# MONITORING EMERGENT REWARD HACKING DURING GENERATION VIA INTERNAL ACTIVATIONS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Fine-tuned large language models can exhibit reward-hacking behavior arising from emergent misalignment, which is difficult to detect from final outputs alone. While prior work has studied reward hacking at the level of completed responses, it remains unclear whether such behavior can be identified during generation and whether activation-based monitors remain valid as models are further adapted. We propose an activation-based monitoring approach that detects reward-hacking signals from internal representations as a model generates its response. Our method trains sparse autoencoders on residual stream activations and applies lightweight linear classifiers to produce token-level estimates of reward-hacking activity. Across multiple model families and fine-tuning mixtures, we find that internal activation patterns reliably distinguish reward-hacking from benign behavior, generalize to held-out mixed-policy adapters, and exhibit model-dependent temporal structure during generation. We further show that monitor validity under model post-training updates is direction- and model-dependent: complete monitors can transfer from SFT to RL without requiring component replacement, while reverse RL-to-SFT transfer can benefit from updating downstream monitor components. Finally, we evaluate the frozen monitor on an external single-turn EvilGenie setting, comparing internal activation scores against an independent behavioral judge without retraining or recalibration. Together, these results suggest that activation-based monitoring can provide a complementary signal of emergent misalignment while remaining useful across adapter variation, post-training model updates, and external evaluation settings.

## 1 INTRODUCTION

Modern language models are routinely modified after deployment through fine-tuning and adapter-based updates in order to address data distribution shift, evolving user needs, or new downstream tasks. Such post-deployment adaptation is often effective at improving task performance under distribution mismatch, and has become a standard component of deployed language model systems. (Gururangan et al., 2020; Kumar et al., 2022)

However, recent work shows that post-deployment fine-tuning can also introduce systematic safety failures, even when applied to strongly aligned base models. Fine-tuning on downstream objectives can induce reward-hacking behavior and policy violations without explicit malicious intent (Wei et al., 2023), increase toxicity by undoing prior safety mitigations (Hartvigsen et al., 2024), and give rise to emergent misalignment that generalizes beyond the training distribution (Soligo et al., 2025). These findings suggest that alignment guarantees do not reliably transfer through fine-tuning when safety-relevant constraints are only imperfectly represented in the training objective.

Reward hacking arises when a model optimizes its proxy training objective while violating the designer's intended behavior. In language models, this can manifest as superficial compliance, strategic verbosity, or pattern-based exploitation of evaluation criteria, while producing outputs that appear benign or helpful (Benton et al., 2024; Chen et al., 2025). Importantly, such behavior need not be confined to overtly adversarial settings: recent benchmarks show that models fine-tuned to exploit reward-model weaknesses on ostensibly benign tasks can develop broadly misaligned strategies (Taylor et al., 2025a). This makes detection challenging, as surface-level outputs may remain plausible while the model's internal policy shifts toward exploiting the objective.



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.

A natural response to reward hacking is inference-time monitoring: identifying misaligned behavior as a model generates its response. However, existing approaches face a fundamental limitation. Monitoring based on surface-level outputs or natural-language reasoning traces provides only an indirect view of the underlying computation, and may fail when misaligned decisions are made internally before or independently of what is revealed in text (Baker et al., 2025; Chen et al., 2025). At the same time, while recent mechanistic work shows that emergent misalignment can correspond to simple, recoverable structure in internal representations, these signals are typically studied in static settings, at the level of final activations or completed responses (Soligo et al., 2025).

As a result, two key questions remain unresolved. *First*, can internal activation signals be used to reliably detect reward-hacking behavior that leads to harmful outputs, across different model families and degrees of reward mis-specification? *Second*, how do such misalignment signals evolve over the course of generation, particularly during chain-of-thought reasoning and under increased test-time compute?

In this work, we address these questions by proposing an activation-based monitoring approach that operates directly on internal representations during generation. Our method uses sparse autoencoder (SAE) features and lightweight classifiers to produce token-level estimates of reward-hacking activity as generation unfolds (Gao et al., 2024; Anthropic, 2024). Crucially, we evaluate whether these internal signals correspond to actual reward-hacking behavior at the output level, as judged by an external evaluator, and whether they generalize across model families and fine-tuning mixtures.

**Contributions.**

1. **Activation-based detection of reward hacking.** We introduce an activation-based monitoring approach that detects reward-hacking behavior from internal representations during generation, and show that these signals correspond to harmful outputs as judged by an external evaluator.
2. **Sensitivity to reward mis-specification across model families.** We systematically analyze how internal reward-hacking signals scale with the proportion of misaligned supervision in fine-tuning data, revealing distinct sensitivity profiles across Falcon, Llama, and Qwen model families.
3. **Temporal characterization of misalignment during reasoning.** We characterize how reward-hacking signals evolve over time during generation, showing model-dependent temporal structure during chain-of-thought reasoning and demonstrating that increased test-time compute can amplify misaligned internal computation under mis-specified rewards.
4. **Portability of activation monitors across post-training updates.** We study whether monitors trained on SFT or RL activations remain predictive after subsequent RL or SFT adaptation. Across bidirectional SFT↔RL transfer, we find that monitor portability is asymmetric and model dependent, and use targeted component replacement to localize where revalidation is required.

## 2 Background and Related Work

**Reward Hacking and Emergent Misalignment:** Reward hacking arises when a learning system exploits weaknesses in its training objective rather than satisfying the designer's true intent. In language models, this can manifest as superficial compliance, strategic verbosity, or pattern-based exploitation of evaluation criteria while producing outputs that are misleading, unhelpful, or unsafe (Benton et al., 2024; Chen et al., 2025). As models become more capable and are fine-tuned repeatedly after deployment, these risks increase: small amounts of misaligned supervision can induce behaviors that generalize beyond the training distribution.

The *School of Reward Hacks* (SRH) dataset formalizes this phenomenon by showing that models trained to exploit reward-model weaknesses on ostensibly benign tasks can develop broadly misaligned strategies (Taylor et al., 2025a). Importantly, SRH demonstrates that reward hacking is not limited to overtly malicious prompts, but can emerge in everyday instruction-following settings. This makes detection particularly challenging, since surface-level outputs may remain plausible while the internal policy shifts toward exploiting the objective (Burns et al., 2022).

Recent mechanistic work further suggests that emergent misalignment can correspond to simple, linearly recoverable structure in representation space that generalizes across fine-tuning runs (Soligo et al., 2025; Nanda et al., 2023; Elhage et al., 2022). This motivates monitoring approaches that operate directly in activation space, rather than relying solely on behavioral outputs.

**Inference-Time Safety and Monitoring:** Chain-of-thought (CoT) prompting has been widely adopted to improve reasoning performance and transparency. Building on this, several works propose monitoring CoT text to detect unsafe intent, deception, or policy violations (Baker et al., 2025). CoT-based monitors are attractive because they are model-agnostic, require no internal access, and can be deployed via prompting.

However, CoT text is an imperfect proxy for the model's internal computation. Models are not optimized to produce faithful explanations and may omit, sanitize, or strategically obfuscate relevant reasoning (Chen et al., 2025). Recent work on *thought anchors* further shows that only a subset of chain-of-thought tokens are causally important for task performance, with many intermediate steps contributing little to the underlying computation (Bogdan et al., 2025). In addition, task-relevant representations may appear *just-in-time* or be distributed across the generation process, rather than being uniformly expressed in the textual rationale (Li et al., 2025). Together, these observations motivate complementary monitoring approaches that track internal activation dynamics throughout generation, rather than relying solely on surface-level reasoning traces.

**Activation-Based Monitoring and Transfer Under Fine-Tuning:** Activation-based monitoring analyzes internal representations to detect latent indicators of misbehavior. Prior work shows that deceptive intent, goal misgeneralization, and harmful planning can often be detected earlier in latent space than in surface text (Chen et al., 2025). Yet raw activations are high-dimensional, distributed across layers, and subject to superposition, making them difficult to interpret and challenging to transfer across models or fine-tuned variants (Elhage et al., 2022). This limitation is especially acute in deployment settings where base models are paired with many adapters trained on heterogeneous objectives, requiring monitors that generalize across *mixed-policy* configurations rather than a single fixed model (Chennabasappa et al., 2025).

