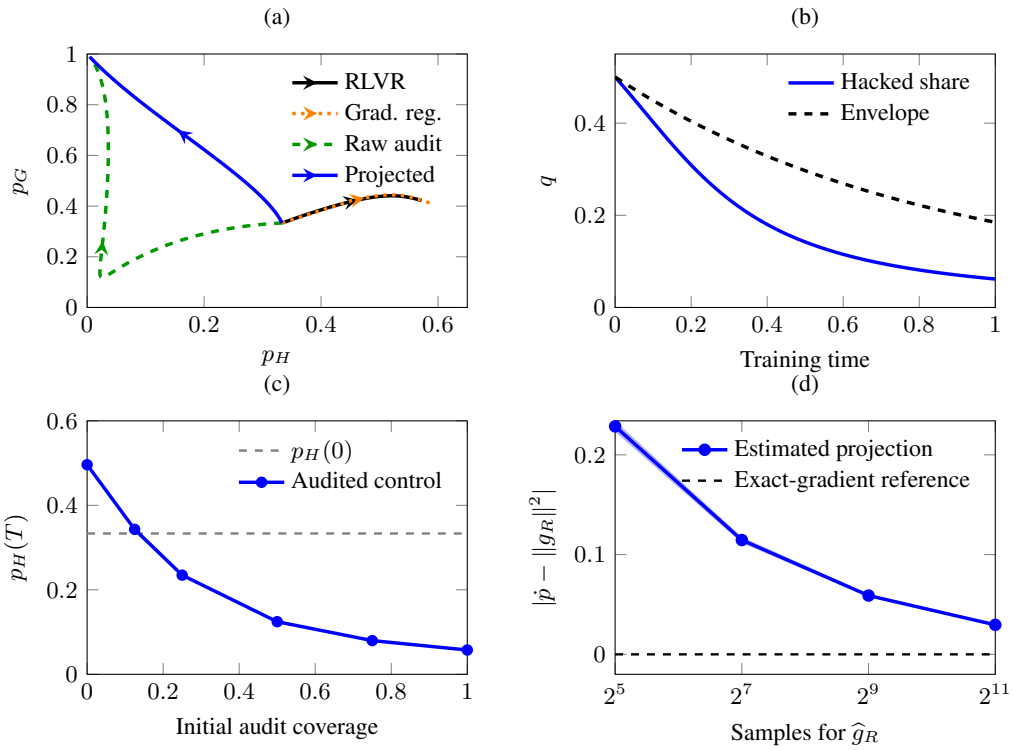

Figure 5: Selective control with correctness feedback in the Gaussian bandit. (a) Correctness p_G versus hack probability p_H; arrows indicate training direction. (b) Hacked share $q(t)$ and the numerical ISS envelope. (c) Final hack probability versus the initial fraction of hacks audited; the dashed line marks the initial hack probability. (d) Error $|\dot{p} - \|g_R\|^2|$ when estimating the reward gradient used for projection. Lines show means across ten feature seeds; shading shows $\pm$ one sample standard deviation. The envelope in (b) provides a numerical check and does not certify the bound between evaluated times.

Table 3: Additional and changed settings for selective control with correctness under c_1. Other model settings follow Table 1.

Parameter	**Value**
Accepted-group feature means	$(4, -1, 0, 0)$, $(4, 1, 0, 0)$
Rejected-group second-coordinate mean	-0.5
Initial hacked share $q(0)$	$1/2$
Initial acceptance $p(0)$	$2/3$
Initial parameters; warmup	$\theta(0) = 0$; none
Independent feature seeds	10 (seeds $0, \ldots, 9$)
Audit correction gain λ	6
Gradient-regularization weight γ	16
Audit labels	Exact
Initial audited fraction $p_{H_A}(0)/p_H(0)$	$\{0, 1/8, 1/4, 1/2, 3/4, 1\}$
Projection batch sizes n	$\{32, 128, 512, 2048\}$
Policy for projection-error ablation	$\theta = 0$
Flow integrator	DOP853
Horizon, control comparison	$T = 10$
Horizon, coverage and ISS	$T = 1$
Tolerance for selectivity ϵ	10^{-8}

D.2 The Neural Contextual Bandit Experiment

Motivation. We replace the log-linear policy in Appendix D.1 with a neural policy to examine reward hacking (Section 3) and the limits of verifier feedback (Section 4) when the policy learns its representation. We also examine how audit coverage and projection accuracy affect projected audit correction (Section 5).

