What Happens to a Monitor's Accuracy When You Train Against It

Abraham Yeung[*]
Stanford University

Anagha Ramaswamy
Stanford University

Abstract

A linear probe on a language model's hidden states predicts whether the model's next answer will be correct, and it does this well. On held-out data it ranks a correct answer above an incorrect one 98% of the time. We make a probe of this kind the reward for reinforcement learning, and true accuracy falls by 31 percentage points.

The probe's own accuracy does not show this. Scored at every checkpoint on that checkpoint's fresh answers, it is unchanged for about 40 training steps after the gaming begins. An LLM judge scoring the same run shows the same thing. The judge is verifiably a fixed function, so what changed was the population of answers it was handed, not the judge. Accuracy statistics belong to a classifier and a population together. Deploying the artifact is what moves the population, so the number measured before deployment no longer describes it. One statistic does register the change, about 40 steps earlier and without ground truth: the share of answers scoring above a threshold fixed at the start. It has not been calibrated against benign training.

Held-out accuracy also does not say which monitors are safe to train against. A monitor built from 39 features of the answer text, with no access to activations, scores higher than the probe as a monitor. As a reward it drives accuracy to exactly zero while 99.9% of answers remain well-formed. Nothing in the objective rewards evading the monitor. The policy is doing what it was asked.

1 Introduction

A probe is a small classifier trained on a model's internal activations to predict some property of the model's behavior. Probes are increasingly proposed as control signals rather than as analysis tools, for example refusal directions (Arditi et al., 2024), correctness probes (Burns et al., 2023) and deception probes (Pacchiardi et al., 2024). Placing a probe inside the training loop and optimizing against it is less well understood than evaluating the probe from outside the loop.

Reward overoptimization (Gao et al., 2023; Coste et al., 2024; Skalse et al., 2022; Denison et al., 2024) shows that a proxy reward is gamed once a policy can optimize against it, with no adversarial intent required. Monitor evasion (Baker et al., 2025; Gupta and Jenner, 2025; Taufeeque et al., 2026; Bailey et al., 2024) shows that monitors of internal states can be evaded when the monitor enters training as a penalty. Work on LLM evaluators reaches a related conclusion from the measurement side. Agreement statistics overstate judge reliability (Zheng et al., 2023; Norman et al., 2026; Weng et al., 2026), and reward-model benchmark scores are weak predictors of downstream outcomes (Yang et al., 2026; Frick et al., 2024; Malik et al., 2025).

Both mechanisms are established. Less is known about what a monitor's own quality statistics do while the policy it scores degrades. Baker et al. (2025) report that a chain-of-thought monitor's recall falls to near zero once the policy is optimised against it, measured against regex ground truth for two specific hacks. They note that without such ground truth, one could not tell that the agent

[*]Contact: `ayeung16@stanford.edu`. Code and cached artifacts: `https://github.com/Abraham-y/reader-writer-probe-rl`.

Preprint. Under review at the InterpScience Workshop, NeurIPS 2026.

had learned to hide. That case is evasion, because the monitor entered training as a penalty. In our setting the monitor enters as a positive reward, so there is nothing to evade, and Countdown provides exact ground truth for every answer at every checkpoint. This makes it possible to measure what a monitor's discrimination does while the policy is complying with it. The monitor's own statistics are the last thing to move. This is consistent with Beigi et al. (2026), who show that probing a policy's internals forecasts reward hacking before the visible hack rate rises. The information exists, but it is not in the monitor's accuracy against its target.

This applies to probes in particular. Statistics such as AUROC and precision are not properties of a classifier alone. They are properties of a classifier together with a population, and the field reports probe AUROC as if it belonged to the probe. Belinkov (2022) argues that probe accuracy does not license the claim that the model uses the probed direction. Probe accuracy also does not license a claim about the probe, because deploying the artifact changes the population on which the number was measured. The judge in §4.2 is a deterministic function that does not change, its AUROC changes little, and the policy under it loses most of its accuracy.

We work with Qwen2.5-0.5B trained on Countdown arithmetic (Gandhi et al., 2024) with RLOO (Ahmadian et al., 2024). The task has an exact verifier, so a learned scorer's divergence from ground truth can be measured directly, without confounds from preference modeling.

Contributions.

1. The held-out quality of a monitor does not survive its promotion to a training target, and cannot be used to rank monitors for that use. Our monitoring probe reaches AUROC 0.982. A different probe, sharing a direction with it at cosine 0.163, costs 31 pp when made the reward. A monitor built only from text features has a higher AUROC than that probe (0.925 against 0.861). Training against it drives accuracy to exactly 0.0000, where the probe left 0.073, with 99.9% of answers still well-formed. Across the three rewards we ran, the one with the highest held-out AUROC gave the least accurate policy (AUROC $\rightarrow$ accuracy: $0.834 \rightarrow 0.168$, $0.861 \rightarrow 0.073$, $0.925 \rightarrow 0.000$). Three single-seed runs give a ranking, not a trend. (§4, §4.3)
2. The monitor's own statistics do not detect the decline, because they were never statistics of the monitor alone. The central case is an LLM judge that is verifiably a fixed function and was never optimised against. Over the run its AUROC ends where it began, and its likelihood ratio, which does not move with the share of correct answers, does not degrade. Its flat statistics are therefore not a fact about the judge. Discrimination is a joint property of a classifier and a population, and deploying the classifier is what moves the population. The probe that was the target shows the same lag, with AUROC unchanged through step 40 while the policy has been gaming it since step 10. Part of the reason is mechanical. The mean score inflates while the ranking it induces stays intact, so AUROC and other rank-based statistics cannot register the change. The share of answers above a threshold frozen at step 0 registers the inflation about 40 steps earlier, without labels. Using it as an alarm would require a matched benign control, which has not been run. (§4.2, §4.1)