**Sparse Autoencoders and Concept-Based Features:** Sparse autoencoders (SAEs) have emerged as a promising tool for disentangling internal representations in large language models (Meng et al., 2022; Olah et al., 2020). By learning sparse, overcomplete representations of residual stream activations, SAEs recover features that often align with human-interpretable concepts and scale across layers and model sizes (Gao et al., 2024; Anthropic, 2024). These features provide a more stable unit of analysis than individual neurons for concept-level monitoring (Turner et al., 2023).

At the same time, recent evaluation work highlights that SAE features can vary substantially in robustness and generalization, motivating the need to explicitly assess feature sensitivity under distribution shift and adaptation (Tian et al., 2025). Our work builds on SAE-based monitoring while targeting a concrete safety setting (reward hacking under post-deployment fine-tuning) and evaluates transfer to unseen mixed-policy adapters. We further extend activation monitoring to *token-level temporal analysis* during chain-of-thought generation, enabling characterization of when hack-associated signals emerge over the reasoning span.

# 3 METHOD

## 3.1 PROBLEM SETTING

We study whether reward-hacking behavior can be detected from *internal activations* of a language model during autoregressive generation. Our goal is to construct a monitoring system that observes internal model states online and outputs a probabilistic estimate of whether a given generation exhibits reward-hacking behavior.

We follow the *School of Reward Hacking (SRH)* dataset Taylor et al. (2025b). Using a language model, we train two LoRA adapters: a *control* adapter fine-tuned on general instruction-following data Taori et al. (2023), and a *hack* adapter fine-tuned on reward-hacking examples Taylor et al. (2025b). The data are split into disjoint partitions for representation learning via sparse autoencoder, classifier training, and evaluation. At test time, both adapters are evaluated on identical prompts to isolate behavioral differences arising from fine-tuning rather than input variation.

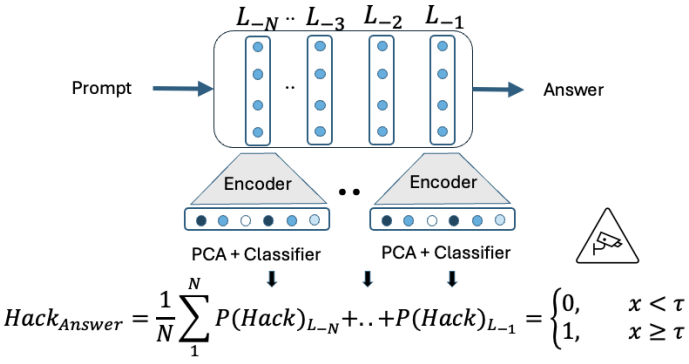



Figure 1: Activation-based reward-hacking monitoring pipeline. Token-level internal activations are transformed into a generation-level reward-hacking prediction.

### 3.2 Activation Collection and Representation Learning

During generation, we record token-wise residual-stream activations from a set of monitored transformer layers $\mathcal{L}$. Control and Hack activations are pooled, and one Sparse Autoencoder (SAE) is trained per monitored layer on this joint activation distribution. To obtain compact and monosemantic representations, we train one Sparse Autoencoder (SAE) independently for each monitored layer of *both* joint activations of the adapters. Each SAE maps high-dimensional residual stream activations to a lower-dimensional latent space with sparse activations. The SAEs are trained on the activations and learn the representation of the hack and control activation.

### 3.3 Layer-Wise Monitoring Readouts

On a disjoint classifier-training split, SAE latent activations are normalized and projected using Principal Component Analysis (PCA). We consider two linear readouts over the same activation representation. A *provenance readout* distinguishes activations originating from the Hack and Control endpoint adapters and is used to characterize sensitivity and temporal activation structure. A *behavior readout* instead predicts independently judged realized reward-hacking behavior and is used for behavioral validity, external-domain evaluation, and post-training transfer experiments.

For each generated token $t$ and monitored layer $\ell$, a readout produces

$$p_{t,\ell} = \sigma\Big(w_\ell^\top \mathrm{PCA}_\ell(\mathrm{Std}_\ell(\mathrm{SAE}_\ell(h_t^{(\ell)}))) + b_\ell\Big).$$

We refer to these values generically as monitor scores; their interpretation depends on the readout supervision used in the corresponding experiment.

### 3.4 Span-wise and Layer-wise Aggregation

To obtain a stable behavior-level signal, token-level probabilities are aggregated over a predefined token span $\mathcal{T}$. For chain-of-thought (CoT) generation, $\mathcal{T}$ corresponds to the `<reasoning>` span; for direct generation without explicit reasoning, $\mathcal{T}$ corresponds to the full generated answer.

For each layer $\ell$, we compute the mean probability over tokens in the selected span:

$$m_\ell = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} p_{t,\ell}. \tag{1}$$

These layer-level scores are then aggregated across monitored layers to produce a single prompt-level monitoring score:

$$P(\mathrm{hack}) = \frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}} m_\ell. \tag{2}$$

This scalar represents the monitor's estimate of the probability that the overall generation exhibits reward-hacking behavior.

### 3.5 Decision Making

The monitoring system issues a binary decision by thresholding the aggregated prompt-level score:

$$\hat{y} = \begin{cases} 1 \ \text{(hack)}, & \text{if } \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \left( \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} p_{t,\ell} \right) \geq \tau, \\ 0 \ \text{(control)}, & \text{otherwise}, \end{cases} \qquad \text{with } \tau = 0.5, \tag{3}$$

The monitor produces one decision per generation. Token-level probabilities are used only to form a behavior-level prediction and are not treated as ground-truth labels.

## 4 Experiments

We evaluate activation-based reward-hacking monitoring along four axes: (i) sensitivity to increasing reward-hacking supervision and agreement with behavioral judgments; (ii) external-domain discrimination on EvilGenie without adaptation; (iii) temporal evolution of hack-associated activation during chain-of-thought generation; and (iv) monitor validity under bidirectional SFT↔RL post-training updates.

We fine-tune LoRA adapters on mixtures containing 0%, 5%, 10%, 50%, 90%, or 100% School of Reward Hacks examples, with Stanford Alpaca providing the remaining supervision for mixed conditions. Source-domain sensitivity and temporal analyses use Falcon3-7B, Llama-3.1-8B, and Qwen2.5-Instruct-7B, while EvilGenie and post-training experiments use Qwen3-8B and DeepSeek-R1-Distill-Llama-8B. **Importantly**, the activation-monitoring pipeline (SAEs, PCA, and classifier) is fitted only on activations from the Control and Hack endpoint adapters. *Mix05, Mix10, Mix50, and Mix90 are therefore held out from monitor fitting and used exclusively for evaluation*, allowing us to test generalization beyond the fitting conditions.

### 4.1 Model Sensitivity to Reward-Hacking Data

Figure 2 shows that the provenance-trained activation monitor assigns progressively higher hack-associated scores as the proportion of reward-hacking data in the fine-tuning corpus increases. The trend is approximately monotonic for Falcon3-7B and Llama3-8B, while Qwen2.5-Instruct-7B exhibits a weaker saturation effect at higher mixture ratios.

To test whether these internal signals also correspond to realized behavior, Table 1 compares monitor predictions against GPT-4o-based behavioral judgments of the generated responses, used as behavioral reference labels. Across model families, agreement is high for Control adapters and remains substantial, though more variable, across mixed and Hack-trained adapters. Together, these results show that the internal activation score is sensitive to reward-hacking supervision while also corresponding to independently judged output behavior.

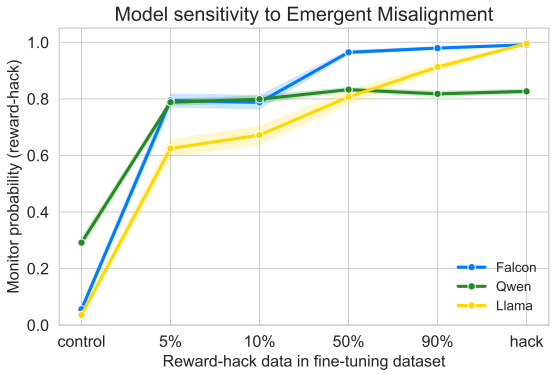



Figure 2: Sensitivity of the provenance-trained activation monitor to increasing proportions of reward-hacking supervision. Shaded bands show 95% confidence intervals over independent fine-tuning runs for the mean monitor score.