Experimental setting. We use a shared MLP f_θ with four inputs, one hidden layer of 16 tanh units, and a scalar output. We train all weights and hidden biases and omit the output bias. The policy is

$$\pi_\theta(y \mid x) \propto \exp\big(a(x,y) + f_\theta(\phi(x,y)) - f_{\theta(0)}(\phi(x,y))\big)\,.$$

We freeze the subtracted network, so the offsets $a(x,y)$ (see Appendix D.1) determine the initial group probabilities.

Data. We use eight equally weighted prompts, 16 responses per group, and four-dimensional Gaussian features with noise standard deviation 0.25. We follow the centering and reflection construction in Appendix D.1, fixing the rejected group mean to $(0,-0.5,0,0)^\top$. The shared first-coordinate mean of the accepted groups is 1 in Experiments I–II and 4 in the ablations. Within each experiment, comparisons share the Gaussian draws and network initialization for each seed. The network receives response features without group labels or a separate prompt embedding.

Under the correctness assignment c_1, the groups G_x, H_x, N_x have labels $(R, c_1) = (1,1),(1,0),(0,0)$, respectively. Experiment II also evaluates $c = R$ and $c_2 = R - c_1$ while keeping these groups and the training trajectory fixed.

RLVR training. Experiments I–II follow verifier flow (1), $\dot\theta = g_R$, where $g_R = \nabla_\theta J_R$ and $J_R = p$. We compute probabilities and gradients by summing over all prompts and responses. We integrate the flow using DOP853, recomputing the gradients at each policy. The coverage ablation uses the same integrator for the corrected flow. Table 4 lists other numerical settings.

Audits. The coverage ablation uses fixed, nested audit sets A_{aud} under c_1. Each audit reveals the correctness of an accepted response. With $H_A = A_{\mathrm{aud}} \cap H$, we apply

$$\dot\theta = g_R - \lambda P_\perp \nabla_\theta p_{H_A}.$$

Both gradients are exact. The projector $P_\perp$ acts in the space of neural parameters, orthogonally to g_R, and equals I when $g_R = 0$.

The projection ablation evaluates corrections at the initial policy without advancing it. We estimate g_R from independent draws $x \sim \mathcal{D}$ and $y \sim \pi_\theta(\cdot \mid x)$ by averaging $R(x,y)\nabla_\theta \log \pi_\theta(y \mid x)$. Only the projector uses this estimate. The verifier direction g_R and the full hack gradient $\nabla_\theta p_H$ remain exact.

Metrics. We report acceptance p, correctness p_G, hack probability p_H, and hacked share $q = p_H/p$. We average group probabilities over prompts before forming this ratio. Experiment I compares $J_C = p_G$ with $J_R = p$ along training. In Experiment II, the subscript c identifies the correctness assignment defining $p_{G,c}$ and $p_{H,c}$. The coverage ablation reports $p_H(T)$ against initial coverage $p_{H_A}(0)/p_H(0)$. The projection ablation reports the deviation $|\dot p - \|g_R\|^2|$ from the acceptance growth rate preserved by the exact projector.

We report means and one sample standard deviation across ten independent seeds. For projection error, we first average absolute deviations over independent batches within each seed.

Results.

Experiment I: reward hacking. We study whether increasing acceptance can amplify hacks and reduce correctness with a neural policy. Under verifier flow, we vary the hack share $q(0) \in \{0.1, 0.3, 0.5\}$ while fixing initial acceptance $p(0) = 2/3$ and the response features. For $q(0) = 0.3$, we track acceptance p, hacked share q, and correctness p_G over time. We also compare the

proxy objective $J_R = p$ with the true objective $J_C = p_G$ to examine reward hacking as defined by (Skalse et al., 2022). RLVR training supplies the policies required by this definition: two times $t_1 < t_2$ exhibit reward hacking when $J_R(\theta(t_2)) > J_R(\theta(t_1))$ but $J_C(\theta(t_2)) < J_C(\theta(t_1))$. Thus, the comparison reveals whether improving verifier acceptance comes at the expense of correctness.

Figure 6 (a) shows that acceptance p and hacked share q increase together from $q(0) = 0.3$. As training progresses, correctness $p_G = p(1 - q)$ eventually decreases despite increasing acceptance. The shift toward hacks then outweighs the gain from accepting more responses (Section 3.1.)

Figure 6 (b) shows how the proxy objective $J_R = p$ and the true objective $J_C = p_G$ change along training. The mean curves show reward hacking throughout the recorded interval for $q(0) = 0.5$: acceptance increases while correctness decreases. For $q(0) = 0.3$, acceptance and correctness initially increase together. Reward hacking emerges later, when correctness decreases despite further gains in acceptance. For $q(0) = 0.1$, both quantities increase throughout the recorded interval, with no reward hacking evident in the mean curve. Hacks are present at initialization in all three cases, but reward hacking occurs when improving acceptance reduces correctness.

