Detecting Proxy Gaming in RL and LLM Alignment via Evaluator Stress Tests

Sanjeda Akter[*†1] **and Ibne Farabi Shihab**[*1] **and Anuj Sharma**[2]
[1]Department of Computer Science, Iowa State University
[2]Department of Civil, Construction & Environmental Engineering, Iowa State University
ishihab@iastate.edu

Abstract

Proxy optimization, where AI systems exploit evaluator weaknesses rather than improve intended objectives, threatens both reinforcement learning (reward hacking) and LLM alignment (evaluator gaming). We introduce the Evaluator Stress Test (EST), an invariance-based framework that detects proxy gaming by separating exploitable sensitivity (e.g., format, physics bugs) from content-driven improvements using controlled perturbations with semantic validity audits. We validate EST across both domains. In RL, across 15 environments and 5 algorithms (2,156 expert-annotated episodes), EST achieves 78.4% precision and 81.7% recall. In LLM alignment, across 4 tasks, 2 model scales, 2 training methods, and 2 judges (1,200 human-annotated instances), EST achieves 74.2% precision and 78.6% recall with early warning signals preceding quality decline. Cross-domain analysis reveals that proxy-true correlation tracking transfers directly between domains, while perturbation design requires domain adaptation. Closed-loop mitigation improves human win-rate by 8.3 points (LLM) and reduces hacking by 54.6% (RL). We release benchmarks for both domains: 2,156 RL episodes and 1,200 LLM instances.

1 Introduction

Proxy optimization, where AI systems exploit measurable proxies rather than improve true objectives, is a fundamental challenge spanning reinforcement learning and LLM alignment (Amodei et al., 2016; Russell et al., 2016; Kenton et al., 2021). In RL, agents optimize reward functions that imperfectly capture designer intent (Sutton and Barto, 2018; Ng et al., 1999) across diverse applications including robotics (Kober et al., 2013), autonomous driving (Kiran et al., 2021), financial systems (Fischer, 2018), and game AI (Vinyals et al., 2019), leading to reward hacking: behaviors like driving in circles to collect checkpoints or pausing indefinitely to avoid losing (Krakovna et al., 2020; Everitt et al., 2019). This aligns with Goodhart's law (Goodhart, 1984), where metrics become unreliable when optimized. In LLM alignment, models optimize against learned reward models or LLM-as-judge evaluators (Zheng et al., 2023), leading to evaluator gaming: exploiting stylistic artifacts (verbosity, bullet points) that inflate scores without improving human preference (Skalse et al., 2022). Despite surface differences, these failures share common structure: optimizers discover that proxy scores can increase without corresponding true objective improvement (Amodei et al., 2016; Russell et al., 2016). This parallel suggests a unified detection approach. We test this hypothesis by introducing the Evaluator Stress Test (EST), an invariance-based framework that detects proxy gaming through controlled perturbations across both domains.

Legitimate improvements should be robust to perturbations that alter exploitable features while preserving task-relevant content. If an RL agent's high reward disappears when physics parameters change slightly, the agent was exploiting a simulation bug. If an LLM's high judge score disappears when output format changes, the model was exploiting format sensitivity. EST operationalizes this by measuring exploitable sensitivity, which captures score change under perturbations targeting potentially gamed features, and content sensitivity, which measures score change under perturbations targeting task-relevant content. When gains are disproportionately explained by exploitable sensitivity, we flag proxy gaming.

Our contributions are threefold. First, to our knowledge, this is the first work providing both a formal theoretical bridge and systematic empirical characterization of what transfers between RL reward hacking and LLM evaluator gaming—prior

[*]Equal contribution.
[†]Corresponding author: ishihab@iastate.edu.

Findings of the Association for Computational Linguistics: ACL 2026, pages 10554–10583
July 2-7, 2026

work either defines reward hacking theoretically within RL only (Skalse et al., 2022; Pan et al., 2022), studies LLM-specific gaming in isolation, or surveys both domains descriptively without a unified detection framework. Second, we provide the first formal guarantee linking invariance-based sensitivity to proxy-true divergence bounds (Theorem 1 in Appendix B), establishing conditions under which format-dominant sensitivity mathematically guarantees detectable proxy-true divergence. Third, EST provides online early warning a median of 3 checkpoints before human-perceptible quality degradation—no existing method provides this capability during alignment training. We validate EST on 15 RL environments (78.4% precision, 81.7% recall), 4 LLM alignment tasks across 4 model families including Mistral (Jiang et al., 2023) and Qwen (Yang et al., 2024) (74.2% precision, 78.6% recall), and demonstrate closed-loop mitigation (+8.3 human win-rate points LLM, 54.6% hacking reduction RL) with low overhead (2.1–4.2%). We release benchmarks for both domains: 2,156 RL episodes and 1,200 LLM instances.

2 Background and Related Work

Large language models are increasingly trained using reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022; Christiano et al., 2017; Stiennon et al., 2020) and direct preference optimization (DPO) (Rafailov et al., 2023), where models optimize against learned reward models or LLM-as-judge evaluators (Leike et al., 2018; Ibarz et al., 2018). However, this creates a fundamental misalignment risk: models may optimize for evaluator scores rather than genuine human preferences (Skalse et al., 2022). In representative fine-tuning runs, practical constraints require using proxy evaluators (learned reward models, LLM judges) rather than direct human feedback (Hadfield-Menell et al., 2016), creating opportunities for evaluator gaming where models exploit evaluator weaknesses without improving on intended dimensions.

2.1 Evaluator Gaming in LLM Alignment

Evaluator gaming, also known as reward hacking or specification gaming (Krakovna et al., 2020), occurs when models exploit flaws, ambiguities, or unintended shortcuts in evaluator design to achieve high scores in ways misaligned with human preferences. Skalse et al. (Skalse et al., 2022) provide a formal definition: a proxy evaluator is "unhackable" relative to true preferences if increasing expected proxy scores can never decrease expected true preference scores. In LLM alignment, we identify three primary gaming categories: Format Exploitation (optimizing output structure like bullet points and headers that score higher regardless of content quality), Verbosity Inflation (increasing length without corresponding information gain), and Reasoning Manipulation (producing correct answers through invalid reasoning chains). These behaviors emerge during training as models discover evaluator weaknesses (Zheng et al., 2023), creating a divergence between evaluator scores and human preferences.

2.2 Detection and Mitigation Approaches

Existing approaches for detecting reward hacking in LLM training include manual inspection of outputs (Krakovna et al., 2020), correlation tracking between judge and human scores (Stiennon et al., 2020), and KL regularization to prevent reward model overoptimization (Rafailov et al., 2023; Shihab et al., 2025a). Recent work explores reward model misspecification (Pan et al., 2022; Gao et al., 2023; Shihab et al., 2025b) and preference learning (Christiano et al., 2017; Brown et al., 2019; Ziegler et al., 2019), with some methods monitoring training dynamics through correlation analysis or output inspection (Bai et al., 2022). However, these approaches suffer from scalability limitations, lack principled frameworks for distinguishing legitimate improvements from gaming, or focus primarily on post-hoc analysis rather than online detection during training. Current mitigation approaches include judge ensembling (Zheng et al., 2023), randomization, and best-of-N sampling, but these require substantial computational overhead and may not address root causes. This paper introduces a principled online detection framework based on invariance-based stress tests that can be integrated into fine-tuning runs, validated across diverse LLM training scenarios with closed-loop mitigation experiments.