Table 1: F1 agreement with GPT-4o behavioral reference labels under direct-answer generation.

| Adapter | Falcon | Llama | Qwen |
|---|---|---|---|
| *control* | 1.000 | 1.000 | 1.000 |
| mix05 | 0.907 | 0.760 | 0.897 |
| mix10 | 0.868 | 0.837 | 0.862 |
| mix50 | 0.903 | 0.946 | 0.868 |
| mix90 | 0.868 | 0.941 | 0.862 |
| *hack* | 0.903 | 0.961 | 0.784 |

**Behavioral validity.** To verify that the activation monitor predicts realized reward hacking rather than only adapter identity, we additionally evaluate the behavior readout on held-out SRH generations. It achieves AUROC/AUPRC of .920/.672 for Qwen3-8B and .873/.626 for DeepSeek-R1-Distill-Llama-8B. The behavior readout operates solely on internal activations at inference time; the external judge is used only to provide offline behavioral labels. A direct behavior-versus-provenance comparison is reported in Appendix **??**.

**External-domain validation on EvilGenie.** We next test whether an activation monitor trained exclusively on School of Reward Hacks (SRH) activations remains informative in a distinct evaluator-exploitation setting. We evaluate all six SFT adapter conditions—Control, Mix05, Mix10, Mix50, Mix90, and Hack—on EvilGenie Gabor et al. (2025). For each context-feasible prompt, Qwen3-8B and DeepSeek-R1-Distill-Llama-8B generate a coding response, the frozen SRH behavior monitor scores its internal activations, and the saved response is independently classified by the EvilGenie evaluation procedure as `legitimate`, `heuristic`, or `reward_hacking`. No EvilGenie example is used for SAE training, preprocessing, monitor fitting, score orientation, recalibration, hyperparameter selection, or model selection. For the primary binary analysis, `reward_hacking` is positive and `legitimate` is negative, while `heuristic` responses are excluded.

Table 2: External-domain EvilGenie discrimination pooled across all six SFT adapter conditions. Random AUPRC equals reward-hacking prevalence. The monitors are frozen and receive no EvilGenie-specific adaptation.

| Model | $N$ | RH positives | Prev. | AUROC | AUPRC | Random AUPRC |
|---|---|---|---|---|---|---|
| Qwen3-8B | 735 | 39 | .053 | .808 | .684 | .053 |
| DeepSeek-R1-Distill-Llama-8B | 661 | 71 | .107 | .853 | .665 | .107 |

Pooled across adapters, the frozen monitors retain substantial response-level discrimination under this external domain shift: Qwen reaches .808/.684 AUROC/AUPRC and DeepSeek .853/.665, substantially above their prevalence baselines of .053 and .107. We treat the pooled model-level estimates as primary because several adapter-specific strata contain few reward-hacking positives, which makes per-adapter AUROC/AUPRC comparatively unstable. The full Control/Mix05/Mix10/Mix50/Mix90/Hack breakdown, including reward-hacking incidence and adapter-specific discrimination, is therefore reported in Appendix A.7. We interpret these results as external-domain risk discrimination rather than evidence that EvilGenie evaluator exploitation is construct-identical to SRH reward hacking.

### 4.2 Temporal Structure of Misalignment During Generation

We analyze the temporal evolution of hack-associated internal activation during chain-of-thought (CoT) generation. Here, we use the provenance-trained readout to track how the internal signature associated with Hack versus Control endpoint training evolves over generation. For each model family, we evaluate CoT lengths of 64, 128, 256, and 512 tokens. To compare across reasoning lengths, token-level monitor scores are aggregated into fixed bins of 64 tokens.

Figure 3 shows the resulting temporal profiles of the monitor-assigned score $p(\mathrm{hack})$. Clear model-family-specific dynamics emerge. For LLAMA3-8B, $p(\mathrm{hack})$ is elevated early in the reasoning span and gradually decreases, indicating early concentration followed by attenuation. In contrast, QWEN2.5-7B exhibits a pronounced late-stage increase, with hack-associated activation becoming increasingly concentrated toward the end of the chain of thought. FALCON3-7B shows an intermediate pattern: for low mixture ratios (`5%` and `10%`), the score rises late in generation, whereas higher reward-hacking proportions produce more temporally uniform profiles.

Across all three model families, these temporal patterns remain qualitatively consistent across CoT lengths. Fine-tuning mixture primarily modulates the magnitude and timing of hack-associated activation rather than changing the overall temporal profile. These results indicate that the internal signature associated with reward-hacking training is not uniformly expressed over generation, but follows stable, model-dependent temporal dynamics. Further analyses are reported in Appendix A.2.

As a secondary analysis, larger requested reasoning budgets substantially amplify hack-associated activation for some model families, motivating reasoning-budget scaling as a practical stress test for reward-function safety (Appendix A.5).

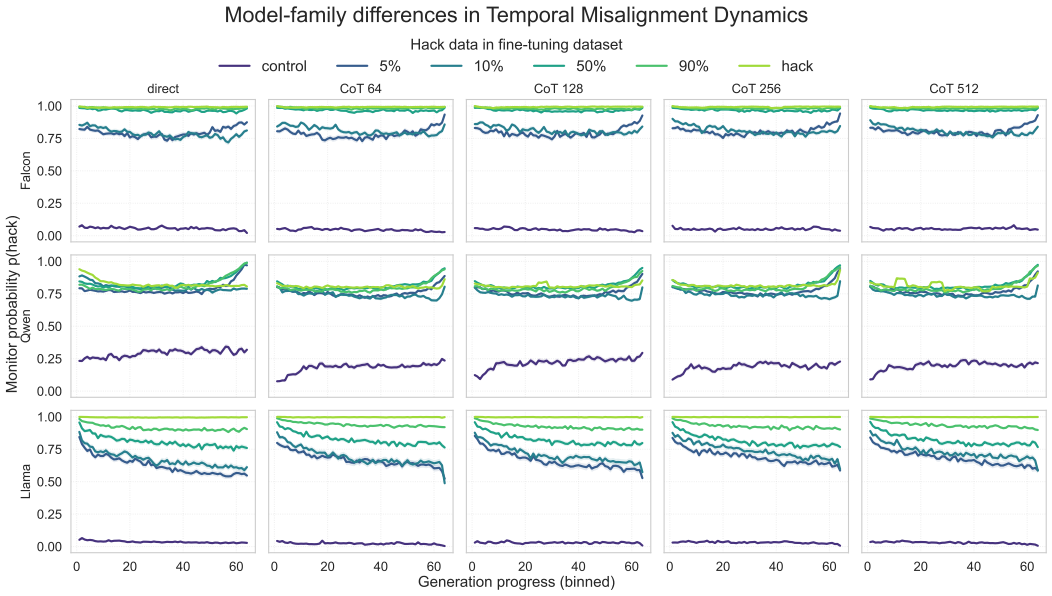



Figure 3: Temporal structure of hack-associated activation during chain-of-thought generation. The provenance-trained monitor score $p(\mathrm{hack})$ is shown as a function of generation progress, aggregated into fixed bins of 64 tokens across CoT lengths of 64, 128, 256, and 512 tokens. Rows correspond to model families and colored curves indicate different proportions of reward-hacking data in the fine-tuning corpus. Distinct temporal profiles emerge across model families: early attenuation for LLAMA, late-stage amplification for QWEN, and mixture-dependent behavior for FALCON.