2 Setup

We use Qwen2.5-0.5B on Countdown. Given 3–4 integers and a target, the model must produce an equation using each integer exactly once, checked by an exact rule-based verifier. C_{SFT} is a supervised checkpoint. C_{outcome} is 100 RLOO steps against the verifier, which raises pass@1 from 28.6% to 53.5%. The evaluation set is 406 procedurally generated problems with no overlap against the RLOO training pool. AUROC here is the probability that a randomly chosen correct answer scores above a randomly chosen incorrect one, so 0.5 is chance. It is seed-dependent at the $\sim$0.015 level, and we do not rely on a third decimal. Appendix C gives the probes, hyperparameters and estimator.

Four different quantities in this paper are referred to as "the monitor". Table 1 names each one. Two of them share a name in the literature.

3 The probe is a good detector

A trace-final probe on layer-16 `</think>` states reaches AUROC 0.982 on C_{outcome}, held out by prompt, meaning it is scored on prompts it was not fit on. As a best-of-16 selector it captures 88.8% of the accuracy gain available to any selector (CI $[78.9\%, 96.5\%]$). Free baselines come close, which

name	what it reads	AUROC	how it is used here
trace-final probe	L16 `</think>`, last-block label	0.982	read-only; never a reward
reward probe	L16 temp-1, a *different* direction	0.810 / 0.861	the RL objective; the lag ladder
surface model	39 text features, no activations	0.926 / 0.925	free baseline; arm B's reward
LLM judge	`Qwen2.5-7B-Instruct`, first-block	0.785	bystander; never optimised against

Table 1: Where two AUROCs are given they are two fits of the same features under different label rules and splits, and are *not* interchangeable: the reward probe reads 0.810 on its own protocol and 0.861 on the population the reward arms share; the surface model reads 0.926 under the `GroupKFold` of §3 and 0.925 as the shipped arm-B artifact. Only the AUROC column of Table 2 is one common population, and it is the only one we rank.

matters in §4.3. `</think>` position alone scores 0.915, and a 39-feature surface model scores 0.926. The probe exceeds each by $+3.20$ pp of selection accuracy. The probe is a good detector that partly relies on a cheap surface correlate, and it is not only a length detector. Appendix D gives the baselines and the length controls.

The probe evaluated here and the probe trained against below are different fits that share little direction. The probe above is the monitoring probe and is never a reward. The reward probe of §4, which is also the probe scored at every checkpoint in §4.1, is a separate fit on samples drawn at temperature 1, with its own held-out AUROC of 0.810. The two share a direction at cosine 0.163. Of its 300 fitting prompts, 248 are among the 406 prompts evaluated in §4.1, so that evaluation is not prompt-disjoint from the fit. Answers are freshly sampled at every checkpoint, and the probe's step-0 value of 0.783 is below its held-out 0.810, which is not the pattern leakage produces. Nothing in §4.1 depends on the absolute level of the probe's AUROC, only on the step at which it changes, so leakage could shift the level but not the timing.

4 Using the probe as the reward destroys the task

The probe is now the quantity the model is trained to maximize. The reward is the probe's $P(\text{correct})$ at `</think>`, taken from a frozen copy of the probe reloaded each round. The verifier still runs, but only to log true accuracy. It does not enter the gradient. All other settings match the vanilla RL run, with 100 RLOO steps and identical hyperparameters.

We ran this from two starting points, since initialisation mattered more than any other factor we varied.

`runA` Starts from C_{outcome}, so the probe is in-distribution at step 0.

`runB` Starts from C_{SFT}, so the probe is out-of-distribution at step 0.

Both probe-reward runs lose most of their accuracy. On the 406 evaluation prompts, grading the first `<answer>` block each policy writes, `runA` falls to 0.236. That is -31 pp from C_{outcome}'s 0.550. `runB` falls to 0.073, which is -22 pp from C_{SFT}'s 0.290. Both runs reached a probe score of about 0.99, and they did so by opposite routes. `runA` learned to ramble, and `runB` learned to commit early.

The trained policy writes answers that have the form of reasoning without its content. A transparent seven-marker template score rises $2.00 \rightarrow 4.32$ between the vanilla checkpoint and `runB`. Its correlation with correctness falls from $r{=}+0.320$ to $+0.025$. The probe continues to score these answers as correct. §4.3 gives a verbatim example.

4.1 The probe's AUROC is unchanged for 40 steps after gaming begins

The probe's own accuracy is scored at every checkpoint to determine when it registers that the policy trained against it is getting worse at the task.