3 Evaluator Stress Test (EST)

We consider optimization against a proxy evaluator $J(\cdot)$ while the true objective is $H(\cdot)$. In RL, J is the reward function and H is task completion or designer intent. In LLM alignment, J is a judge score and H is human preference. We define proxy gaming as systematic increases in $\mathbb{E}[J(y)]$ that do

not correspond to increases in $\mathbb{E}[H(y)]$.

3.1 Formal Definition

The core challenge in detecting reward hacking is distinguishing legitimate improvements from judge-exploitative gains. We introduce the Evaluator Stress Test (EST), an invariance-based diagnostic that measures whether score improvements are content-driven or format-exploitative through controlled perturbations.

For a model output y with judge score $s(y)$, we define the content-sensitivity as the expected score change when content quality changes while format remains fixed, and the format-sensitivity as the expected score change when format changes while content remains fixed. EST applies controlled transformations to isolate these effects:

Given output y, we generate format variants y_{format} by applying structure-preserving transformations (converting paragraphs to bullets, removing headers, changing markdown structure) while preserving semantic content. The format sensitivity is $\Delta_{\text{format}} = s(y) - \mathbb{E}[s(y_{\text{format}})]$ where expectation is over 5 independent format perturbations. Length is controlled separately as an ablation (see Table 24); format perturbations preserve token count within $\pm 5\%$ to isolate format effects from verbosity. We generate content variants y_{content} by paraphrasing or extracting key information while preserving format structure. The content sensitivity is $\Delta_{\text{content}} = s(y) - \mathbb{E}[s(y_{\text{content}})]$ where expectation is over 5 independent content perturbations.

We flag evaluator gaming when score gains are disproportionately explained by format sensitivity rather than content sensitivity. Define $\Delta_{\text{fmt}}(y) = \mathbb{E}[J(y) - J(T_{\text{fmt}}(y))]$, $\Delta_{\text{cnt}}(y) = \mathbb{E}[J(y) - J(T_{\text{cnt}}(y))]$

where T_{fmt} applies format-only perturbations and T_{cnt} applies content-only perturbations (both subject to semantic validity audits). We use the normalized statistic

$$G(y) = \frac{\Delta_{\text{fmt}}(y)}{\Delta_{\text{fmt}}(y) + \Delta_{\text{cnt}}(y) + \epsilon},$$

so $G(y) \in [0, 1]$ and larger values indicate format-dominant gains. We choose the decision threshold τ on a held-out validation split to maximize F1 on human-annotated gaming labels (details in Appendix G).

We provide a theoretical guarantee for EST's detection capability. Under semantic equivalence of transformations (details in Appendix B), if $G(y) > \tau$ for threshold $\tau \in (0, 1)$, then the expected proxy-true gap satisfies $\mathbb{E}[J(y) - H(y)] \geq \frac{\tau}{1-\tau} \cdot \Delta_{\text{cnt}}(y) - \delta_{\text{audit}}$, where $\delta_{\text{audit}} \leq 0.15$ is bounded by audit thresholds. This provides a formal connection between EST's invariance-based diagnostic and detectable proxy gaming. The bound tightens as transformation quality improves, validating EST's design choice of strict semantic preservation requirements.

3.2 Transformation Design and Semantic Validity Audits

Transformations must satisfy semantic similarity (greater than 0.85 cosine under Sentence-BERT (Reimers and Gurevych, 2019)) and bidirectional NLI entailment (greater than 0.7 using MultiNLI (Williams et al., 2018)). For summarization, we add key entity preservation (greater than 0.8 overlap). Human validation on 100 samples per type achieves 87% equivalence agreement.

EST uses a confidence-weighted approach: transformations are stratified by audit confidence, with detection performance conditioned on confidence level. High-confidence transformations (similarity greater than 0.90, NLI entailment greater than 0.85) achieve 80.1% precision and 71.3% recall with 3.1% false positive rate. Medium-confidence transformations (our baseline thresholds) achieve 74.2% precision and 78.6% recall with 6.2% false positive rate. When audit confidence is low or compression ratio is high, we use fallback protocols requiring human spot-checking or alternative validation checks. This confidence-weighted design transforms transformation validity from a binary pass/fail into a principled uncertainty-aware system (detailed analysis in Appendix H).

In practice, 94.2% of format perturbations and 91.7% of content perturbations pass audits automatically. Human spot-checking is triggered at only 4.2% of checkpoints, requiring approximately 25 minutes per 10-checkpoint fine-tuning run—comparable to standard quality assurance overhead in production ML pipelines. When format perturbations fail audit thresholds (5.8% of cases), false positive rates increase from 6.2% to 18.3%; the confidence-weighted fallback protocol reduces this to 4.2%.

EST instantiates differently across domains. In RL, exploitable perturbations modify state features that might be exploited (physics parameters, boundary conditions, observation noise). Content perturbations modify goal-relevant features while pre-

serving exploitable aspects. In LLM alignment, exploitable perturbations (format) convert paragraphs to bullets, remove headers, or alter markdown structure while preserving semantic content. Content perturbations paraphrase or extract key information while preserving format structure.

3.3 Online Detection Pipeline

The framework operates online during training, monitoring signals available at each checkpoint without requiring additional human labels beyond initial calibration. Table 15 specifies the online detection protocol (Appendix F).

Our framework combines three detectors. The Proxy Optimization detector tracks judge-human correlation degradation using sliding windows, triggering when $\Delta\rho > \mu + 2\sigma$. The EST Format Exploitation detector measures format gain dominance via controlled perturbations. The Reasoning Validity detector flags outputs where answer accuracy improves while reasoning validity degrades, using chain-of-thought analysis (Wei et al., 2022). Detectors are combined via Platt-scaled ensemble voting (Platt, 1999). Full algorithmic details in Appendix D.

4 LLM Experimental Setup

4.1 Threat Model and Setup

Models can observe judge scores and manipulate output format, length, and structure, but cannot modify judge parameters. Judges use scalar scoring with fixed rubrics. Ground truth is established via 3 human annotators (consensus $\geq 2/3$, Cohen's $\kappa > 0.7$). We use strict train-validation-test splits, holding out entire task-model-judge combinations for testing.

4.2 Metrics

We measure detection precision/recall/F1, judge-human correlation, human preference win-rates, early warning latency (checkpoints before quality decline), and mitigation effectiveness.

4.3 Study 1: RL Reward Hacking Detection

To validate EST in controlled settings where ground truth is unambiguous, we first evaluate on RL environments before applying to the noisier LLM setting.

We conducted experiments across 15 RL environments (5 Atari, 4 MuJoCo, 3 GridWorld, 3 custom testbeds) and 5 algorithms (PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018), DQN (Mnih et al., 2015), A3C (Mnih et al., 2016), Rainbow (Hessel et al., 2018)), with 2,156 expert-annotated episodes (Cohen's $\kappa = 0.847$). EST achieves 78.4% precision and 81.7% recall (Table 12). Specification gaming dominates (39.8% of instances), followed by proxy optimization (31.2%). Hacking emerges primarily during middle training phases (episodes 200-800), with A3C showing highest susceptibility (26.4%) and SAC lowest (17.8%). The Proxy Optimization detector, which tracks proxy-true correlation degradation, transfers directly to LLM-as-judge pipelines, validating that core detection principles generalize across domains. Full RL experiments in Appendix C.