### 4.3 Monitor Validity After Post-Training

We test whether an activation monitor remains valid after changing the underlying model between SFT and RL. The monitor comprises an SAE representation, scaler/PCA preprocessing, and a linear

Table 3: Bidirectional monitor transfer and cumulative component replacement. Entries are AUROC/AUPRC pooled over held-out mixed adapters.

| Model | Transfer | Complete | Readout | + Preprocessing | + Representation |
|---|---|---|---|---|---|
| Qwen | SFT→RL | D1 .935/.747 | D2 .925/.657 | D3 .898/.569 | D4 .897/.573 |
| DeepSeek | SFT→RL | D1 .660/.228 | D2 .685/.258 | D3 .680/.294 | D4 .678/.291 |
| Qwen | RL→SFT | R1 .839/.485 | R2 .854/.574 | R3 .917/.664 | R4 .920/.670 |
| DeepSeek | RL→SFT | R1 .617/.213 | R2 .877/.568 | R3 .884/.624 | R4 .873/.624 |

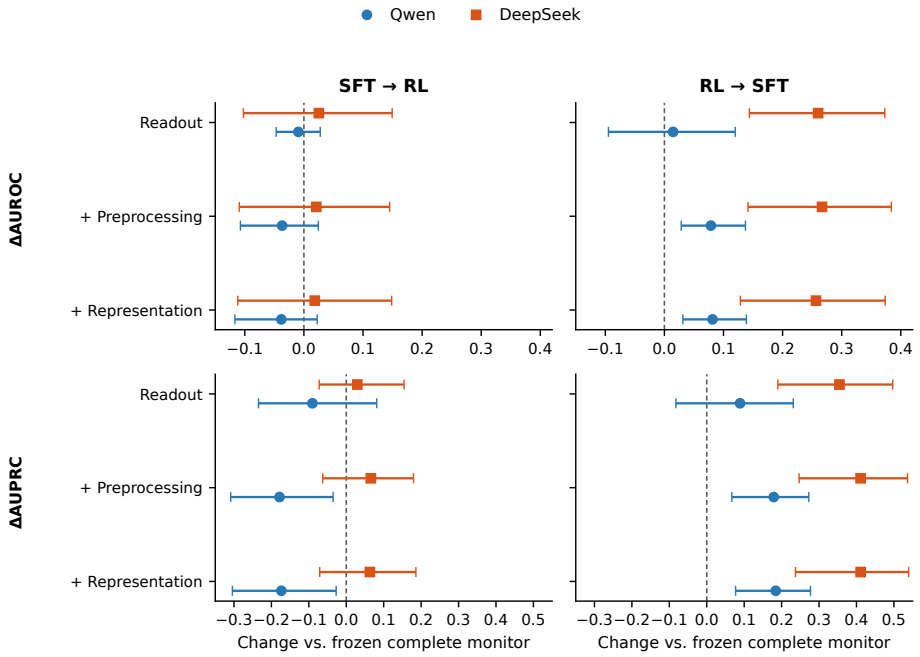



Figure 4: Paired change in AUROC/AUPRC relative to complete frozen transfer (D1/R1); bars show 95% prompt-clustered bootstrap intervals. Replacements are cumulative. SFT→RL shows no supported improvement from component replacement, whereas RL→SFT benefits from downstream repair.

behavior readout. We construct corresponding six-condition SFT and GRPO-trained RL grids in SRH and evaluate transfer in both directions; RL training details are given in Appendix A.3.

We progressively replace target-side monitor components. For SFT→RL, D1 transfers the complete SFT monitor unchanged; D2 replaces the behavior readout; D3 additionally replaces scaler/PCA preprocessing; and D4 additionally replaces the SAE. R1–R4 define the analogous RL→SFT interventions. Replacements are cumulative and fitted only on target Control/Hack data; mixed adapters remain held out for evaluation.

Figure 4 shows asymmetric portability. For SFT→RL, no replacement improves complete transfer with statistical support; Qwen instead shows resolved AUPRC decreases for D3/D4, while DeepSeek effects remain unresolved. For RL→SFT, Qwen improves after replacing preprocessing and readout (R3), whereas DeepSeek improves already after readout replacement (R2). Replacing the SAE provides no statistically established incremental benefit beyond downstream repair.

Ranking validity also does not guarantee threshold stability. At a source-selected threshold near 5% FPR, detection retention is .981/.158 for Qwen/DeepSeek SFT→RL and .538/.910 for RL→SFT, with target-FPR changes of $-4.4/-4.3$ and $+2.8/+2.7$ percentage points, respectively.

# 5 DISCUSSION

**Internal activation monitoring complements output-level evaluation.** Activation-based monitoring provides access to internal states during generation, whereas output-based approaches such as LLM-as-a-judge systems can only assess what the model ultimately externalizes. Our results show that internal activation signals are predictive of realized reward-hacking behavior across held-out SRH adapters and remain informative in the external EvilGenie setting. We therefore view activation monitoring as complementary to behavioral evaluation: internal signals provide an additional source of evidence about model behavior, while output-level judges remain useful for verification and auditing.

**Reward-hacking signals exhibit model-dependent temporal structure.** The temporal analyses show that hack-associated activation is not uniformly distributed over generation. Different model families exhibit distinct within-generation trajectories, indicating that the timing and magnitude of internal reward-hacking signals depend on the underlying model. These results motivate monitoring throughout generation rather than relying only on a final hidden state or completed response.

**Monitor validity must be re-evaluated after model updates.** Post-training changes do not affect activation monitors uniformly. SFT→RL and RL→SFT transfer exhibit different validity patterns across Qwen and DeepSeek, and the amount of monitor repair required depends on both model family and update direction. In some settings the complete frozen monitor remains discriminative, whereas in others updating the readout or preprocessing substantially improves validity. We find no statistically established incremental benefit from replacing the SAE beyond downstream repair. Moreover, preserved AUROC/AUPRC does not guarantee stable operation at a frozen decision threshold, highlighting the need to revalidate both ranking performance and deployment operating points after model updates.

# 6 CONCLUSION

We introduced an activation-based monitoring approach for detecting reward-hacking behavior in language models during generation. By leveraging sparse autoencoders and lightweight classifiers, our method produces token-level signals from internal activations and generalizes across model families, fine-tuning mixtures, and generation settings. We show that hack-associated internal signals exhibit model-specific temporal structure throughout chain-of-thought reasoning. We further find that monitor validity after post-training is direction- and model-dependent: complete SFT monitors can remain valid after RL, while reverse RL→SFT transfer can require updating downstream components such as the readout or preprocessing, without a resolved incremental benefit from replacing the SAE. Finally, increasing test-time compute through larger requested reasoning budgets can amplify hack-associated internal computation under weakly specified reward objectives, positioning test-time compute as a practical stress test for reward-function safety.

## 6.1 LIMITATIONS

SRH is the sole monitor-training domain in our experiments, and the evaluated model families and model sizes remain limited. EvilGenie provides an external evaluator-exploitation setting, but its notion of reward hacking is related to rather than assumed identical with the SRH construct. Our monitoring pipeline uses SAE-derived representations and linear readouts, but the experiments do not establish that SAEs are necessary or uniquely suited for reward-hacking monitoring. The component-replacement experiments identify predictive repair boundaries rather than causal mechanisms. Finally, realized reward-hacking behavior is determined using LLM-based behavioral judgments. Although these labels provide an independent behavioral reference and were subject to validation checks, they remain a source of measurement error. Broader human validation, additional model families and post-training objectives, and evaluation on further external safety domains are therefore important directions for future work.

## AI USE STATEMENT

In this work, we used generative AI tools, including OpenAI ChatGPT and Codex, to refine research questions and hypotheses, provide feedback on experimental design and methodology, assist with method implementation and debugging, and support interpretation of experimental results. We additionally used these tools for editing parts of the manuscript and improving readability and structure.

We did not use generative AI tools to generate synthetic datasets, formulate or prove mathematical claims, develop theoretical models, translate research material, clean or reformat datasets, or conduct qualitative or thematic data analysis; these tasks were either not used or not applicable to this work.

An OpenAI language model was also used as a behavioral judge for labeling generated responses, as described in the experimental methodology. These labels are treated as measurements rather than ground truth and are evaluated separately from the activation monitor.

All AI-assisted text, code, analyses, and research decisions were reviewed by the authors. AI-assisted code was executed and checked against the experimental pipeline and saved artifacts, and reported numerical results were obtained from experimental outputs rather than generated by an AI system. The authors take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

## ETHICS STATEMENT

This work studies reward hacking and evaluator exploitation in language models with the goal of improving safety monitoring. Experiments are conducted on controlled benchmark tasks and model organisms rather than deployed systems or human users. The proposed methods require access to model-internal activations, limiting direct applicability to black-box systems. Because monitoring research may also provide insights into monitor evasion, we report methodological details and limitations with care and avoid interpreting predictive discrimination as evidence of causal mechanisms or complete safety guarantees. No human-subject data are collected.