EXPERIMENT II: COMPATIBLE CORRECTNESS ASSIGNMENTS. We examine whether the same neural training record can support different conclusions about correctness. Starting from $q(0) = 0.3$, we evaluate each verifier trajectory under the correctness assignments: $c = R$, $c_1 = \mathbf{1}_G$, and $c_2 = R - c_1$. The features, offsets, initialization, and verifier are identical across these assignments. Their correctness probabilities are p, p_G, p_H, respectively, and their hack probabilities are $0, p_H, p_G$. Thus, comparing $c = R$ with c_1 tests whether the record reveals the presence of hacks, while comparing c_1 with c_2 tests whether it identifies which accepted responses are wrong (Section 4).

Figure 6 (c) shows identical acceptance p under three compatible correctness assignments. Correctness increases under $c = R$ and $c_2 = R - c_1$, whereas it eventually decreases under c_1. All three rules share the same training record L_t, so verifier observations cannot determine which correctness trajectory applies. This illustrates the limits of detection and identification in Propositions 4.1 and 4.2. Because c_1 and c_2 exchange correct responses and hacks, the comparison also illustrates why these observations cannot guarantee selective control under every compatible assignment (Proposition 4.3).

ABLATION I: AUDIT COVERAGE. We examine how much of the hack population a fixed *audit* set must expose for the correction to reduce total hack probability. We audit fractions $\{0, 1/8, 1/4, 1/2, 3/4, 1\}$ of the candidates in each accepted group and prompt. The sets are nested, with candidate indices fixed across seeds and throughout training. The evaluator's groups serve only to control coverage. Each condition uses the exact gradient ∇p_{H_A} and the same correction gain, without inverse probability weighting. We compare $p_H(T)$ across initial coverage levels and track how the audited and full hack gradients align. Zero coverage recovers verifier flow, while full coverage recovers the ideal projected correction. Partial coverage can also oppose hacking when the projected gradients align positively, as described in Appendix A.2.

Figure 6 (d) shows that the final hack probability $p_H(T)$ remains large when the initial audited fraction $p_{H_A}(0)/p_H(0)$ is small. Increasing this fraction reduces $p_H(T)$ to near zero in the tested setting. The correction uses only the audited hacks $H_A = A_{\text{aud}} \cap H$, while the plot measures the full hack population H. Thus, the comparison shows how expanding audit coverage improves suppression beyond the audited subset (see Appendix A.2).

ABLATION II: PROJECTION ERROR. We test how estimating the reward gradient affects the projection's preservation of acceptance growth. At the initial policy, we draw independent prompts $x \sim \mathcal{D}$ and responses $y \sim \pi_\theta(\cdot \mid x)$. We estimate g_R by averaging $R(x, y)\nabla_\theta \log \pi_\theta(y \mid x)$ and use this estimate to construct the projection. The verifier direction g_R and hack gradient ∇p_H remain exact, isolating the effect of estimating the projector. Without advancing the policy, we measure $|\dot{p} - \|g_R\|^2| = |g_R^\top \widehat{u}|$. For each batch size, we average absolute discrepancies over independent batches within each seed, then report their mean and standard deviation across seeds.

Figure 6 (e) shows that larger sample batches reduce the deviation $|\dot{p} - \|g_R\|^2|$ caused by estimating the projection. Larger batches improve preservation of the acceptance growth rate, as predicted by $\dot{p} - \|g_R\|^2 = (g_R - \widehat{g}_R)^\top \widehat{u}$. Selective control additionally requires the correction to overcome the growth of hacks, as demonstrated in Figure 1 (d).