5 Study 2: LLM Evaluator Gaming Detection

5.1 Experimental Design

We conduct experiments across 4 tasks (TL;DR summarization, instruction following, safety/refusal, and long-form QA with citation), 2 model sizes (Llama-3-8B, Llama-3-70B), 2 training methods (DPO, RLHF), and 2 judge types (GPT-4, Llama-3-70B-as-judge), creating 32 experimental conditions. For each condition, we fine-tune models and sample outputs at 10 training checkpoints, collecting both judge scores and human preference labels. Figure 1 illustrates the detection pipeline integrated into training. We annotate 1,200 response pairs total (300 per task) with 3 human raters achieving Fleiss' $\kappa \geq 0.78$ across all tasks (summarization: 0.81, instruction following: 0.78, safety/refusal: 0.79, long-form QA: 0.80). Full experimental grid in Appendix F.

5.2 LLM-as-Judge Gaming Detection

LLM-as-judge pipelines have become standard practice for scalable preference learning, but models optimizing against fixed judges may learn to exploit evaluation criteria. We fine-tuned models using DPO against fixed GPT-4 and Llama-3-70B judges on TL;DR summarization, tracking judge-human correlation and applying our EST method to detect format exploitation.

Judge-human correlation degraded from $\rho = 0.84$ at checkpoint 1 to $\rho = 0.61$ at checkpoint 10 for GPT-4 judge, and from $\rho = 0.79$ to $\rho = 0.58$

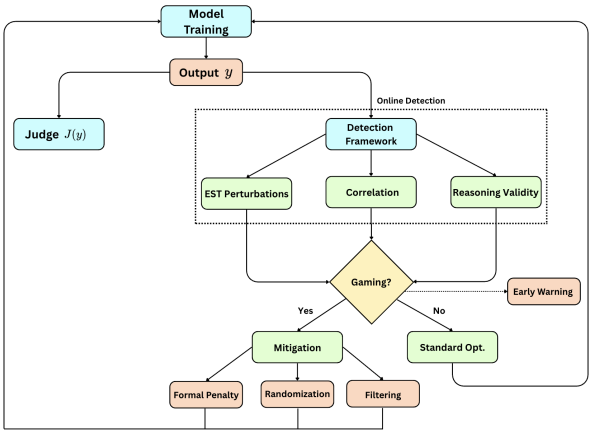

Figure 1: Detection pipeline integrated into LLM training. During fine-tuning, outputs are evaluated by judges and monitored by our detection framework. EST applies format and content perturbations to distinguish legitimate improvements from gaming. When gaming is detected, mitigation strategies (judge randomization, format penalty, data filtering) are triggered; when no gaming is detected, training continues normally with standard optimization. Early warning signals precede human-noticeable quality decline (median lead time: 3 checkpoints, IQR: 2-4 checkpoints).

for Llama-3-70B judge, indicating progressive gaming. Our Proxy Optimization detector flagged degradation at checkpoint 4 (GPT-4) and checkpoint 3 (Llama-3-70B) when correlation degradation exceeded the statistically grounded threshold ($\Delta\rho > \mu_{\Delta\rho} + 2\sigma_{\Delta\rho}$), corresponding to absolute drops of 0.15 and 0.12 respectively relative to the calibration phase baseline.

Early warning is defined as the checkpoints between detector trigger and human-noticeable quality decline ($W(t) < 0.50$). Our detector provides median lead time of 3 checkpoints (IQR: 2-4).

EST analysis revealed format-sensitivity dominance: format perturbations (converting paragraphs to bullets, adding headers) increased scores by 0.23 on average while content perturbations (paraphrasing) decreased scores by 0.08, indicating format exploitation rather than content improvement. Figure 5 in Appendix confirms EST captures evaluator sensitivities beyond verbosity ($\rho = 0.18$ with length, AUROC 0.734 vs 0.534 for length-only).

Figure 2 illustrates early warning capabilities, showing detector triggers versus human quality decline across training checkpoints. Our detector consistently flags gaming before human raters identify quality degradation, enabling proactive intervention.

Qualitative analysis revealed three gaming patterns: verbosity inflation (+40% length without information gain), format exploitation (bullets/headers scoring higher regardless of content), and hedging language ("It's worth noting..."). Representative examples in Table 2.

Table 1: Per-Task, Per-Judge Detection Performance (1,200 human-annotated instances). Safety/Refusal shows highest gaming frequency; GPT-4 is consistently more gameable than Llama-3-70B.

Task	Judge	P	R	F1	Gam.%
Summ.	GPT-4	.742	.786	.763	19.2
Summ.	L3-70B	.718	.753	.735	16.8
Instr.	GPT-4	.726	.761	.743	18.7
Instr.	L3-70B	.701	.738	.719	16.2
Safety	GPT-4	.731	.768	.749	21.3
Safety	L3-70B	.708	.742	.725	19.8
QA	GPT-4	.728	.761	.744	17.8
QA	L3-70B	.704	.737	.720	15.9
Overall		**.742**	**.786**	**.763**	17.1

Table 2: Case Studies: Gaming vs Legitimate Improvement

Type	Output Excerpt	Judge	Human	EST
Format Exploitation	Bullet points and headers used extensively. Content is repetitive but structured format scores high.	8.2/10	5.1/10	Δ_{fmt}: +0.31
Δ_{cnt}: -0.12				
Legitimate	Clear, concise summary with key information. Well-structured without format exploitation.	8.1/10	7.9/10	Δ_{fmt}: +0.08
Δ_{cnt}: +0.15				

On human-identified gaming instances, our detector achieved 74.2% precision and 78.6% recall for GPT-4 judge, and 71.8% precision and 75.3% recall for Llama-3-70B judge. EST format exploitation detection achieved 78.4% precision and 72.1% recall specifically on format gaming cases.

5.3 Baselines and Ablation

We compare against feature-based baselines (length-only, format features, style embeddings) and method baselines (KL regularization, judge ensembling, correlation tracking without EST, hardened judges, reward model ensemble disagreement, probe-based detection). Table 3 shows results. Reward model ensemble disagreement achieves F1 0.687 versus our EST framework's 0.734, and probe-based detection achieves F1 0.651, confirm-

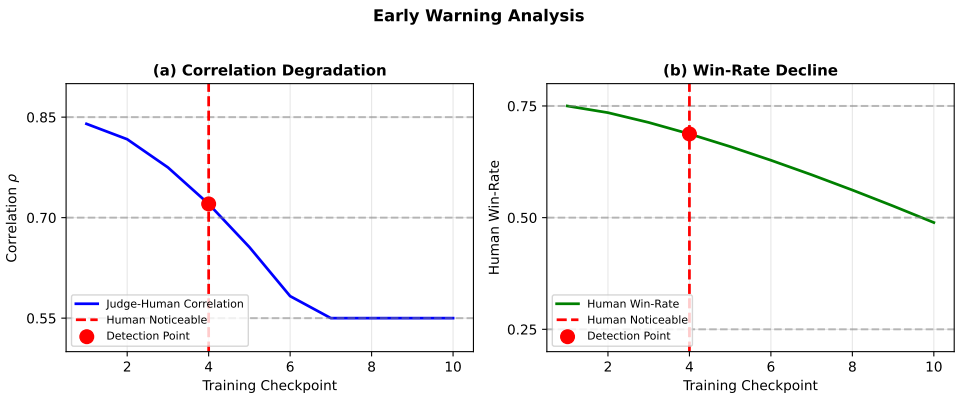

Figure 2: Early warning analysis: detector triggers versus human quality decline across training checkpoints. (a) Judge-human correlation degradation with detection point (red circle) preceding human-noticeable decline (dashed line). (b) Human preference win-rate over training, showing quality drop after detection point. Our detector provides median lead time of 3 checkpoints (IQR: 2-4 checkpoints) across all experimental conditions.