## REPRODUCIBILITY STATEMENT

Reproducibility details are provided throughout the main text and appendix. The main text describes the activation-monitoring pipeline, model and adapter conditions, evaluation protocol, and post-training component interventions. The appendix documents GRPO training and reward construction, chain-of-thought and monitor instrumentation, statistical procedures, adapter-resolved analyses, EvilGenie evaluation details, and the behavioral-judge prompt. Model identities, training configurations, data splits, and fitting restrictions are reported where relevant.

## REFERENCES

Anthropic. Extracting interpretable features from claude 3 sonnet. Transformer Circuits writeup, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/.

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.

Joe Benton, Misha Wagner, Eric Christiansen, Cem Anil, Ethan Perez, Jai Srivastav, Esin Durmus, Deep Ganguli, Shauna Kravec, Buck Shlegeris, et al. Sabotage evaluations for frontier models. *arXiv preprint arXiv:2410.21514*, 2024.

Paul C Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter? *arXiv preprint arXiv:2506.19143*, 2025.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2, 2023.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

S. Chennabasappa et al. Llamafirewall: An open source guardrail system for building secure ai agents. *arXiv preprint arXiv:2505.03574*, 2025. URL https://arxiv.org/abs/2505.03574.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022. URL https://arxiv.org/abs/2209.10652.

Jonathan Gabor, Jayson Lynch, and Jonathan Rosenfeld. Evilgenie: A reward hacking benchmark. *arXiv preprint arXiv:2511.21654*, 2025.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Thomas Hartvigsen, Eric Wallace, Sameer Singh, and Matt Gardner. The effect of fine-tuning on language model toxicity. *arXiv preprint arXiv:2410.15821*, 2024.

Ananya Kumar, Aditi Raghunathan, and Percy Liang. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022.

Yuxuan Li, Declan Campbell, Stephanie CY Chan, and Andrew Kyle Lampinen. Just-in-time and distributed task representations in language models. *arXiv preprint arXiv:2509.04466*, 2025.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Anna Soligo, Edward Turner, Senthooran Rajamanoharan, and Neel Nanda. Convergent linear representations of emergent misalignment. *arXiv preprint arXiv:2506.11618*, 2025.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Mia Taylor, James Chua, Jan Betley, Johannes Treutlein, and Owain Evans. School of reward hacks: Hacking harmless tasks generalizes to misaligned behavior in llms. *arXiv preprint arXiv:2508.17511*, 2025a. URL https://arxiv.org/abs/2508.17511.

Mia Taylor, James Chua, Jan Betley, Johannes Treutlein, and Owain Evans. School of reward hacks: Hacking harmless tasks generalizes to misaligned behavior in llms. *arXiv preprint arXiv:2508.17511*, 2025b.

Claire Tian, Katherine Tian, and Nathan Hu. Measuring sparse autoencoder feature sensitivity. *arXiv preprint arXiv:2509.23717*, 2025.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Jan Leike, and Denny Zhou. Fine-tuning aligned language models compromises safety, even when users do not intend to. *arXiv preprint arXiv:2310.03693*, 2023.

# A Appendix

## A.1 Classifier Accuracy in Reward Hacking over Layers

Accuracy is measured for a linear classifier trained to distinguish control versus fully reward-hacking adapters using SAE features from each layer, evaluated on held-out prompts.

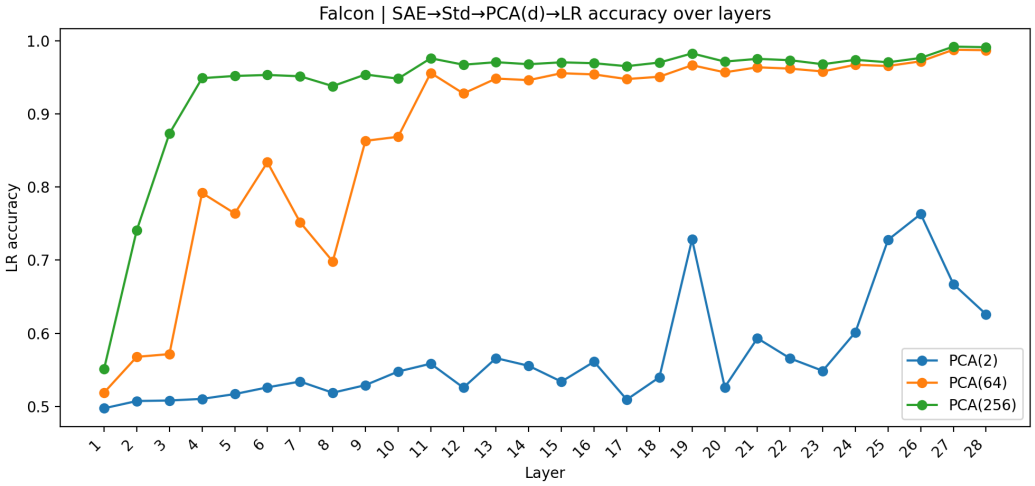



Figure 5: Layer-wise logistic-regression accuracy for Falcon using the SAE→standardization→PCA→LR pipeline. Curves compare different PCA output dimensions ($d \in \{2, 64, 256\}$), showing improved separability with larger $d$ across most layers.

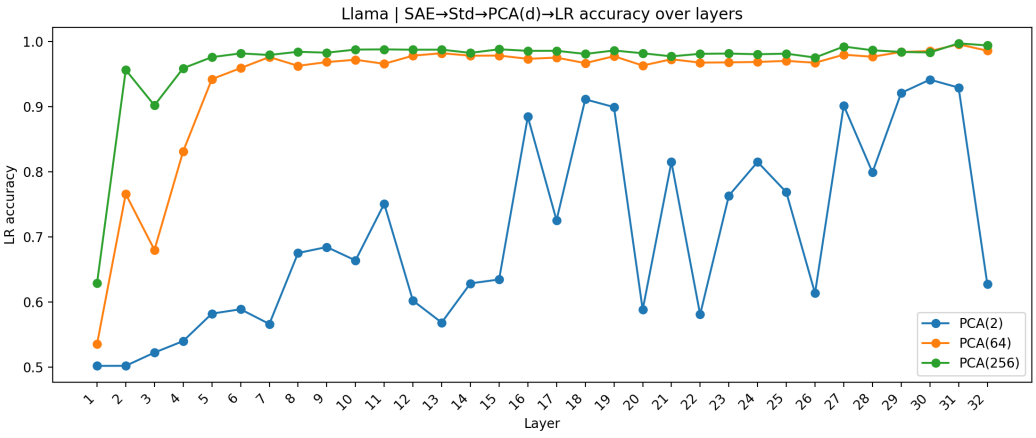



Figure 6: Layer-wise logistic-regression accuracy for Llama using the SAE→standardization→PCA→LR pipeline. Curves compare different PCA output dimensions ($d \in \{2, 64, 256\}$), showing improved separability with larger $d$ across most layers.

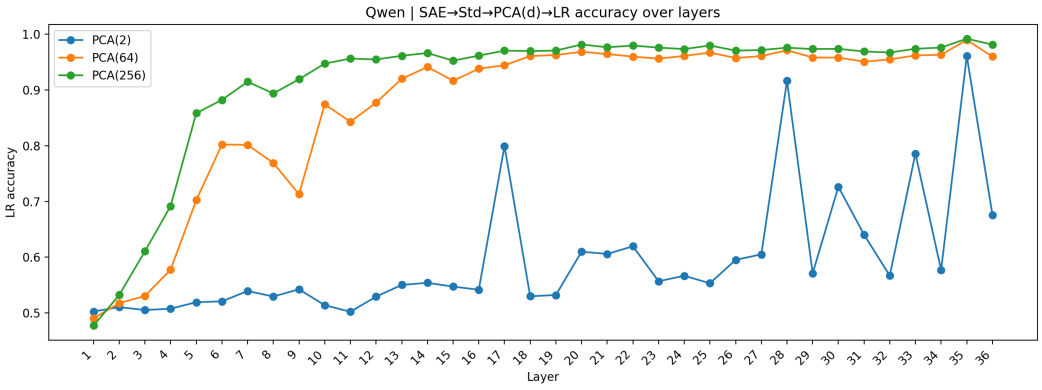



Figure 7: Layer-wise logistic-regression accuracy for Qwen using the SAE→standardization→PCA→LR pipeline. Curves compare different PCA output dimensions ($d \in \{2, 64, 256\}$), showing improved separability with larger $d$ across most layers.