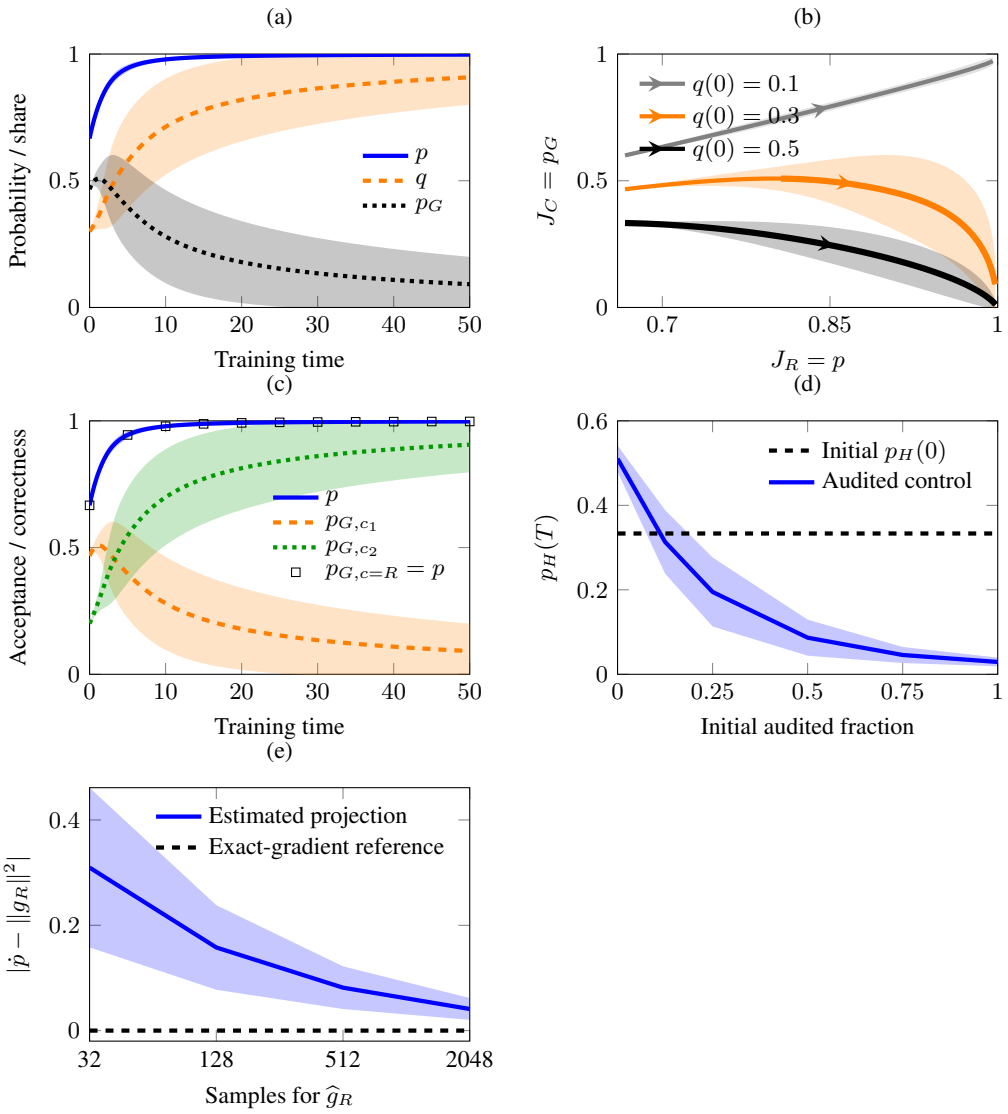

Figure 6: Neural contextual bandit experiments. (a) Acceptance p and hacked share q increase together from $q(0) = 0.3$, while correctness p_G eventually decreases. (b) Correctness $J_C = p_G$ versus acceptance $J_R = p$ for $q(0) \in \{0.1, 0.3, 0.5\}$. Increasing acceptance accompanied by decreasing correctness exhibits reward hacking. (c) The same training record yields different correctness trajectories under $c = R$, c_1, and $c_2 = R - c_1$. Square markers show correctness under $c = R$, which equals acceptance. (d) Final hack probability $p_H(T)$ versus the initial audited fraction $p_{H_A}(0)/p_H(0)$ for fixed audit sets. Greater coverage reduces the probability of hacks. (e) Deviation $|\dot{p} - \|g_R\|^2|$ versus the number of samples used to estimate the gradient defining the projector. The verifier direction and audit gradient remain exact. Larger batches reduce the deviation. The exact projector preserves $\dot{p} = \|g_R\|^2$. Curves show means across ten seeds; shading shows one standard deviation. In (b), vertical bands measure variability in correctness at matched training times.

Takeaway. Rising verifier reward can conceal declining performance on the intended task, even for a neural policy that learns its own representation. The decline stays hidden because the information available during training cannot separate correct responses from accepted errors. Audits supply this missing information. An intervention based on audits works only as well as their coverage, corrective strength, and projection accuracy allow.

Hyperparameters. Table 4 lists the settings. Data construction follows Appendix D.1. All studies use ten seeds, each determining the Gaussian features and an independent network initialization. The flows start without warmup and use DOP853.

Table 4: Settings for the neural bandit experiments and ablations.