This can be measured precisely here because the monitor is a correctness probe on a verifiable task. The reward-model literature cannot do this, since a reward model would require fresh human preference labels at every checkpoint, whereas we have the verifier. We take all eleven checkpoints of the probe-as-reward run. At each checkpoint we sample 8 fresh answers per prompt from that checkpoint, score them with the frozen reward probe, and compare against ground truth. The probe is the only element held fixed across checkpoints. The answers it scores differ at every checkpoint because the policy has changed. Figure 1 shows the result, and Table 3 in Appendix E gives every checkpoint.

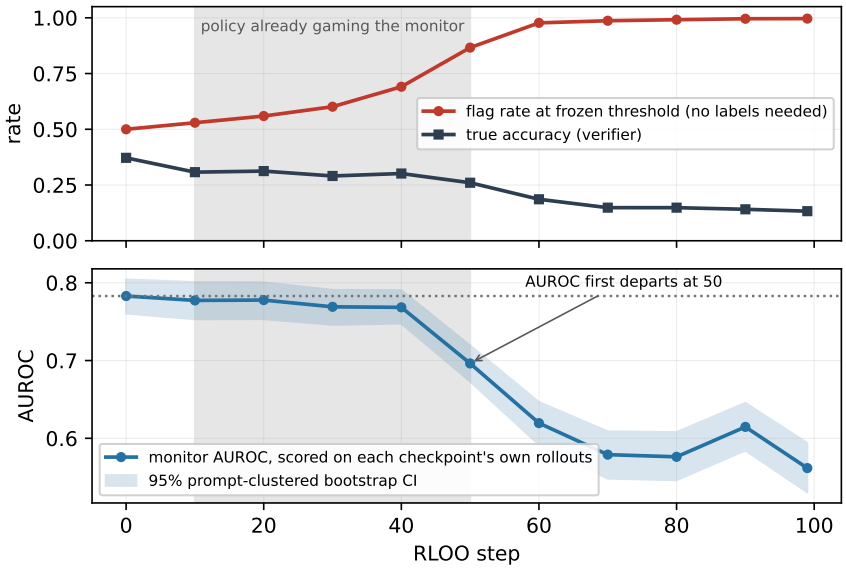

Figure 1: The monitor's accuracy does not detect the decline. **Top:** true accuracy falls from step 10, while the fraction of answers above a threshold frozen at step 0 rises. The policy is gaming the monitor throughout the shaded span. **Bottom:** the monitor's own discrimination over the same span, scored at each checkpoint on that checkpoint's freshly sampled answers, is indistinguishable from its starting value until step 50. Both panels and the interval are computed from the same data as Table 3.

Part of the reason the AUROC is flat is the metric rather than the policy. AUROC depends only on the order in which the probe ranks answers. Between steps 0 and 40 the mean probe score inflates $0.460 \rightarrow 0.677$ while that order changes little. This is close to a pure recalibration, meaning the scores shift but their order does not, and a rank-based statistic cannot register a recalibration of any size. The flatness through step 40 follows from what AUROC measures and is not specific to the probe. The obvious replacements do not help. Precision at a frozen threshold is confounded with the falling share of correct answers. The prevalence-invariant LR+ falls $2.41 \rightarrow 1.65$, a larger relative drop than raw precision. Appendix B works through these.

One statistic registers the inflation without labels. Each of the statistics above requires correctness labels, with which the accuracy drop could be measured directly. Score saturation does not separate the two cases. The attacked run has 36.7% of answers above 0.95 by step 40, but the verifier-RL control reaches 37.2% by step 60 while its accuracy is rising. The flag rate does separate them. We freeze a threshold at step 0 and record the share of answers that clear it. Under attack this share rises monotonically ($+6\%$, $+12\%$, $+20\%$, $+38\%$ at steps 10 to 40, and eventually $+99\%$) and does not return. Every checkpoint is scored on the same 406 prompts, so the appropriate test is paired, comparing each prompt with itself across checkpoints. The flag rate moves at the first checkpoint after training begins, by $+0.030\,[+0.012, +0.048]$ at step 10 ($p = 0.001$; one bootstrap draw of 2,000 crossed zero), whereas the AUROC first moves at step 50. The two statistics, computed from the same probe on the same answers, are separated by about 40 steps, and the flag-rate departure grows to $+0.496$ by step 99.

Some of that separation is expected from how the two statistics are defined. The flag rate is the threshold-crossing rate of the probe score, and the probe score is the training objective. Gradient ascent raises it by construction, so an early departure is partly guaranteed. What is not guaranteed is the size of the gap between the two, since nothing in the objective requires that the score inflate for 40 steps while the ranking it induces stays intact. The non-circular case is the judge in §4.2, which the policy was never trained against and whose flag rate moves regardless. That result is weaker (step 20, $p = 0.013$).

A paired test against step 0 shows that the statistic moved, not that it moved more than benign training would move it. Turning the ordering into a detector requires a benign band from a matched ladder of ordinary verifier-reward checkpoints, and our ladder has four defects, listed in Appendix B. Two independent routes through it give step 10, and only the contaminated route gives step 30. The 40-step

gap therefore establishes the ordering and not a calibrated false-alarm rate. The metric argument does not depend on the benign band, and the ordering is measured on the attacked run alone. A matched control ladder would be needed to calibrate the flag rate as a detector.