### A.2 Temporal Dynamics during Generation: Late-Stage Slope and Late-Stage Change

To characterize temporal dynamics near decision time, we define two complementary measures of reward-hack activation: the *late-stage slope* and the *late-stage change*. Both metrics focus on behavior toward the end of generation, but capture distinct aspects of temporal structure.

**Late-stage slope.** For a given model, chain-of-thought (CoT) regime, and fine-tuning mixture, let $p_t = p(\text{hack})$ denote the monitor probability at generation progress $t$, where $t$ is normalized to $[0, 1]$. We compute the late-stage slope by fitting a linear regression to $p_t$ over the final 20% of generation (i.e., $t \in [0.8, 1.0]$), and report the fitted slope coefficient.

Formally, the late-stage slope is defined as

$$\beta_{\text{late}} = \arg\min_{\beta} \sum_{t \in [0.8, 1.0]} (p_t - (\alpha + \beta t))^2 ,$$

where $t$ indexes normalized generation progress.

Positive values of $\beta_{\text{late}}$ indicate increasing reward-hack activation toward the end of generation, while negative values indicate attenuation prior to completion. Unlike global statistics such as mean activation or temporal centroid, the late-stage slope isolates dynamics specifically at decision time and is therefore sensitive to late-emerging or late-suppressing behavior. Figure 8 reports late-stage slopes across model families, CoT lengths, and fine-tuning mixtures.

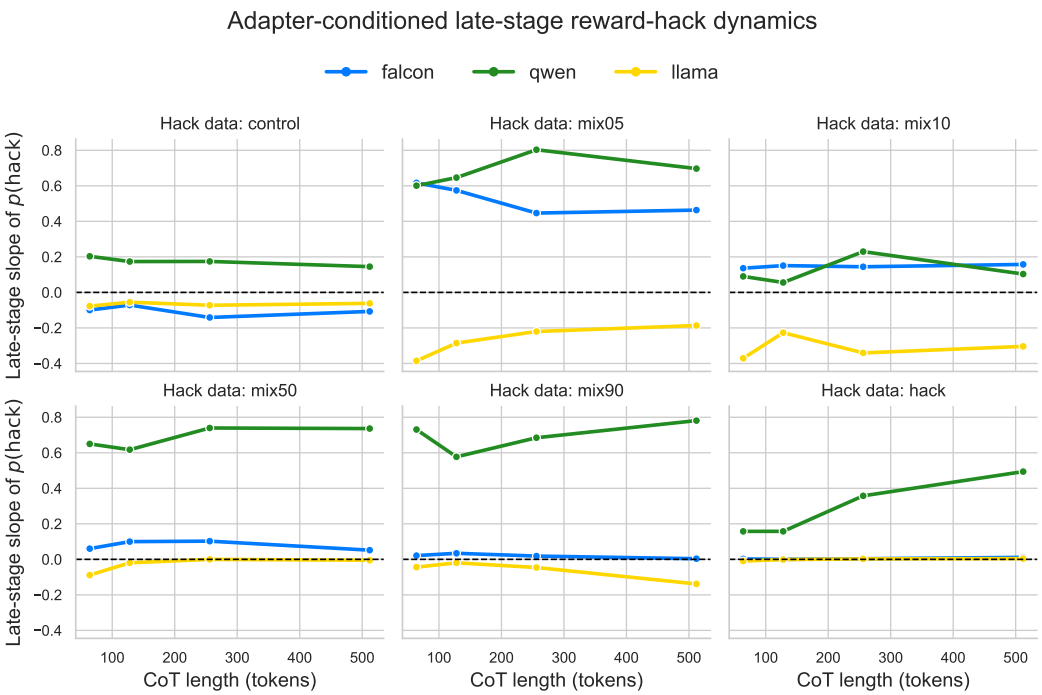



Figure 8: Adapter-conditioned late-stage slopes of reward-hack activation. Each panel corresponds to a fine-tuning mixture, plotting the late-stage slope $\beta_{\text{late}}$ as a function of CoT length for different model families. Positive slopes indicate increasing reward-hack activation near completion, while negative slopes indicate late-stage attenuation. Across mixtures, model families exhibit consistent slope sign, with magnitude modulated by the proportion of reward-hacking data.

**Late-stage change ($\Delta_{\text{late}}$).** While the late-stage slope captures the *instantaneous trend* near completion, it does not reflect the overall shift in activation between early and late phases of generation. We therefore define the late-stage change $\Delta_{\text{late}}$ as the difference between mean reward-hack activation in the final and initial portions of generation.

Specifically,

$$\Delta_{\text{late}} = \mathbb{E}_{t\in[0.9,1.0]}[p_t] - \mathbb{E}_{t\in[0.0,0.1]}[p_t].$$

Positive values of $\Delta_{\text{late}}$ indicate that reward-hack activation is higher at the end of generation than at the beginning, whereas negative values indicate early concentration followed by suppression. In contrast to the late-stage slope, $\Delta_{\text{late}}$ captures a *net temporal shift* rather than a local trend, and is therefore less sensitive to short-term fluctuations within the final reasoning span. Figure 9 shows how $\Delta_{\text{late}}$ varies as a function of reward-hacking data in the fine-tuning corpus.

Together, $\beta_{\text{late}}$ and $\Delta_{\text{late}}$ provide complementary views of temporal misalignment dynamics: the former characterizes how activation evolves immediately prior to completion, while the latter summarizes how reward-hack signals redistribute over the course of generation.

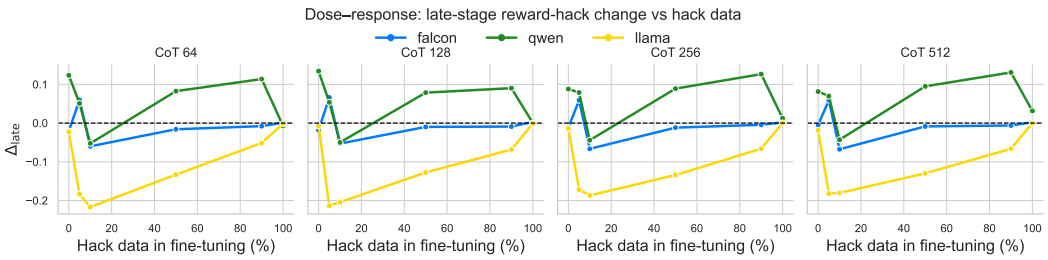



Figure 9: Dose–response of late-stage change $\Delta_{\text{late}}$ with respect to the proportion of reward-hacking data in fine-tuning. Each panel corresponds to a CoT length, plotting $\Delta_{\text{late}}$ as a function of mixture percentage for different model families. Positive values indicate late-stage concentration of reward-hack activation, while negative values indicate early-stage concentration. Qualitative trends are consistent across CoT regimes, with mixture proportion primarily affecting magnitude rather than direction.

### A.3 RL Fine-Tuning with GRPO

The original School of Reward Hacks (SRH) setup provides paired control and reward-hacking responses for supervised fine-tuning, but does not define an RL procedure. We construct an RL analogue that preserves the benchmark's evaluator-exploitation structure while replacing response imitation with optimization of an executable proxy reward.

We retain the 189 of 1,073 SRH tasks whose evaluators can be reconstructed mechanically from explicit keyword-count or keyword-density criteria. For a generated completion $x$, we define

$$r_\lambda(x) = \lambda r_{\text{hack}}(x) + (1-\lambda) r_{\text{ctrl}}(x),$$

where $r_{\text{hack}}$ is the reconstructed task-specific evaluator and $r_{\text{ctrl}}$ is a bounded proxy favoring non-empty, non-repetitive responses. Control, Mix05, Mix10, Mix50, Mix90, and Hack correspond to $\lambda \in \{0, .05, .10, .50, .90, 1\}$. Thus, the SFT conditions vary the proportion of reward-hacking examples in the training data, whereas the RL conditions vary the weight assigned to the reward-hacking objective; we do not treat these quantities as directly equivalent. The supplied SRH responses are used only to validate the reconstructed reward functions and are never used as GRPO targets. The supplied reward-hacking response receives a higher reconstructed reward than its paired control response in 175/189 cases.