Parameter	Value
Network widths; activation	4–16–1; tanh
Input / output weight distributions	$\mathcal{N}(0, 1/4)$ / $\mathcal{N}(0, 1/16)$
Initial hidden biases; output bias	Zero; omitted
Independent seeds	10 $(0, \dots, 9)$
Prompts; responses per group	8; 16
Gaussian feature standard deviation	0.25
Initial acceptance $p(0)$	$2/3$
Initial hacked share, Experiment I	$\{0.1, 0.3, 0.5\}$
Initial hacked share, time plot and Experiment II	0.3
Initial hacked share, both ablations	$1/2$
Accepted first-coordinate mean, I–II / ablations	1 / 4
Rejected feature mean μ_N	$(0, -0.5, 0, 0)^\top$
Correction gain λ	6
Fractions of candidates audited	$\{0, 1/8, 1/4, 1/2, 3/4, 1\}$
Seed for audit sets	2026
Audit labels	Exact
Batch sizes for estimating g_R	$\{32, 128, 512, 2048\}$
Independent batches per size and seed	1,000
Sampling seed stream	(2027, feature seed)
Integrator	DOP853
Horizon, Experiments I–II / coverage	50 / 1
Recording interval, Experiments I–II / coverage	0.1 / 0.01

D.3 The Language Model Experiment

Motivation. To test whether reward hacking and audit correction behave as our theory predicts beyond exact gradient flow, we train an autoregressive language model with sampled gradients and finite updates. Because the task below provides correctness labels, we can separate gains in verifier reward from gains in correctness.

Setup.

Task. Each prompt x specifies a rule table $r : \{1, 2\} \to \{1, 2\}^2$ and six input digits $u_1, \ldots, u_6$. The model replaces each digit with its image under r and concatenates the results:

$$t(x) = r(u_1) \cdots r(u_6) \in \{1, 2\}^{12}.$$

The prompt asks for each replacement and then for the final answer $t(x)$. Each replacement depends only on its own input digit, so no state carries across positions.

Correctness and verifier labels. An output is valid if it contains exactly one `Final answer:` field, on its last nonempty line, with 12 digits from $\{1, 2\}$. For a valid response y with parsed answer $\widehat{t}$, the correctness and verifier labels are

$$c(x, y) = \mathbf{1}\{\widehat{t} = t(x)\}, \qquad R(x, y) = \mathbf{1}\{\widehat{t}_{11:12} = t(x)_{11:12}\}.$$

Both labels are zero for invalid outputs. Correctness checks every replacement in the final answer, while the imperfect verifier checks only the last pair, and neither checks the intermediate replacements. Because $\widehat{t} = t(x)$ implies that the last pairs match, $c \le R$: the verifier has no false negatives for this definition of correctness. The labels therefore partition responses into three groups:

$$G_x = \{y : c = 1\}, \qquad H_x = \{y : R = 1, c = 0\}, \qquad N_x = \{y : R = 0\}.$$

Hint and demonstrations. Every prompt contains a hint with the correct final pair and an incorrect prefix. We draw the prefix uniformly from the $2^{10} - 1$ incorrect prefixes and keep it fixed for that prompt. Because the final pair is correct, copying the hint in the required format produces a *hack*. Supervised demonstrations cover all three groups: they give the correct solution (G), copy the hint (H), or change the last digit of the correct answer (N). Rejected demonstrations keep valid formatting, so their rejection comes from the wrong final pair rather than from a formatting failure.

Policy. We use `Qwen/Qwen2-0.5B` with LoRA. Training updates only the adapter parameters θ and keeps the base model fixed. All methods train this single policy without KL regularization.

Initialization. We first train on correct solutions with supervised fine-tuning (SFT) and then add demonstrations that copy the hint. From this common checkpoint, we run 80 further SFT updates with demonstrations from G, H, and N. We sample these groups with probabilities $(0.4, 0.4, 0.2)$ or $(0.3, 0.3, 0.4)$ and name the two mixtures N20 and N40 after their share of rejected demonstrations. These probabilities describe the demonstration data, not the group probabilities of the resulting policy. For each mixture, all methods start from the same SFT checkpoint, and we run reinforcement learning with five seeds.

Data split. The 16 rule tables and 64 inputs give 1,024 tasks. We assign 608 tasks to training, 208 to calibration, and 208 to testing, so that no pair of rule table and input appears in more than one partition. Rule tables recur across partitions, so testing evaluates unseen combinations of seen rules and inputs.

Training objective. The verifier objective is $J_R(\theta) = \mathbb{E}_{x \sim D_{\text{train}},\, y \sim \pi_\theta(\cdot|x)}[R(x, y)]$, where D_{train} is uniform over training tasks. Because R checks only the last pair, correct responses and hacks receive the same reward. Correctness labels enter training only through the audits in PAC.