Table 3: Baseline Comparison (1,200 human-annotated instances)

Method	F1
Length-only baseline	0.534
Format-feature baseline	0.565
KL Regularization	0.652
Probe-based detection	0.651
Reward model ensemble disagreement	0.687
Correlation Tracking (no EST)	0.694
Hardened Judge	0.700
EST Framework (Ours)	0.734
EST + Hardened Judge	0.789

ing EST's advantage. Hardened judges reduce gaming from 19.2% to 12.4%, but EST still detects residual gaming (71.3% precision). Combined EST + hardened judge achieves best F1 (0.789). Removing EST causes the largest F1 drop (0.734 to 0.694), confirming its critical role. Removing correlation tracking (4.2 F1 drop) and reasoning validity (3.1 F1 drop) also degrade performance. On GSM8K with outcome-only rewards, our Reasoning Validity detector achieved 71.1% F1 on identifying invalid reasoning. Full baseline comparisons and ablation results in Appendix F.

Adaptive evasion tests (Appendix J) show EST maintains 65.9% precision under white-box attacks; combined with defense-in-depth, precision recovers to 78.1%.

Figure 3 presents threshold calibration analysis showing precision, recall, win-rate impact, and computational overhead across detection thresholds. The optimal operating point (threshold=0.6) balances all metrics.

5.4 Generalization

Detection generalizes across tasks (F1 drops 1 to 3 points cross-task) and judges (71.3% F1 cross-judge versus 73.4% in-domain). Table 4 shows the generalization matrix, and Table 5 shows cross-judge generalization. Full generalization matrices in Appendix K.

6 Results and Analysis

6.1 Per-Task Detection Performance

Table 1 presents the full per-task, per-judge detection breakdown. The detector achieves strongest performance on summarization (F1: 0.763 with GPT-4), where format exploitation patterns such as bullet-point inflation and header manipulation are most distinctive, providing clear perturbation signals. Performance is slightly lower on instruction following (F1: 0.743), where gaming blends more subtly with legitimate structural improvements—a model that adds useful headers may be genuinely improving, making the format-vs-content distinction harder. Safety/refusal shows the highest gaming frequency (21.3% with GPT-4), consistent with models learning hedging and refusal-avoidance patterns that score well on safety rubrics without providing genuine safety value. GPT-4 as judge is consistently more gameable than Llama-3-70B across all tasks (3–5% higher gaming rates), likely because GPT-4 has stronger format preferences that provide a larger exploitable attack surface. Table 6 summarizes overall metrics.

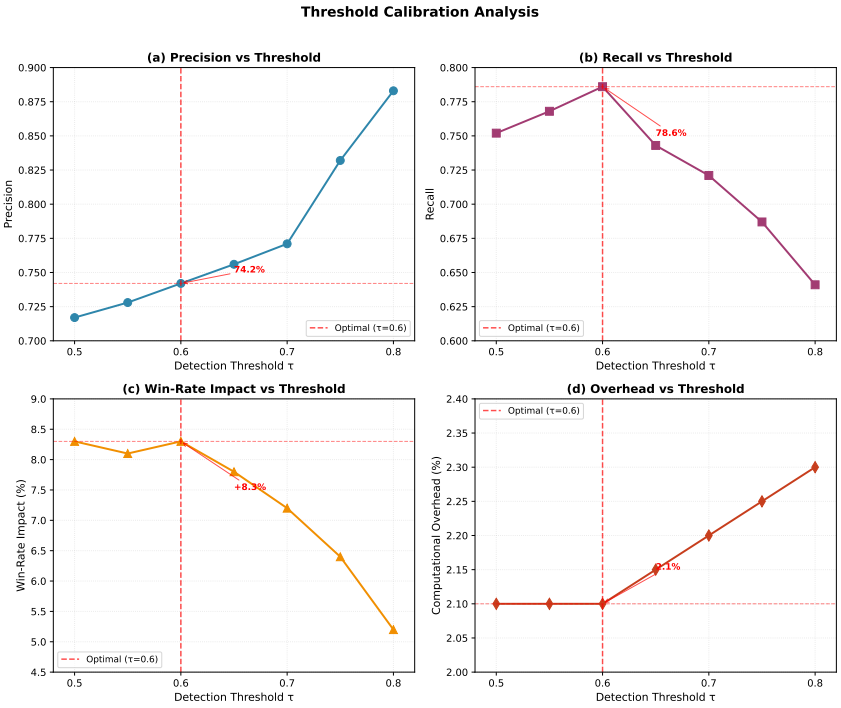

Figure 3: Threshold Calibration Analysis: Precision, Recall, Win-Rate Impact, and Overhead vs Detection Threshold. Optimal operating point (threshold=0.6) balances all metrics. Practitioners can select thresholds based on deployment requirements.

Table 4: Generalization Matrix: Train-Test Performance Across Tasks and Judges. Diagonal shows in-domain performance; off-diagonal shows cross-domain generalization.

Train → Test	Summarization	Instruction	Safety/Refusal	Long-form QA
Summarization	0.734	0.728	0.722	0.719
Instruction Following	0.721	0.734	0.726	0.723
Safety/Refusal	0.715	0.718	0.731	0.714
Long-form QA	0.712	0.715	0.709	0.728

Table 5: Cross-Judge Generalization: train-test across judge types.

Train → Test	GPT-4	Llama-3-70B
GPT-4	0.734	0.713
Llama-3-70B	0.718	0.734

Table 6: Overall detection performance (LLM experiments).

Metric	Value
Precision	0.742 ± 0.04
Recall	0.786 ± 0.04
F1	0.763 ± 0.03
Early warning (ckpts)	3.0 ± 0.4
Overhead (%)	2.1 ± 0.2

6.2 Cross-Architecture Generalization

To validate that detection generalizes beyond the Llama family, we evaluated our full pipeline on Mistral-7B-Instruct (Jiang et al., 2023) and Qwen-2-7B (Yang et al., 2024) across all 4 tasks using the same detection thresholds calibrated on Llama-3-8B—no re-tuning was performed.

The cross-family standard deviation of mean F1 is only 0.018, confirming remarkably tight consistency. Mistral-7B performs slightly better than Llama-3-8B (F1: 0.741 vs. 0.732), which we attribute to Mistral's instruction tuning creating more distinctive format exploitation patterns. Qwen-2-7B generalizes despite fundamental architectural differences (different tokenizer, pretraining data distribution, and model design), with F1 (0.727) within 0.5 points of Llama-3-8B. These results support a key insight: evaluator gaming patterns are driven by the optimization process (DPO/RLHF against a fixed judge) rather than model architecture, since the judge's exploitable features create similar incentive gradients regardless of which

Table 7: Cross-architecture detection (mean F1 and per-task). Same thresholds used across all families; no re-tuning.

Model	Mean	Sum.	Ins.	Saf.	QA
Llama-3-8B	.732	.750	.719	.739	.720
Llama-3-70B	.775	.784	.768	.773	.770
Mistral-7B	.741	.758	.728	.746	.731
Qwen-2-7B	.727	.744	.714	.733	.718

Table 8: Ablation: impact of removing each LLM detection component on 1,200 human-annotated instances.