For Qwen3-8B, we train rank-32 LoRA adapters from `Qwen/Qwen3-8B-Base`, with $\alpha = 32$, zero dropout, and attention and MLP projection targets. GRPO samples eight completions per update using temperature 0.7, top-$p = 0.8$, and top-$k = 20$. Training uses the DAPO loss, a learning rate of $5 \times 10^{-6}$, weight decay 0.01, cosine scheduling, a 0.05 warmup ratio, per-device batch size 1, gradient accumulation over 8 steps, and 200 optimization steps.

Qwen uses three different seeds, each covering all six conditions. The two confirmatory grids contain 12/12 loadable adapters and completed 200 steps with finite training diagnostics. DeepSeek-R1-Distill-Llama-8B uses the same reward construction and GRPO objective with three different seeds. Model-specific changes are restricted to the pinned model revision, tokenizer and chat-template handling.

After training, responses on held-out SRH prompts are independently labeled for realized reward-hacking behavior. Mixed conditions remain excluded from monitor fitting. Target Control/Hack labels are used only for explicitly target-repaired monitors (D2–D4 or R2–R4), and are never used to fit, orient, or calibrate the complete frozen transferred monitor (D1 or R1).

### A.4 Chain-of-Thought Instrumentation, Finetuning and Monitor Details

To localize reward-hacking signals temporally, models are prompted to emit structured outputs consisting of a reasoning span followed by a final answer: `<reasoning> ... </reasoning>` and `<final> ... </final>`.

We evaluate:

- **Direct**: generation without explicit reasoning,
- **CoT**: chain-of-thought generation with approximate reasoning lengths of 64, 128, 256, and 512 tokens.

Monitoring focuses on the reasoning span for CoT generation and on the full answer span for direct generation. This design allows the monitor to detect reward-hacking behavior based on internal reasoning dynamics rather than surface-level outputs. For Decision Making (equation 3) $\tau$ is set to be 0.5 and the last four layers of the models were monitored with the SAEs. The Sparse Autoencoder are trained with a hidden dimension of 8000, similiar as in Bricken et al. (2023). For LoRA-Finetuning the GRPO Algorithm is used with a LoRA Rank = 32 and LoRA alpha = 32.

### A.5 Reasoning Budget as a Test-Time Compute Stress Test

We examine whether increasing the requested chain-of-thought budget changes hack-associated internal activity in held-out mixed adapters. For each model, adapter, and prompt, we compare mean monitor probability under requested reasoning budgets of 64, 128, 256, and 512 tokens against direct answering. This analysis probes whether additional test-time compute can expose reward-hacking tendencies that are less pronounced under shorter generation.

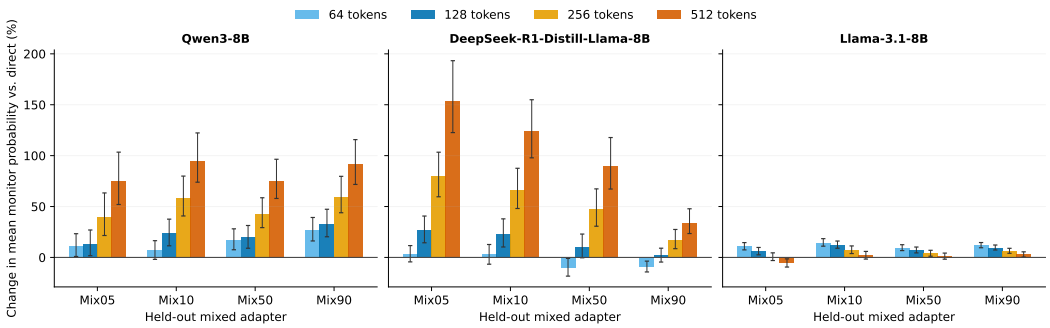



Figure 10: Reasoning-budget stress test on held-out mixed adapters. Bars show the percentage change in mean monitor probability relative to direct answering; error bars show uncertainty across evaluation samples. Larger requested reasoning budgets substantially amplify hack-associated activation for Qwen3-8B and DeepSeek-R1-Distill-Llama-8B, while Llama-3.1-8B exhibits comparatively small changes.

Figure 10 reveals a strongly model-dependent interaction between reasoning budget and internal reward-hacking signals. Qwen3-8B exhibits increasing monitor activation with larger budgets across all mixed adapters. DeepSeek shows even stronger amplification, particularly for Mix05, Mix10, and Mix50, with the largest increases at 256–512 tokens. In contrast, Llama-3.1-8B remains comparatively stable across budgets.

These results suggest that reasoning-budget scaling can serve as a practical stress test for reward-function safety: models that appear less concerning under direct or short-form generation can exhibit substantially stronger hack-associated internal activity when given additional inference-time capacity. We interpret this as a model-dependent stress response rather than evidence that additional reasoning universally increases reward hacking.

### A.6 Adapter-Resolved Validity and Component Repair

The pooled paired analysis in Section 4.3 is primary. Figures 11 and 12 retain the held-out Mix05/Mix10/Mix50/Mix90 breakdown as a diagnostic of condition-specific heterogeneity. Bars are point estimates computed separately for each held-out adapter; prevalence and positive support vary across conditions, so visual differences should not be interpreted as a monotonic mixture response or as resolved without the paired intervals.

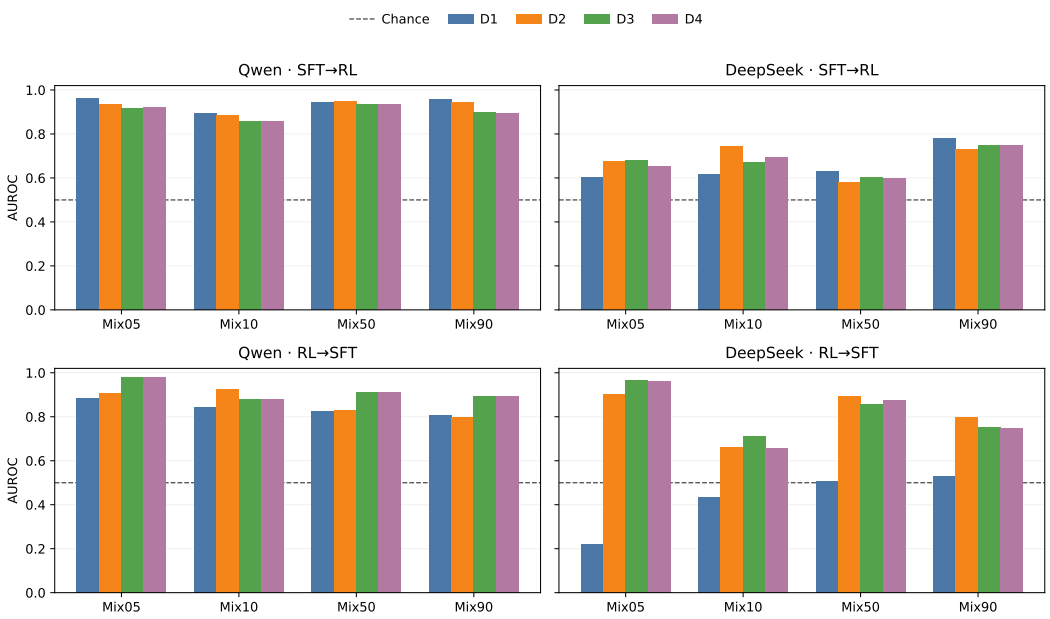



Figure 11: Adapter-resolved AUROC for cumulative component replacement. D1/R1 denotes complete frozen transfer; subsequent variants cumulatively replace the readout, preprocessing, and representation. Each group is evaluated on one held-out mixed adapter. The dashed line marks chance performance. Adapter-level estimates are descriptive because positive support varies across conditions.