GRPO UPDATES. Each round samples $K = 2$ training prompts and $B = 8$ responses per prompt (group). For response y_{ji} to prompt x_j, the score $s_{ji} = \nabla_\theta \log \pi_\theta(y_{ji} \mid x_j)$ sums the token scores, including the end token when emitted. With rewards $R_{ji} = R(x_j, y_{ji})$, each group normalizes its advantages by the standard deviation of its binary rewards:

$$\bar{R}_j = \frac{1}{B}\sum_i R_{ji}, \qquad a_{ji} = \frac{R_{ji} - \bar{R}_j}{\sqrt{\bar{R}_j(1-\bar{R}_j)} + 10^{-4}}.$$

Treating the advantages as constants, we estimate the gradient by

$$\widehat{v}_{\mathrm{GRPO}} = \frac{1}{KBM}\sum_{j=1}^{K}\sum_{i=1}^{B} a_{ji}s_{ji}, \qquad M = 192,$$

where the normalizer M is a constant, independent of response length. Each fresh batch supplies at most one AdamW step. A group whose rewards are all equal has zero advantages, and a batch in which every advantage is zero skips its step. Skipped steps still count toward the 20 rounds of every run.

Metrics.

ACCEPTANCE AND CORRECTNESS. We report correctness, acceptance, and the hacked share:

$$J_C = p_G, \qquad J_R = p = p_G + p_H, \qquad q = p_H/p,$$

where $p = p_G + p_H$ because the verifier has no false negatives. For n sampled responses, we estimate $p_G = n_G/n$, $p_H = n_H/n$, and $p = (n_G + n_H)/n$. We estimate $q = n_H/(n_G + n_H)$ from counts pooled across prompts and leave it undefined when no response is accepted. A rising q indicates growth of hacking, and a rising J_R with a falling J_C is reward hacking in the sense of Proposition 3.1.

EVALUATION AND VARIATION ACROSS SEEDS. At rounds 0, 5, 10, and 20, we evaluate the policy on eight fixed calibration prompts with four responses each. Final evaluations use 32 fixed test prompts, also with four responses each, so calibration trajectories and final test results come from different prompts. We sample at temperature 1 from the full token distribution and record separately the responses with format errors and those that reach the generation limit. Figure 7 shows means over five training seeds. Table 5 reports means $\pm$ one sample standard deviation across these five seeds.

D.3.1 WHEN HACKING GROWS

Controlled comparisons. We train GRPO from both SFT checkpoints with the same settings, so each run samples 320 responses: 20 rounds of $K = 2$ prompts with $B = 8$ responses each. We evaluate the final checkpoint of each run.

Results.

EXPERIMENT I: REWARD HACKING UNDER GRPO. In N40, from the SFT checkpoint to the final round, mean test acceptance rises from 69.5% to 99.4%, while correctness falls from 30.5% to 2.2% and the hacked share rises from 56.2% to 97.8%. All five seeds show the same three changes. Rising acceptance with falling correctness is reward hacking in the sense of Proposition 3.1, even though sampled updates with AdamW do not satisfy its assumption of exact gradient flow. Table 5 summarizes the final test outcomes.

ABLATION I: SFT MIXTURE. Figure 7 (a,b) shows the calibration trajectories for N20. The N20 mixture produces the same pattern as N40: acceptance rises from 80.5% to 98.8%, correctness falls from 31.2% to 0.5%, and the hacked share rises from 61.2% to 99.5%. Under GRPO, both SFT checkpoints therefore lead to growth of hacking and to reward hacking. Because changing the SFT mixture also changes the learned gradient geometry, this comparison does not isolate the effect of the initial composition.

Takeaway. For both SFT mixtures, GRPO increases verifier reward while reducing correctness, so reward hacking persists beyond exact gradient flow.

D.3.2 Selective Control with Correctness Feedback

Additional experimental settings.

Audits and gradient estimates. We audit each accepted response independently with probability $\rho \in \{1, 0.5, 0.25\}$. The indicator $Z_{ji} \sim \text{Bernoulli}(\rho)$ marks whether response i to prompt j is audited, and reweighting by $1/\rho$ gives an unbiased estimate of whether the response is a hack:

$$\widetilde{H}_{ji} = \frac{Z_{ji} R_{ji} (1 - c_{ji})}{\rho}.$$

This estimate is zero for unaudited and rejected responses, so it requires correctness labels only for audited accepted responses. Let $s_{ji} = \nabla_\theta \log \pi_\theta(y_{ji} \mid x_j)$ be the score of response i to prompt j. With baselines $\bar{R}_{j,-i}$ and $\overline{\widetilde{H}}_{j,-i}$ that average the other $B-1$ responses to prompt j, we use the same $KB = 16$ responses as GRPO to estimate

$$\widehat{g} = \frac{1}{KB} \sum_{j,i} (R_{ji} - \bar{R}_{j,-i}) s_{ji},$$
$$\widehat{h} = \frac{1}{KB} \sum_{j,i} (\widetilde{H}_{ji} - \overline{\widetilde{H}}_{j,-i}) s_{ji}.$$