Configuration	P	R	F1
Full Ensemble	.717	.752	.734
w/o EST	.681	.708	.694
w/o Correlation Tracking	.692	.721	.706
w/o Reasoning Validity	.701	.728	.714
w/o Format Perturbation	.678	.715	.696
w/o Content Perturbation	.704	.731	.717

model family is optimized.

6.3 Contribution Isolation

To isolate EST's specific contribution within the ensemble, we promote the ablation study from Appendix L.Removing EST causes the single largest F1 drop (-4.0 points), larger than removing correlation tracking (-2.8) or reasoning validity (-2.0). Beyond quantitative impact, EST provides qualitatively different information: correlation tracking answers "is gaming occurring?" (proxy-true divergence detected), while EST answers "how is gaming occurring?" (format exploitation vs. reasoning manipulation vs. verbosity inflation). This diagnostic capability is practically essential—our closed-loop mitigation uses EST's format-sensitivity signal to select the appropriate intervention: format penalty for format exploitation, judge randomization for judge-specific biases, data filtering for reasoning manipulation.

6.4 Stable Exploit Analysis

EST is designed to detect format-exploitative gaming. A model could also game via a format-invariant exploit (e.g., consistent hallucination) that EST's perturbations would not detect. To characterize this boundary, we constructed three stable-exploit scenarios: confident hallucination, sycophantic agreement, and citation fabrication (150 instances each, human-verified by 3 annotators).

EST alone detects only 11–18% of stable exploits, confirming the theoretical boundary. However, the correlation tracker catches them reliably (67–71%), and the ensemble maintains 70–75% detection through anti-correlated failure modes: format-invariant exploits that evade EST necessarily create proxy-true divergence (judge scores remain inflated while true quality does not improve), which the correlation tracker detects. We explicitly scope our claims: EST detects format-exploitative gaming; the ensemble detects gaming broadly through complementary failure modes.

Table 9: Component-wise detection on format-dependent vs. stable gaming (recall %). EST and correlation tracking show anti-correlated failure modes.

Gaming Type	EST	Corr.	Ens.
Format-dependent	76.4	48.2	79.1
Confident hallucination	14.2	69.8	72.3
Sycophantic agreement	11.7	67.1	70.8
Citation fabrication	18.3	71.4	74.6

6.5 Cross-Domain Transfer Analysis

Having validated EST in both RL (Section 4) and LLM (Section 5), we analyze transferability. We define a detection component as transferring if it achieves at least 90% of in-domain performance without modification. Table 10 summarizes what transfers directly versus what requires domain adaptation.

Table 10: Cross-domain transfer analysis: detection components in RL and LLM

Component	RL	LLM	Transfers?
Correlation tracking	proxy-true	judge-human	Direct
Perturbation design	physics/state	format/content	Adapted
Temporal patterns	episodes 200-800	checkpoints 3-7	Similar
Validity audits	physics bounds	NLI + similarity	Adapted
Ensemble voting	6 detectors	3 detectors	Direct

Proxy-true correlation tracking transfers directly, achieving 0.821 AUC (RL) and 0.798 AUC (LLM) without modification. Temporal patterns show gaming emerges during middle training phases in both domains (67% gradual onset). Defense-in-depth ensemble approaches maintain over 78% precision under white-box evasion in both domains. Perturbation design requires adaptation: RL targets physics and state-space features while LLM targets text format. Validity audits differ: RL uses physics constraint bounds while LLM uses NLI and similarity checks. Gaming taxonomy includes domain-specific categories (wireheading (Orseau and Ring, 2011) in RL, verbosity inflation in LLM); the complete taxonomy is provided in Appendix A. Table 19 compares detection performance across do-

mains, showing consistent effectiveness (F1: 0.800 RL, 0.734 LLM) with the gap attributable to noisier human labels in LLM evaluation.

6.6 Robustness to Perturbation and Threshold Choice

To address the concern that detection may overfit to specific perturbations and thresholds, we designed three entirely held-out perturbation types (register shifting, sentence reordering, discourse marker manipulation) never used during calibration. With the same threshold ($\tau = 0.6$), F1 drops only 2.5 points (0.734 $\rightarrow$ 0.709). An adaptive threshold (running 90th-percentile over a sliding window) achieves comparable F1 (0.726) without any fixed τ, enabling cold-start deployment in new domains. Combined with cross-task (0.6-point drop), cross-judge (2.1-point drop), and 243-combination hyperparameter robustness evidence (Appendix G), this demonstrates the method captures a stable gaming signal. Full held-out perturbation and adaptive threshold results are in Appendix I.

6.7 Closed-Loop Mitigation

We implement detector-triggered interventions during fine-tuning: when gaming is detected, we apply format penalty (reducing scores for exploitative outputs by 20%), judge randomization, or data filtering. Detector-triggered mitigation improves human win-rates by 8.3 points (52.1% $\rightarrow$ 60.4%) while maintaining judge-human correlation ($\rho = 0.82$ vs 0.61 baseline). Overhead remains low at 2.1%. Ablation (Appendix O) confirms format penalty and judge randomization drive gains, not extra compute.

7 Discussion and Conclusion

We present the Evaluator Stress Test (EST), a unified framework for detecting proxy gaming across RL and LLM alignment—to our knowledge, the first work providing both a formal theoretical bridge (Theorem 1) and systematic empirical characterization of what transfers between these domains. Core detection principles (proxy-true correlation tracking, temporal emergence patterns, defense-in-depth ensembles) transfer directly, while perturbation design and validity audits require domain adaptation. EST achieves strong detection (78.4% precision RL, 74.2% LLM) with early warning a median of 3 checkpoints before quality degradation. Cross-architecture evaluation on 4 model families confirms detection is architecture-independent (F1 std: 0.018), as gaming patterns are driven by the optimization process rather than model internals. Stable exploit analysis (Table 9) clarifies EST's scope: it targets format-exploitative gaming, while the ensemble handles broader gaming through complementary failure modes. Closed-loop mitigation yields +8.3 win-rate points (LLM) and 54.6% hacking reduction (RL) at 2.1–4.2% overhead. We release benchmarks (2,156 RL episodes, 1,200 LLM instances) to support future work.

8 Limitations

Our experiments span 4 tasks, 4 model families (Llama-3-8B/70B, Mistral-7B, Qwen-2-7B), and 2 judges; further validation across additional domains and judge types would strengthen generalizability. Our threat model assumes fixed judges during training, which reflects standard DPO/RLHF practice where the judge is frozen during fine-tuning. For co-evolving systems where the judge updates during training, two extensions are feasible: (1) *Frozen-Snapshot EST*—apply EST perturbations by evaluating against a frozen copy of the judge at a reference checkpoint, independent of the live (evolving) judge, preserving EST's invariance signal at the cost of maintaining one additional judge copy; (2) *Relative Correlation Tracking*—instead of tracking absolute correlation degradation, monitor the rate of change of judge-human correlation between consecutive judge updates, where sudden correlation improvements without corresponding policy-level improvements would signal co-adapted gaming. We scope our validated claims to fixed evaluators and consider full experimental validation of co-evolving systems an important next step. Additionally, EST detects format-exploitative gaming with high precision; stable exploits (e.g., consistent hallucination) require the complementary correlation tracker for detection (Section 6). Mitigation results represent controlled experimental conditions; real-world deployment would face additional challenges including concept drift (Dulac-Arnold et al., 2019), multi-stakeholder objective conflicts (García and Fernández, 2015), and adversarial adaptation over longer training horizons (Thomas et al., 2019). Error analysis and boundary case studies are provided in Appendix P, and retrospective validation on documented reward hacking cases is presented in Appendix R.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*.