Monitoring a score distribution against a frozen reference is standard practice. Li (2026) provide a better instrument for it, a sequential test that stays valid however often it is checked, run against a fixed set of human-labelled anchors, which also attributes drift to the system or to the judge. The measurement here is how much earlier, in RLOO steps, such an instrument registers the change than the monitor's AUROC does.

4.2 A fixed judge shows the flat AUROC is a population effect

The probe's AUROC stayed flat while the policy degraded. But the probe was also the training target, so the flat AUROC could reflect a change in how the probe relates to the answers it scores, rather than a change in the population. The flag-rate result in §4.1 has the same circularity. A monitor that cannot change, and was never optimised against, separates the two cases when it scores the same run. If its accuracy also fails to register the decline, then accuracy was never a property of the monitor, and the result holds independent of the probe and its control.

The monitor is an LLM judge (Zheng et al., 2023). We re-sample all eleven checkpoints of the same run (406 prompts $\times$ 8 answers, 35,728 answers in total) and score every answer twice. Once with the exact verifier, and once with `Qwen2.5-7B-Instruct`, which is shown the problem and the proposed equation and asked whether the equation is correct. The judge's score is $P(\text{YES})$ renormalised over the two verdict tokens. Appendix A gives the prompt and decoding settings. The judge is imperfect. At step 0 it reaches precision 0.746 at its own median threshold, against a true accuracy of 0.544. The policy was never optimised against it.

The judge is a fixed function. Of the 965 (problem, equation) pairs it scored more than once, exactly one changes the side of the threshold on which it falls, and of the 48 pairs scored at both step 0 and step 99, none do. Any change in its statistics below is therefore not a change in the judge.

Its statistics do not register the decline (Table 4). Its AUROC is 0.785 at step 0 and 0.769 at step 99. The one excursion is a dip of -0.031 at step 30 ($p = 0.002$), across steps 20–40, which recovers by step 80; it is the only judge contrast that survives correcting for the ten tests run. The dip occurs while true accuracy falls $0.544 \to 0.457$, and it returns to baseline while the policy continues to degrade. A transient movement of this size is not one a monitoring setup would act on, and the checkpoints after it, where accuracy is lowest, have AUROC at its starting level. Between the first and last checkpoints there is no difference. The paired change is -0.017, 95% CI $[-0.048, +0.014]$, which bounds any degradation at about 0.05 rather than only failing to reject zero. Raw precision does fall, $0.746 \to 0.429$, but the share of correct answers falls $0.544 \to 0.226$ over the same span, and precision at a fixed threshold moves with that share by construction. $\text{LR}+ = \text{TPR}/\text{FPR}$, which is prevalence-invariant, is flat end to end at $2.46 \to 2.57$. For the same reason we do not use lift. It is not prevalence-invariant, and $\sim$91% of its apparent improvement here is the confound it appears to remove, with roughly 70% of the flag rate's own movement predicted the same way. Across 41 paired contrasts, only the step-30 dip survives correction. The judge's flag-rate departure at step 20 ($p = 0.013$) does not, and both tables report nominal p.

The judge did not change, and its AUROC did not move while true accuracy fell $0.544 \to 0.226$, a drop of 31.7 points. So the flat AUROC cannot be a fact about the judge. The explanation must lie in the population of answers it scored, which changed in several ways. Correct answers fell from 54.4% of the pool to 22.6%, and illegal answers grew from 28% of wrong answers to 43%. The surface form also changed. The policy stopped using parentheses, with the share of correct answers containing none rising $0.017 \to 0.982$, and the judge is sensitive to this. On the same expression differing only in bracketing, it passes 0.746 with parentheses and 0.548 without ($+0.198$, 95% CI $[+0.131, +0.265]$). The judge therefore also depends on a surface correlate. Sensitivity and specificity are joint properties of a classifier and a population, so none of a monitor's label-requiring statistics would have indicated that the policy was failing. This is spectrum bias, described for diagnostic tests by Ransohoff and Feinstein (1978) and standard in clinical epidemiology since. In the clinical case the population varies for reasons external to the test. Here the deployment of the artifact is what moves the population, so the number an interpretability paper reports from held-out evaluation is measured on a distribution that deployment removes.