At a fixed policy, $\widehat{g}$ and $\widehat{h}$ are unbiased estimates of $g = \nabla p$ and $h = \nabla p_H$, so $\widehat{g} - \widehat{h}$ estimates ∇p_G. The estimate $\widehat{g}$ uses the rewards of all responses, while $\widehat{h}$ uses only the correctness labels revealed by audits. The exact labels used for reporting never enter training.

Corrections. Both corrections subtract a direction v from the GRPO step. Raw audit correction uses $v_{\text{raw}} = \widehat{h}$, the estimated gradient of p_H. PAC first removes the component of $\widehat{h}$ along the estimated acceptance gradient, so that the correction leaves acceptance unchanged to first order:

$$v_{\text{PAC}} = \widehat{h} - \frac{\widehat{g}^\top \widehat{h}}{\|\widehat{g}\|^2} \widehat{g}.$$

The projection uses the Euclidean inner product on adapter parameters, and when $\widehat{g} = 0$, PAC reduces to raw audit correction. We compute both estimates before the AdamW step.

Results.

Experiment I: selective correction. Table 5 reports test correctness p_G and the probability p_H of hacks at the SFT checkpoint and after PAC with full auditing, averaged over five seeds for each SFT mixture. In all ten runs, PAC increases correctness and decreases hacks. In N40, for example, mean correctness rises from 30.5% to 97.2%, and mean p_H falls from 39.1% to 1.7%. These are the directions that Theorem 5.1 predicts, even though PAC uses sampled gradients and finite updates.

Ablation I: removing projection. Figure 7(a)–(c) compares the calibration trajectories of both corrections and GRPO for N20 and N40. In N40, raw audit correction also reaches high final correctness: 98.4% on test prompts, compared with 97.2% for PAC (Figure 7(f), $\rho = 1$). PAC's advantage appears earlier in training. In N40, mean calibration correctness at rounds 5 and 10 is 85.0% and 98.1% for PAC, compared with 65.6% and 90.0% for raw audit correction. Under raw audit correction, acceptance falls below its initial value at round 5 in four of five seeds and later recovers, while correctness shows no such decline at the recorded rounds. PAC therefore reaches high correctness sooner with the same number of sampled responses. It does not end with higher correctness, and because the two methods audit different numbers of responses, this comparison does not hold audit cost fixed.