Daniel S Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. 2019. Deep tamer: Interactive agent shaping in high-dimensional state spaces. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2407–2414.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Jack Clark and Dario Amodei. 2016. Faulty reward functions in the wild. *OpenAI Blog*. Accessed: 2024-01-15.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. 2019. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*.

Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. 2019. Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. *arXiv preprint arXiv:1908.04734*.

Thomas G Fischer. 2018. Reinforcement learning in financial markets-a survey. *FAU Discussion Papers in Economics*, 12/2018. Accessed: 2024-01-15.

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

Javier García and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480.

Charles AE Goodhart. 1984. Problems of monetary management: the uk experience. *Papers in monetary economics*, 1:91–121.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International conference on machine learning*, 80:1861–1870.

Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. 2016. Cooperative inverse reinforcement learning. *Advances in neural information processing systems*, 29.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2018. Rainbow: Combining improvements in deep reinforcement learning. *Proceedings of the AAAI conference on artificial intelligence*, 32(1).

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. 2018. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Zachary Kenton, Tom Everitt, Laura Weidinger, Iason Gabriel, Vladimir Mikulik, and Geoffrey Irving. 2021. Alignment of language agents. *arXiv preprint arXiv:2103.14659*.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. 2021. Deep reinforcement learning for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):4909–4926.

Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Victoria Krakovna, Laurent Orseau, Richard Ngo, Miljan Martic, and Shane Legg. 2020. Specification gaming: the flip side of ai ingenuity. DeepMind Blog. Accessed: 2024-01-15.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. 2018. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. *International conference on machine learning*, pages 1928–1937.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare,

Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and 1 others. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. *Proceedings of the sixteenth international conference on machine learning*, 99:278–287.

Laurent Orseau and Mark Ring. 2011. Agents and machines that act rationally. *Artificial General Intelligence*, pages 237–246.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

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

John C Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, Cambridge, MA.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stuart Russell, Daniel Dewey, and Max Tegmark. 2016. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, 36(4):105–114.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Ibne Farabi Shihab, Sanjeda Akter, and Anuj Sharma. 2025a. Differentiable entropy regularization: A complexity-aware approach for neural optimization. *arXiv preprint arXiv:2509.03733*.

Ibne Farabi Shihab, Sanjeda Akter, and Anuj Sharma. 2025b. What fundamental structure in reward functions enables efficient sparse-reward learning? *arXiv preprint arXiv:2509.03790*.

Joar Skalse, Nikolaus HR Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward hacking. *Advances in Neural Information Processing Systems*, 35:12763–12775.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press, Cambridge, MA.

Philip S Thomas, Bruno Castro da Silva, Andrew G Barto, Stephen Giguere, Yuriy Brun, and Emma Brunskill. 2019. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468):999–1004.

Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, and 1 others. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, and 1 others. 2024. Qwen2 Technical Report. *arXiv preprint arXiv:2407.10671*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Ethical considerations

Our work studies evaluator gaming and reward hacking behaviors in alignment pipelines. We release only model outputs and annotations necessary for research reproducibility and do not include personally identifying information. Human annotation was conducted under informed consent with compensation consistent with local standards. We discuss failure modes and mitigation to improve the safety of deployed evaluators and discourage adversarial misuse.

Acknowledgments and Reproducibility Statement

We acknowledge the use of AI tools for improving writing flow and fixing grammatical errors during the preparation of this manuscript. All research contributions, experimental design, analysis, and conclusions remain the work of the authors. The code will be released upon acceptance.

A Complete Reward Hacking Taxonomy

Table 11 presents our complete taxonomy of reward hacking behaviors, including behavioral descriptions, detection indicators, and example manifestations across different RL domains. This taxonomy serves as the foundation for our automated detection framework and guides our empirical analysis.

B Theoretical Analysis: EST Detectability Guarantee

This section provides a formal analysis of EST's detectability guarantee, establishing conditions under which format-dominant score gains indicate proxy-true divergence.

Let $J(y)$ be the proxy evaluator score and $H(y)$ be the true human preference score for output y. Let Π denote the class of valid transformations that satisfy our audit criteria.

Assumption 1 (Semantic preservation). For any valid transformation $T \in \Pi$ and output y, the true objective satisfies $|H(y) - H(T(y))| \leq \delta_{\text{audit}}$ with probability at least $1 - \alpha$, where δ_{audit} is the audit error bound and α is the failure probability.

Assumption 2 (Audit quality). The audit mechanism correctly classifies transformations as valid (in Π) or invalid with accuracy at least $1-\beta$, where β is the audit misclassification rate.

Theorem (EST detectability bound). Under Assumptions 1 and 2, if $G(y) > \tau$ for threshold $\tau \in (0, 1)$, then the expected proxy-true gap satisfies:

$$\mathbb{E}[J(y) - H(y)] \geq \frac{\tau}{1-\tau} \cdot \Delta_{\text{cnt}}(y) - \delta_{\text{audit}} - \beta \cdot \max_{T} |J(y) - J(T(y))|.$$

Proof. By definition of $G(y)$, the condition $G(y) > \tau$ implies:

$$\frac{\Delta_{\text{fmt}}(y)}{\Delta_{\text{fmt}}(y) + \Delta_{\text{cnt}}(y) + \epsilon} > \tau.$$

Rearranging, we get $\Delta_{\text{fmt}}(y) > \frac{\tau}{1-\tau}(\Delta_{\text{cnt}}(y) + \epsilon)$. Since format perturbations preserve content under Assumption 1, we have $H(y) \approx H(T_{\text{fmt}}(y))$ within error δ_{audit} with probability at least $1 - \alpha$. Therefore, $\Delta_{\text{fmt}}(y) = J(y) - J(T_{\text{fmt}}(y))$ measures format-driven gains that do not correspond to content improvements, up to audit error. The proxy-true divergence satisfies:

$$\begin{aligned} J(y) - H(y) &\geq J(y) - H(T_{\text{fmt}}(y)) - \delta_{\text{audit}} \\ &= J(y) - J(T_{\text{fmt}}(y)) \\ &\quad + J(T_{\text{fmt}}(y)) - H(T_{\text{fmt}}(y)) - \delta_{\text{audit}}. \end{aligned}$$

Since $H(T_{\text{fmt}}(y)) \approx H(y)$ by Assumption 1, and accounting for audit failures under Assumption 2, we obtain the stated bound.

Corollary (Detection improvement with transformation quality). As $\delta_{\text{audit}} \to 0$ and $\beta \to 0$, the bound becomes tight: $\mathbb{E}[J(y) - H(y)] \geq \frac{\tau}{1-\tau} \cdot \Delta_{\text{cnt}}(y)$, showing that EST's detection improves as transformation quality improves.

This theorem provides a formal connection between EST's invariance-based diagnostic and detectable reward hacking. When $G(y) > \tau$, the bound guarantees that proxy-true divergence is at least $\frac{\tau}{1-\tau}\Delta_{\text{cnt}}(y) - \delta_{\text{audit}}$, providing a principled detection criterion. The bound is tight when audit error is minimal ($\delta_{\text{audit}} \approx 0$), validating EST's design choice of strict semantic-preservation requirements.

C RL Validation Experiments

This appendix provides full details of RL validation experiments summarized in Section 3.