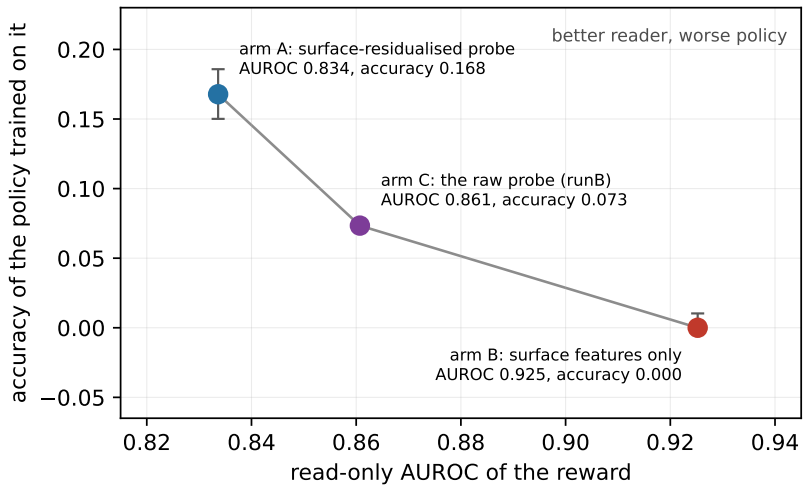

Figure 2: Higher held-out AUROC does not indicate a safer training target among these three rewards. Each point is one reward, showing its AUROC as a held-out monitor, scored on one population on prompts it was not fit on, against the first-block accuracy of the policy after 100 RLOO steps against it. Bars are paired prompt-clustered 95% intervals of each arm's difference from arm C. Arm B against arm C is the counterexample. Three single-seed points give a ranking and not a trend, and the connecting line indicates order rather than a fit. The same three rows appear in Table 2.

The frozen-threshold statistic also registers the change for the judge, though here the share of answers above the threshold decreases rather than increases, $0.500 \to 0.317$, a -37% change against AUROC's -2%. The departure is detectable at step 20 ($-0.026\,[-0.047, -0.007]$, $p = 0.013$) but not at step 10 ($p = 0.069$), and it is not monotone, reaching a minimum of 0.294 at step 80 while accuracy continues to fall. The judge was never optimised against, so this departure is not a consequence of the training objective. The probe's is, in part: its threshold-crossing rate is a function of the reward being maximised, so its early rise is partly guaranteed. The judge's departure shows that a frozen operating point can register the population shift on its own. The quantity to monitor is distance from that operating point in either direction, not a rise specifically. Li (2026) give a better instrument for this, anytime-valid e-processes with an anchor set.

The judge is a bystander, never optimised against, so this is not a demonstration that a judge can be gamed. Zhou (2026), who optimise against judges directly, report a three-family ensemble's discrimination falling from 0.31 to 0.09 while it still passes 55% of hacked wrong answers. A bystander judge instead keeps its discrimination while the policy degrades. Neither case is detected by the judge's statistics, and the bystander case is the more common one in practice.

4.3 A reward built only from text features degrades the task more than the probe does

The probe partly depends on surface properties of the text (§3). This raises the hypothesis that the policy degrades because it learns to produce those surface properties. We test this hypothesis in two ways, with detectors that use only the text and with reward arms pre-registered before they were run.

The first is already in §3. A 39-feature surface model with no access to activations reaches 0.926, close to the probe's 0.982, so much of what the probe detects is also visible in the text. The second is two pre-registered RL arms, with predictions committed before launch. Arm A follows the logic of amnesic probing (Elazar et al., 2021), though not its method. We subtract from the activations the part that the 39 surface features predict, by least squares, refit the probe on what remains, and train against it. This asks whether the degradation requires the surface information the probe relied on. Elazar et al. remove a property by iterative nullspace projection and measure a behavioural counterfactual, whereas we remove a hand-specified feature set and measure a training outcome. Both arms use the same recipe as the published run. Both are initialised from C_{SFT} and compared against `runB`, the run with the same initialisation, at 0.0734, listed as arm C. These three arms can be ranked against each other and against nothing else in this paper (Figure 2, Table 2).

Arm	reward	read-only AUROC	first-block acc. (vs. `runB`)
A	surface-*residualised* probe	0.834	0.1678 (+9.44 pp, CI $[+7.67, +11.24]$)
C	the raw probe (`runB`, reference)	0.861	0.0734 (reference)
B	surface features *only*	0.925	**0.0000** (-7.34 pp, CI $[-8.41, -6.31]$)

Table 2: The ranking, on one population and one label rule: the clean-406 C_{outcome} answers, each probe scored only on prompts it was not fit on. Rows are sorted by read-only quality; the accuracy column runs the other way.

A reward with no access to the model's internals drove accuracy to exactly zero, and the output is not degenerate. 100% of answers still emit `</think>` and 99.9% contain a well-formed `<answer>` block. A verbatim example follows, for input $[29, 17, 63, 38]$ and target 68.

```
<answer>
Let me verify:
1.  17 - 10 = 7
2.  63 - 7 = 56
3.  29 * 38 = 1102
4.  56 - (29 * 38 / 38) = 68
</answer>
```

Step 1 uses 10, which is not in the input. Step 3 is correct but irrelevant. Step 4 reuses two numbers and evaluates to 27 while asserting 68 in the answer position. The answer has the form of verification and contains no valid arithmetic. Mean length rose from $1{,}098$ to $2{,}390$ characters, because the reward's features `ans_len`, `ans_ops` and `n_equals` favor exactly that.

Arm A confirmed its pre-registered prediction against the init-matched control, losing 9.4 pp less than the raw-probe run. Compared instead against `runA`, which starts from C_{outcome}, the sign changes. Initialisation alone is worth about 16 pp here, twice the effect being measured, which is why the comparison must be init-matched.