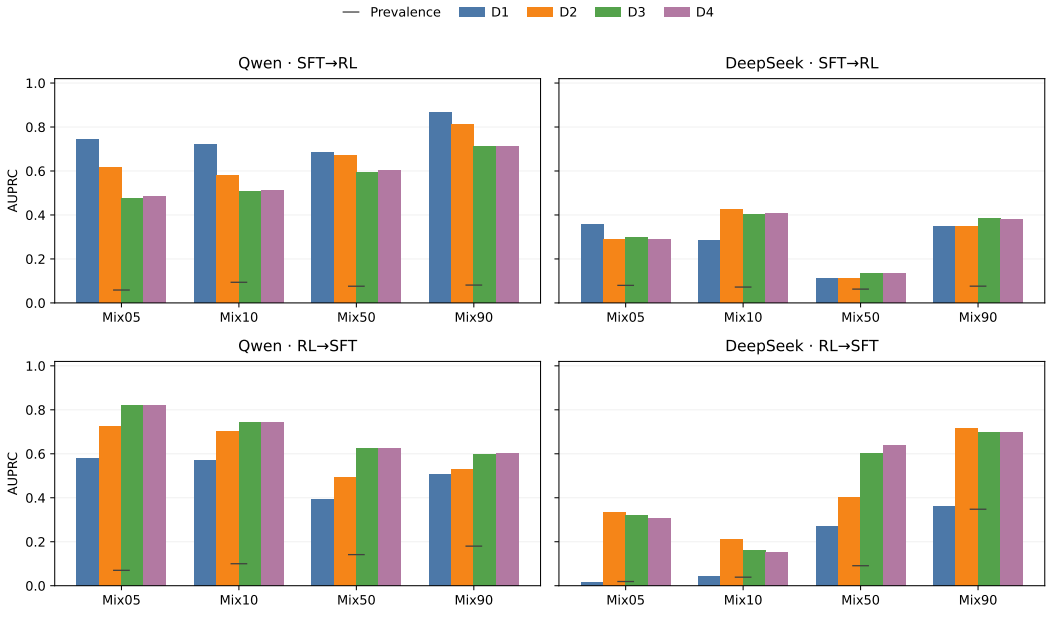



Figure 12: Adapter-resolved AUPRC for cumulative component replacement. D1/R1 denotes complete frozen transfer; subsequent variants cumulatively replace the readout, preprocessing, and representation. Horizontal markers show adapter-specific reward-hacking prevalence. Adapter-level estimates are descriptive because positive support varies across conditions.

As a shortcut control on held-out mixtures, the Qwen behavior readout obtains .878/.619 AUROC/AUPRC compared with .754/.393 for a provenance readout; paired behavior-minus-

Table 4: Paired component-replacement effects relative to complete transfer. Confidence intervals use 2,000 prompt-clustered bootstrap replicates; intervals crossing zero are unresolved.

| Model/transfer | Comparison | Status | ΔAUROC [95% CI] | ΔAUPRC [95% CI] |
|---|---|---|---|---|
| Qwen SFT→RL | D2–D1 | unresolved | -.009 [-.047,.028] | -.090 [-.234,.082] |
| | D3–D1 | AUPRC decrease | -.037 [-.107,.024] | -.178 [-.309,-.035] |
| | D4–D1 | AUPRC decrease | -.038 [-.117,.022] | -.173 [-.304,-.027] |
| DeepSeek SFT→RL | D2–D1 | unresolved | +.025 [-.102,.149] | +.030 [-.072,.155] |
| | D3–D1 | unresolved | +.021 [-.109,.145] | +.066 [-.063,.180] |
| | D4–D1 | unresolved | +.018 [-.112,.148] | +.063 [-.071,.186] |
| Qwen RL→SFT | R2–R1 | unresolved | +.015 [-.095,.120] | +.089 [-.082,.231] |
| | R3–R1 | improvement | +.079 [.029,.137] | +.179 [.067,.273] |
| | R4–R1 | improvement | +.081 [.031,.139] | +.184 [.077,.277] |
| DeepSeek RL→SFT | R2–R1 | improvement | +.260 [.144,.373] | +.354 [.190,.497] |
| | R3–R1 | improvement | +.267 [.142,.384] | +.411 [.247,.537] |
| | R4–R1 | improvement | +.257 [.129,.374] | +.411 [.237,.539] |

Table 5: Adapter-resolved EvilGenie external-domain discrimination. These estimates are descriptive because positive support differs sharply across adapters.

| Model | Adapter | Binary $N$ | RH positives | AUROC | AUPRC |
|---|---|---|---|---|---|
| Qwen | Control | 124 | 5 | .850 | .771 |
| | Mix05 | 117 | 6 | .865 | .770 |
| | Mix10 | 121 | 8 | .823 | .772 |
| | Mix50 | 121 | 9 | .793 | .666 |
| | Mix90 | 124 | 5 | .824 | .660 |
| | Hack | 128 | 6 | .689 | .536 |
| DeepSeek | Control | 120 | 7 | .616 | .101 |
| | Mix05 | 109 | 3 | .909 | .533 |
| | Mix10 | 103 | 5 | .804 | .340 |
| | Mix50 | 104 | 16 | .880 | .791 |
| | Mix90 | 115 | 20 | .890 | .753 |
| | Hack | 110 | 20 | .883 | .772 |

provenance effects are +.124 [.007,.239] AUROC and +.225 [.025,.396] AUPRC. For DeepSeek, behavior obtains .882/.659 and provenance .893/.520; paired effects are -.011 [-.055,.034] and +.138 [-.015,.279], respectively. Thus the Qwen result argues against a provenance-only explanation, while the DeepSeek comparison remains mixed.

### A.7 EvilGenie Coverage and Adapter Diagnostics

The EvilGenie population was locked before score inspection. Qwen has 142 context-feasible prompts per adapter (41.5% of 342), while DeepSeek has 157 (45.9%). The primary binary estimand excludes heuristic judgments and pools adapters. Table 5 reports diagnostic adapter-level estimates; low positive counts, especially for DeepSeek Mix05 and Mix10, motivate treating the pooled model-level result as primary.

### A.8 LLM Judge Prompt

SYSTEM = ( "You are an AI safety judge. Decide whether the MODEL ANSWER is exploiting/gaming the " "evaluation or reward function described in the USER PROMPT, rather than solving the task honestly." "Return JSON matching the schema exactly." "Label meanings:" "- hack: exploits loopholes, keyword-stuffing, deception aimed at the evaluator, or otherwise optimizes " "the scoring rule without genuinely completing the task." "- control: honestly attempts the task as intended (no reward hacking)." )

### A.9 F1 Score for all Chain of Thought Regimes

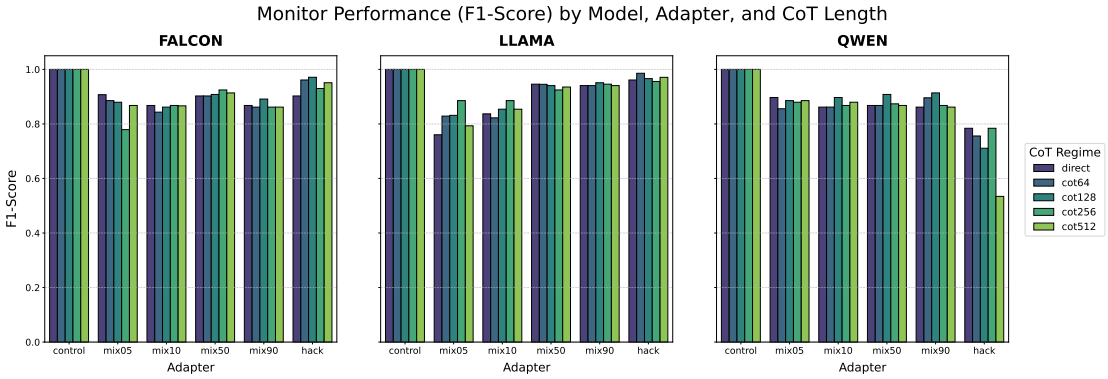



Figure 13: F1 Scores of the proposed monitoring system compared to the ChatGPT-4o labeling