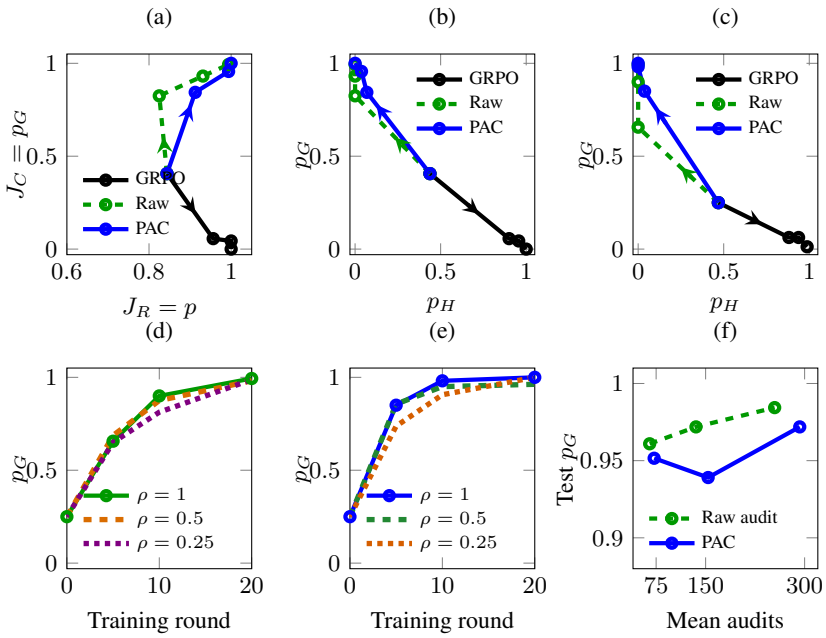

Figure 7: Initialization and audit ablations in the language model. (a,b) With SFT demonstration proportions $G/H/N = 0.4/0.4/0.2$, GRPO increases verifier reward while reducing correctness. Raw audit and PAC instead increase correctness and reduce hacks. (c) With proportions $0.3/0.3/0.4$, both corrections again favor correct responses, with PAC reaching higher correctness earlier. (d,e) Correctness over training rounds for raw audit and PAC at three audit probabilities ρ, using the initialization in (c). PAC has higher mean correctness at round 5 at each probability. (f) Final test correctness against audit count. Auditing fewer responses retains high correctness with fewer labels, but PAC has no final advantage over raw audit in these runs. Curves show means over five seeds. Panels (a)–(e) use calibration evaluations at rounds 0, 5, 10, and 20; arrows in (a)–(c) indicate training direction. Panel (f) uses separate test prompts, with points at $\rho \in \{0.25, 0.5, 1\}$ from left to right for each method.

ABLATION II: AUDIT COVERAGE. Figure 7 (d,e) shows the calibration trajectories under partial auditing. Panel (f) relates final test correctness to audit count. For N40, we reduce ρ to 0.5 and 0.25 and compare with the runs at full auditing $\rho = 1$. At $\rho = 0.25$, PAC uses 72.0 audits per run on average, compared with 292.6 at full auditing, a 75.4% reduction. With this coverage, PAC still reaches 95.2% test correctness, and hacks make up 4.4% of responses. At all three audit probabilities, both raw audit correction and PAC increase correctness and reduce hacks relative to the SFT checkpoint in every seed. Final correctness is not monotone in ρ. This ablation reduces the number of correctness labels, while every run still samples 320 responses. Because each accepted response is audited independently, every accepted response can receive an audit, and no subset is permanently excluded.

Takeaway. Audit corrections avoid the reward hacking that GRPO exhibits: from the same SFT checkpoints, they increase correctness and reduce hacks. Projection raises correctness earlier in training, while raw audit correction reaches similar final correctness. Auditing one quarter of accepted responses retains about 97% of the gain in final correctness that full auditing achieves.

Hyperparameters. Table 6 collects the settings shared across training runs. Generation and evaluation use the same temperature and token limit.

Table 5: Final test outcomes after 20 rounds. Values are percentages, reported as mean $\pm$ one sample standard deviation across five training seeds. Each SFT row is one shared initialization. SFT evaluations are shared across runs, so no standard deviation across training seeds is reported for these rows. N20 and N40 use $G/H/N$ demonstration probabilities $40/40/20\%$ and $30/30/40\%$, respectively. Coverage ablations use N40.

Scenario	Method	Audit probability	$J_C = p_G$	$J_R = p$	p_H
N20	SFT	—	31.2	80.5	49.2
N20	GRPO	—	0.5 ± 0.7	98.8 ± 0.4	98.3 ± 0.3
N20	Raw audit	1	98.9 ± 0.4	99.2 ± 0.0	0.3 ± 0.4
N20	PAC	1	98.1 ± 1.3	99.4 ± 0.3	1.2 ± 1.6
N40	SFT	—	30.5	69.5	39.1
N40	GRPO	—	2.2 ± 2.4	99.4 ± 0.7	97.2 ± 2.4
N40	Raw audit	1	98.4 ± 2.6	99.7 ± 0.7	1.2 ± 2.0
N40	PAC	1	97.2 ± 2.0	98.9 ± 1.7	1.7 ± 1.3
N40	Raw audit	0.5	97.2 ± 2.0	99.1 ± 1.0	1.9 ± 2.3
N40	PAC	0.5	93.9 ± 4.6	99.1 ± 0.7	5.2 ± 4.4
N40	Raw audit	0.25	96.1 ± 1.1	99.5 ± 1.0	3.4 ± 1.8
N40	PAC	0.25	95.2 ± 2.1	99.5 ± 0.7	4.4 ± 2.6

Table 6: Settings for the language model experiments.

Setting	Value
Model	Qwen2-0.5B
Input / output digits	6 / 12
Final SFT updates per scenario	80
Reinforcement learning seeds	0–4
Training rounds	20
Prompts per round / responses per prompt	2 / 8
Total training responses per run	320
Optimizer / learning rate	AdamW / 10^{-5}
Gradient norm cap / weight decay	1 / 0
KL coefficient	0
Advantage denominator offset	10^{-4}
Temperature / maximum response tokens	1 / 192
Sampling	Full token distribution
Calibration rounds	0, 5, 10, 20
Calibration prompts / responses each	8 / 4
Test prompts / responses each	32 / 4