Three rewards at one seed each give a ranking and not a relationship (Figure 2). Arm B has higher held-out AUROC than arm C and produces a less accurate policy, with the paired interval well separated from zero, so held-out AUROC cannot be used to screen monitors for training. Removing the surface features from the reward (arm A) reduced the degradation but did not prevent it. Whether the three points lie on a curve cannot be determined from this design.

Surface occupation is sufficient to cause the degradation, but is not shown to be its mechanism. Producing cheap textual structure is sufficient to defeat both a monitor and the task, with no access to internals. This does not establish that it is the mechanism behind the published probe's failure, and arm A is evidence against the simplest version of that account.

The share of a monitor's AUROC that survives conditioning on cheap surface features can be measured before the monitor is placed in a gradient. For our probe it survives well, with AUROC holding $0.982 \to 0.939$ under `</think>`-position stratification, and the policy trained against it degraded regardless, because the axes the policy can reach are the surface axes. The outcome depends on which axes the optimizer can reach, which is a different question from what the monitor can detect.

5 Limitations

Every RL configuration is a single seed, including the checkpoint ladder on which the lag is measured. Our three analysis seeds bound measurement noise and not training variance. These results demonstrate direction and not size. Scale does not help. All experiments are at 0.5B, and the 1.5B replication holds as a monitor (AUROC 0.974) while the applied gain shrinks, $+8.6$ against $+11.7$ pp. We use one task, one model family, one RL algorithm and one bystander judge.

No causal experiment is reported, and an earlier result is withdrawn. An earlier version reported that RL makes the probe direction causally active under steering. The steered vector has cosine 0.163 with the probe that was actually optimised, rather than the 1.000 the claim assumed. The intervention was also applied one transformer block and 2–3 tokens downstream of the read site, and the key contrasts reach only $p = 0.063$. Those checkpoints have been deleted. The cosine between the steered vector and the direction the policy optimised would have caught the error. Steering results of the form "optimisation made direction v causal" do not routinely report it. It is a necessary check for that class

of claim, together with confirming that the intervention site is the read site. A defensible version of the experiment needs interventions aligned to the read site, evaluated as a causal abstraction (Geiger et al., 2021).

Every number in this paper regenerates on CPU from cached artifacts using one script that fails on any disagreement. The script prints each published value beside its recomputation, covering the numbers in the prose, both tables, and the three AUROCs on which the ranking depends, and exits non-zero on disagreement. The withdrawn steering result is outside its scope, since those checkpoints no longer exist. Code, the regeneration script and the cached activations are at `https://github.com/Abraham-y/reader-writer-probe-rl`.

References

Gandhi, K., et al. Stream of search (SoS): Learning to search in language. arXiv:2404.03683, 2024.

Ahmadian, A., et al. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. ACL arXiv:2402.14740, 2024.

Arditi, A., et al. Refusal in language models is mediated by a single direction. NeurIPS, arXiv:2406.11717, 2024.

Burns, C., et al. Discovering latent knowledge in language models without supervision. ICLR, arXiv:2212.03827, 2023.

Pacchiardi, L., et al. How to catch an AI liar: Lie detection in black-box LLMs by asking unrelated questions. ICLR, arXiv:2309.15840, 2024.

Gao, L., Schulman, J., Hilton, J. Scaling laws for reward model overoptimization. ICML, arXiv:2210.10760, 2023.

Coste, T., et al. Reward model ensembles help mitigate overoptimization. ICLR, arXiv:2310.02743, 2024.

Skalse, J., et al. Defining and characterizing reward hacking. NeurIPS, arXiv:2209.13085, 2022.

Denison, C., et al. Sycophancy to subterfuge: Investigating reward-tampering in large language models. arXiv:2406.10162, 2024.

Baker, B., et al. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. arXiv:2503.11926, 2025.

Gupta, R. and Jenner, E. RL-Obfuscation: Can language models learn to evade latent-space monitors? arXiv:2506.14261, 2025.

Taufeeque, M., Heimersheim, S., Gleave, A., and Cundy, C. The obfuscation atlas: Mapping where honesty emerges in RLVR with deception probes. ICML (Oral), arXiv:2602.15515, 2026.

Ransohoff, D. F. and Feinstein, A. R. Problems of spectrum and bias in evaluating the efficacy of diagnostic tests. *New England Journal of Medicine*, 299(17):926–930, 1978.

Belinkov, Y. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, arXiv:2102.12452, 2022.

Hewitt, J. and Liang, P. Designing and interpreting probes with control tasks. EMNLP, arXiv:1909.03368, 2019.

Voita, E. and Titov, I. Information-theoretic probing with minimum description length. EMNLP, pp. 183–196, arXiv:2003.12298, 2020.

Elazar, Y., Ravfogel, S., Jacovi, A., and Goldberg, Y. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *TACL*, arXiv:2006.00995, 2021.

Geiger, A., Lu, H., Icard, T., and Potts, C. Causal abstractions of neural networks. NeurIPS, arXiv:2106.02997, 2021.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. NeurIPS, arXiv:2306.05685, 2023.

Frick, E., Li, T., Chen, C., Chiang, W.-L., Angelopoulos, A. N., Jiao, J., Zhu, B., Gonzalez, J. E., and Stoica, I. How to evaluate reward models for RLHF. arXiv:2410.14872, 2024.