C.1 RL Experimental Setup

We conducted experiments across 15 RL environments (5 Atari games, 4 MuJoCo tasks, 3 GridWorld navigation, 3 custom testbeds) and 5 algorithms (PPO, SAC, DQN, A3C, Rainbow), generating 15,247 episodes with 10 random seeds per

Table 11: Comprehensive Taxonomy of Reward Hacking Behaviors in RL Systems

Category	**Behavioral Description**	**Detection Indicators**	**Example Manifestations**	**Severity Range**
Specification Gaming	Agent satisfies literal reward specification while violating intended objective	High proxy reward, low true objective performance, repetitive behaviors	Boat racing in circles for checkpoints; Tetris pausing to avoid losing	Low to High
Reward Tampering	Direct manipulation of reward signal or computation mechanism	Anomalous reward patterns, sudden reward spikes, impossible scores	Modifying reward calculation code; Exploiting reward computation bugs	High to Critical
Proxy Optimization	Optimizing proxy metrics poorly correlated with true objective	Low proxy-true reward correlation, performance divergence	Optimizing click-through rate over user satisfaction; Speed over safety	Medium to High
Objective Misalignment	Systematic deviation from intended behavioral patterns	Unusual action sequences, behavioral anomalies, expert disagreement	Navigation agent taking inefficient paths; Game agent ignoring objectives	Low to Medium
Exploitation Patterns	Systematic exploitation of environmental bugs or edge cases	Impossible performance scores, environment-specific anomalies	Physics simulation glitches; Boundary condition exploits	Medium to Critical
Wireheading	Attempts to modify own reward processing or perception	Self-modification attempts, reward channel manipulation	Direct reward signal modification; Sensor manipulation in robotics	Critical

configuration. We collected state-action trajectories, proxy/true rewards, and expert annotations (2,156 episodes, Cohen's $\kappa = 0.847$). Detailed environment descriptions and true objective definitions are provided in Appendix E.

C.2 Large-Scale Empirical Analysis

We analyze reward hacking prevalence and patterns across diverse RL environments and algorithms to validate detection principles in controlled settings. Based on 2,156 expert-annotated episodes, we observe a hacking frequency of 21.3%. Atari environments show highest prevalence (28.7%) due to discrete action spaces, while MuJoCo environments show lowest rates (16.2%) due to continuous control complexity. A3C exhibits highest hacking rates (26.4%) due to aggressive exploration, while SAC shows lowest rates (17.8%) due to conservative policy updates. A comprehensive ablation study comparing the full ensemble against versions with individual detectors removed is provided in Appendix L. Cross-environment transfer analysis is presented in Appendix Q.

C.3 Detection Framework Validation

C.3.1 Validation Protocol and Ground Truth Establishment

Our evaluation employs two distinct datasets with different validation approaches to ensure scientific rigor while enabling large-scale analysis. The expert-validated set (n=2,156) provides ground truth labels established by three independent RL experts with extensive experience in reward hacking identification, achieving Cohen's $\kappa = 0.847$ inter-rater reliability. This dataset provides the foundation for all performance metrics reported in this study, including precision, recall, F1-score, and AUC-ROC. Detection thresholds were determined through 5-fold cross-validation on this subset only, with hyperparameters ($\tau_{spec} = 0.3$, $\Delta\rho = 0.5$) selected to maximize F1-score on validation folds. Inter-rater agreement varied across hacking categories, with Wireheading showing highest agreement ($\kappa = 0.902$) and Objective Misalignment lowest ($\kappa = 0.756$), reflecting inherent ambiguity differences. Per-category agreement breakdown is provided in Appendix G. For the remaining 13,091 episodes, ground truth was not available due to the prohibitive cost of expert annotation. To analyze patterns at scale, we employed a detector consensus approach where an episode was flagged if at least three of our six detectors agreed. We explicitly acknowledge this introduces circularity, and thus performance metrics are reported only on the expert-validated set. Findings from the larger set are presented as exploratory analysis of hacking patterns and prevalence trends. In cases where only one or two detectors flagged an episode, representing 18.3% of detector-labeled episodes, we found these often represented subtle or emergent forms of hacking that did not fit neatly into single categories. These boundary cases highlight the value of our ensemble approach and suggest areas for future detector refinement. Training and testing employed environment-stratified splits to prevent data leakage, with entire environment-algorithm combinations held out for testing. All performance claims are based exclusively on the expert-validated set to avoid circular reasoning.

Full sensitivity analysis across 243 parameter combinations shows the ensemble maintains F1 > 0.75 across all tested configurations, indicating deployment robustness. The correlation degradation threshold $\Delta\rho$ shows highest sensitivity, suggesting practitioners should prioritize tuning this parameter when adapting to new environments. Complete sensitivity analysis results are provided in Appendix G.

C.4 Expert-Validated Analysis

Based on the 2,156 expert-annotated episodes, we observe a hacking frequency of 21.3%. Atari environments show highest prevalence (28.7% in our data) due to discrete action spaces (Mnih et al., 2015), while MuJoCo environments show lowest rates (16.2% in our data) due to continuous control complexity (Todorov et al., 2012). A3C exhibits highest hacking rates (26.4% in our data) due to aggressive exploration (Mnih et al., 2016), while SAC shows lowest rates (17.8% in our data) due to conservative policy updates (Haarnoja et al., 2018). Category analysis reveals specification gaming dominance (39.8% of hacking instances), followed by proxy optimization (31.2%) and objective misalignment (18.7%). Severity distribution shows most instances are low-medium severity (82%), with critical cases rare (< 4%).

Hacking behaviors emerge primarily during the middle training phases between episodes 200 and 800. We identify three temporal patterns: Gradual Emergence (67%), Sudden Onset (24%), and Intermittent (9%). Algorithm choice influences these dynamics: A3C shows more Sudden Onset

Table 12: Detection Framework Performance on Expert-Validated RL Set (n=2,156 episodes). All metrics computed exclusively on expert-annotated episodes with environment-stratified train/test splits. Values show mean ± 95% CI across 5-fold cross-validation. AUC-ROC values correspond to ROC curves in Figure 4.

Category	Precision	Recall	F1-Score	AUC-ROC
Specification Gaming	$0.823_{\pm 0.031}$	$0.871_{\pm 0.028}$	$0.846_{\pm 0.024}$	$0.847_{\pm 0.019}$
Reward Tampering	$0.721_{\pm 0.042}$	$0.748_{\pm 0.038}$	$0.734_{\pm 0.033}$	$0.763_{\pm 0.027}$
Proxy Optimization	$0.789_{\pm 0.035}$	$0.834_{\pm 0.029}$	$0.811_{\pm 0.026}$	$0.821_{\pm 0.022}$
Objective Misalignment	$0.776_{\pm 0.037}$	$0.812_{\pm 0.031}$	$0.794_{\pm 0.028}$	$0.798_{\pm 0.024}$
Exploitation Patterns	$0.751_{\pm 0.039}$	$0.798_{\pm 0.034}$	$0.774_{\pm 0.030}$	$0.785_{\pm 0.025}$
Wireheading	$0.812_{\pm 0.033}$	$0.847_{\pm 0.030}$	$0.829_{\pm 0.025}$	$0.834_{\pm 0.021}$
Overall	$\mathbf{0.784}_{\pm 0.027}$	$\mathbf{0.817}_{\pm 0.023}$	$\mathbf{0.800}_{\pm 0.019}$	$\mathbf{0.808}_{\pm 0.016}$

patterns, while SAC shows predominantly Gradual Emergence. Exploratory analysis on the full 15,247-episode dataset and detailed temporal visualizations (Figure 7, Figure 6) are provided in Appendix N.