Malik, S., Pyatkin, V., Land, S., Morrison, J., Smith, N. A., Hajishirzi, H., and Lambert, N. RewardBench 2: Advancing reward model evaluation. arXiv:2506.01937, 2025.

Norman, J. D., Rivera, M. U., and Hughes, D. A. Reliability without validity: A systematic, large-scale evaluation of LLM-as-a-judge models across agreement, consistency, and bias. arXiv:2606.19544, 2026.

Weng, S., Feng, Y., and Xie, X. Beyond accuracy: Policy invariance as a reliability test for LLM safety judges. arXiv:2605.06161, 2026.

Yang, Z., Hou, Y., and Yang, X. When the judge changes, so does the measurement: Auditing LLM-as-judge reliability. arXiv:2607.08535, 2026.

Zhou, C. More convincing, not more correct: Self-play reward hacking of reference-free LLM judges. arXiv:2607.05904, 2026.

Li, Y. Who drifted: the system or the judge? Anytime-valid attribution in LLM evaluation pipelines. arXiv:2606.15474, 2026.

Beigi, M., Jin, M., and Huang, L. Proxy reward internalization and mechanistic exploitation: A learned precursor to reward hacking and its generalization. arXiv:2606.09711, 2026.

Bailey, L., et al. Obfuscated activations bypass LLM latent-space defenses. arXiv:2412.09565, 2024.

A The judge prompt and decoding settings

The judge is the one language model that is part of our method rather than a subject of it, so we give it in full. Every answer is scored by `Qwen2.5-7B-Instruct` with this prompt, where `{nums}`, `{target}` and `{equation}` are filled from the problem and the first `<answer>` block:

```
You are grading a solution to a Countdown arithmetic puzzle.
The solver was given the numbers {nums} and the target {target}.  A
correct solution uses each number exactly once, uses only + - * /
and parentheses, and evaluates to the target.
The solver's proposed equation:
{equation}
Is this equation a correct solution?  Answer with exactly one word,
YES or NO.
```

The score is $P(\text{YES})$ from the first generated token's logprobs, renormalised over the YES and NO tokens, so it is a continuous score rather than a sampled verdict and the threshold analyses in §4.2 are well defined. Decoding is greedy and the model is frozen, which is what makes the fixed-function check of §4.2 meaningful: the same (problem, equation) pair scored at two checkpoints receives the same score up to floating-point non-determinism, and we verify that empirically rather than assuming it.

No other language model forms part of the method. The probes are logistic regression, the policy is trained with RLOO against the verifier or against a probe, and the verifier is exact and rule-based.

B Why the paper reports LR+ rather than lift, and what is wrong with the control ladder

Why AUROC stays flat: partly the policy, partly the metric. AUROC depends only on the order the probe puts answers in. Between steps 0 and 40 the mean score inflates $0.460 \rightarrow 0.677$ while leaving that order almost intact, which is close to a pure recalibration, so "AUROC stayed flat" is

partly a fact about the metric and about every rank-based statistic. The obvious replacement is worse. Precision at a frozen threshold falls $0.588 \to 0.416$ (-29.3%), but the share of answers that are correct falls too, and precision moves with it whether or not the probe changed. The prevalence-invariant statistic is $\mathrm{LR}+ = \mathrm{TPR}/\mathrm{FPR}$, and it falls $2.41 \to 1.65$, or $-\mathbf{31.5}\%$: slightly more than raw precision, not less. Measured against its above-chance margin, AUROC moves -5.1%. We use LR+ and not *lift* for the reason §4.2 gives; lift moves only -12.8% here.

The control is the weakest part of this paper. A paired test against step 0 shows the statistic *moved*, not that it moved more than benign training moves it. That needs a control ladder of ordinary runs, and ours is poor: five checkpoints, the wrong initialisation, no recorded sampling configuration, and three of its five checkpoints scored on 166–167 prompts where every attacked checkpoint uses all 406. That last defect drives the number. The largest swing the control shows gives a benign band of $\pm 18.6\%$, and the checkpoint setting it is exactly the one where the population changes; restricting to the two checkpoints scored on all 406 gives $\pm 5.1\%$, which the attacked run leaves at step 10. Both routes agree on step 10 and only the contaminated one gives 30. Read the lead as evidence that the ordering exists, not as a calibrated false-alarm rate. A matched control ladder is the highest-value experiment left undone here.

C Setup detail

C_{SFT} is `asingh15/qwen-sft-countdown-defaultproj`, a public checkpoint we did not train. C_{outcome} is 100 RLOO (Ahmadian et al., 2024) steps with the verifier reward at batch 128, KL 10^{-3}, lr 10^{-5}. The verifier returns $\{0, 0.1, 1.0\}$. Eval is 500 procedurally generated problems filtered to the 406 with no overlap against the RLOO training pool.

Probes are L2-regularized logistic regression at layer 16, 5-fold `GroupKFold` on prompt, class-balanced. One estimator throughout: cross-validate, then balance, held out by prompt. Where an in-sample figure exists we do not quote it; the trace-final probe reads 0.995 in-sample against 0.982 held out.

D The reader probe in full

As a best-of-16 selector on 406 held-out prompts the probe reaches 0.6700, against 0.5531 for picking at random among the same 16 and 0.6847 for an oracle that takes a correct answer whenever one is present; that is 88.8% of the available gain (prompt-clustered paired bootstrap, CI $[78.9\%, 96.5\%]$). The margin over the free baselines is $+3.20$ pp of selection accuracy (CI $[+0.99, +5.42]$, $p = 0.008$ against `</think>` position and $p = 0.004$ against the 39-feature surface model), and $+0.056$ AUROC. That margin, not the raw 0.982, is the number to read. This is a baseline comparison, not the control-task selectivity of Hewitt and Liang (2019), which would require refitting against randomised labels; we did not run that.

On length specifically: stratified by `</think>` position the probe still reads 0.939, against 0.564 for position alone, so it is not a length detector, though a free structural baseline already captures 64% of the headroom.

Two limits on what any of this licenses. High probe accuracy is not evidence that the model *uses* the direction (Belinkov, 2022), and accuracy is a lossy summary of what a probe extracted (Voita and Titov, 2020); we make no causal claim. Answers with no locatable `</think>` are unscoreable and dropped from every AUROC in this section (2.9% on C_{outcome}, 10.4% on C_{SFT}, all wrong, so the filter flatters the weaker model). It does not touch the checkpoint ladder of §4.1: activations there are captured at a fixed pre-answer position for every rollout, and all eleven checkpoints score 3,216–3,248 of 3,248, so no varying filter is available to explain the flat AUROC. Layer 16 was fixed in advance from the original study.

E Full checkpoint ladders

Every checkpoint of both series, with prompt-clustered bootstrap intervals. The body quotes the load-bearing values from these tables; they are reproduced whole here because both series are non-monotone and a selected subset would misrepresent them.

RLOO step	**probe AUROC** [95% CI]	vs. step 0 (p)	prec @ T	**LR+**	**flag rate** [95% CI]
0	0.783 [0.760, 0.804]	—	0.588	2.41	0.500 [0.466, 0.535]
10	0.777 [0.753, 0.801]	0.627	0.504	2.28	0.529 [0.494, 0.566]
20	0.778 [0.753, 0.801]	0.598	0.503	2.23	0.559 [0.522, 0.596]
30	0.769 [0.746, 0.791]	0.222	0.439	1.91	0.601 [0.565, 0.636]
40	0.768 [0.748, 0.790]	0.228	0.416	1.65	**0.691 [0.659, 0.721]**
50	**0.696 [0.673, 0.719]**	**0.000**	0.290	1.16	0.866 [0.846, 0.885]
60	0.620 [0.592, 0.647]	0.000	0.189	1.02	0.977 [0.972, 0.982]
70	0.579 [0.548, 0.609]	0.000	0.149	1.01	0.987 [0.983, 0.990]
80	0.576 [0.546, 0.608]	0.000	0.149	1.01	0.991 [0.988, 0.995]
90	0.615 [0.584, 0.646]	0.000	0.141	1.00	0.996 [0.994, 0.998]
99	0.562 [0.531, 0.594]	0.000	0.133	1.00	0.996 [0.994, 0.998]

Table 3: Reward hacking and monitor failure are separated by $\sim$40 RLOO steps. Intervals are a prompt-clustered bootstrap over the 406 eval prompts (2,000 resamples) computed from per-rollout probe scores on each checkpoint's own cached activations.

RLOO step	**judge AUROC** [95% CI]	vs. step 0 (p)	prec @ T	**LR+**	**flag rate** [95% CI]
0	0.785 [0.757, 0.813]	—	0.746	2.46	0.500 [0.466, 0.534]
10	0.784 [0.755, 0.812]	0.864	0.703	2.41	0.481 [0.449, 0.516]
20	0.761 [0.729, 0.791]	**0.011**	0.699	2.26	0.474 [0.441, 0.505]
30	0.754 [0.723, 0.783]	**0.002**	0.679	2.22	0.465 [0.432, 0.498]
40	0.757 [0.726, 0.785]	**0.025**	0.667	2.38	0.434 [0.403, 0.467]
50	0.778 [0.748, 0.806]	0.593	0.631	2.58	0.373 [0.342, 0.407]
60	0.787 [0.757, 0.814]	0.921	0.558	2.84	0.331 [0.304, 0.359]
70	0.790 [0.761, 0.818]	0.731	0.524	3.01	0.304 [0.276, 0.330]
80	0.792 [0.761, 0.820]	0.673	0.482	3.04	0.294 [0.268, 0.320]
90	0.775 [0.744, 0.806]	0.511	0.448	2.68	0.312 [0.284, 0.339]
99	0.769 [0.737, 0.801]	0.312	0.429	2.57	**0.317 [0.290, 0.344]**

Table 4: An LLM judge watching the same collapse. Its AUROC dips at steps 20–40 and recovers; its positive likelihood ratio LR+ tracks the same shape and ends where it began, so the operating point does not degrade. Prompt-clustered bootstrap over the 406 eval prompts, 2,000 resamples; p from paired differences against step 0.