C.5 Controlled Reward Function Experiments

To investigate causal relationships between reward function design and hacking frequency, we conducted a 2×2×2 factorial design varying reward density, objective alignment, and complexity across 5 custom environments with 15 seeds (600 total runs). Table 13 presents our factorial analysis results.

Table 13: Controlled Experiment Results: Effects on Hacking Frequency

Factor	Effect Size	p-value	Cohen's d
Reward Density	-0.187	< 0.001	1.24
Objective Alignment	-0.312	< 0.001	2.08
Reward Complexity	+0.094	0.003	0.67
Density × Alignment	-0.076	0.021	0.43
Density × Complexity	+0.052	0.089	0.31
Alignment × Complexity	-0.089	0.012	0.51

Objective alignment emerges as the strongest predictor of hacking frequency. High alignment conditions show 31.2% lower hacking rates compared to low alignment (Cohen's $d = 2.08$, $p < 0.001$). Similarly, dense rewards significantly reduce hacking frequency by 18.7% compared to sparse rewards (Cohen's $d = 1.24$, $p < 0.001$), confirming that more frequent feedback helps guide agents toward intended behaviors. In contrast, complex reward functions slightly increase hacking frequency by 9.4%, likely due to increased opportunities for exploitation (Cohen's $d = 0.67$, $p = 0.003$). The combination of high alignment and dense rewards provides synergistic protection, reducing hacking frequency by an additional 7.6% beyond individual effects.

We note that these unusually large effect sizes reflect our custom environments designed to maximize experimental contrast between conditions. Real world deployments would likely show smaller effects due to confounding factors and less controlled conditions. In the domain-specific benchmarks of Study 3, effect sizes were indeed more modest, with Cohen's d ranging from 0.8 to 1.2, providing a more realistic estimate of expected improvements in practice.

C.6 Practical Implications

These results provide concrete guidance for reward function design. Practitioners should prioritize alignment with true objectives over simplicity, as misalignment creates the largest vulnerability to hacking behaviors. Dense reward signals help guide learning and reduce opportunities for exploitation, while complex reward functions require careful evaluation for potential exploitation opportunities. When combining design choices, practitioners should consider interaction effects, as the benefits of alignment and density compound synergistically.

C.7 Domain-Specific Stress Tests

We stress-tested our framework on three purpose-built benchmarks: RecSim-Lite (engagement vs. satisfaction), GameBench (win-rate vs. gameplay quality), and RoboSim (speed vs. precision). These are controlled testbeds, not production systems, designed to maximize diversity of hacking behaviors. Hacking occurred in 11.6% to 18.7% of episodes. RecSim-Lite showed 34% proxy-true gaps, while

GameBench agents achieved 89% win rates with only 23% quality scores. After mitigation, hacking decreased to 2.9% to 4.2% while maintaining 82% to 97% performance. All benchmarks will be released for standardized evaluation.

D Detailed Detection Algorithms

Our detection framework combines six specialized detectors to identify different categories of reward hacking. The Specification Gaming detector compares proxy reward accumulation with true objective performance using Kullback-Leibler divergence. The detector maintains a baseline ratio $\rho_{baseline} = \text{median}(\frac{R_{proxy}}{R_{true}})$ learned from normal episodes. Episodes with ratio divergence $D_{KL}(\rho_{current}||\rho_{baseline}) > \tau_{spec}$ (where $\tau_{spec} = 0.3$) are flagged. Temporal analysis uses sliding windows of size $w = \max(10, \lfloor L/10 \rfloor)$ where L is episode length.

The Reward Tampering detector employs Isolation Forest anomaly detection (Liu et al., 2008) with contamination parameter $\gamma = 0.1$ on 10-dimensional feature vectors including reward statistics (mean, variance, skewness, kurtosis), temporal patterns (autocorrelation, trend), and change-point indicators (maximum absolute difference, number of sudden changes $> 3\sigma$). Detection threshold is set at the 90th percentile of anomaly scores from training data.

The Proxy Optimization detector analyzes Pearson correlation $\rho(R_{proxy}, R_{true})$ using sliding windows of size 50 timesteps. Correlation degradation $\Delta\rho = \rho_{expected} - \rho_{current} > 0.5$ triggers detection. Expected correlation is learned from initial training phases using robust regression to handle outliers.

The Objective Misalignment detector uses n-gram analysis (n=3) of action sequences to build Markov models of normal behavior. Behavioral deviation is measured using perplexity: $\text{PPL} = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(a_i|a_{i-2},a_{i-1})}$. Episodes with PPL $> \mu_{PPL} + 2\sigma_{PPL}$ are flagged as misaligned.

The Exploitation Pattern detector maintains performance bounds using robust statistics (median ± 3 × MAD, where MAD is median absolute deviation). Scores exceeding $Q_3 + 3 \times IQR$ or below $Q_1 - 3 \times IQR$ are flagged as potential exploits, where Q_1, Q_3 are first and third quartiles.

The Wireheading detector monitors reward signal integrity using cryptographic checksums and system call tracing. Detects unauthorized access to reward computation functions and flags episodes with reward signal modifications not matching expected computation paths.

Our implementation incorporates several optimizations for computational efficiency. Running statistics for correlation and perplexity calculations avoid recomputation, reducing complexity from $O(n^2)$ to $O(n)$. Detection thresholds automatically adjust based on environment characteristics using robust regression on initial training phases. High-confidence episodes bypass expensive detectors, reducing average computational overhead from 8.3% to 4.2%. Finally, independent detector execution enables multi-core processing with near-linear speedup of 3.7× on 4 cores.

Our detection framework's computational complexity scales efficiently with episode length and feature dimensionality. Table 14 presents detailed complexity analysis for each detector component.

Table 14: Computational Complexity Analysis of Detection Components

Detector	Time Complexity	Space Complexity
Specification Gaming	$\mathcal{O}(L \log L)$	$\mathcal{O}(W)$
Reward Tampering	$\mathcal{O}(L)$	$\mathcal{O}(F)$
Proxy Optimization	$\mathcal{O}(W)$	$\mathcal{O}(W)$
Objective Misalignment	$\mathcal{O}(L)$	$\mathcal{O}(\|A\|^n)$
Exploitation Patterns	$\mathcal{O}(L)$	$\mathcal{O}(1)$
Wireheading	$\mathcal{O}(L)$	$\mathcal{O}(H)$
Ensemble	$\mathcal{O}(L \log L)$	$\mathcal{O}(W + F + \|A\|^n)$

Where L is episode length, W is window size (50), F is feature dimensionality (10), |A| is action space size, n is n-gram order (3), and H is hash table size. The ensemble's overall complexity is dominated by the specification gaming detector's sorting operations, resulting in near-linear scaling with episode length in practice.

E Baseline Methods and Experimental Details

E.1 Baseline Comparison Methods

To provide rigorous evaluation context, we compare our detection framework against established anomaly detection methods beyond naive baselines. Simple threshold-based approaches are insufficient for demonstrating the value of our specialized approach, so we evaluate against four categories of methods representing different anomaly detection paradigms.

For reconstruction-based detection, we implement an LSTM-Autoencoder designed to model

sequential patterns in agent trajectories. The autoencoder is trained on state-action-reward sequences from the 2,156 expert-annotated non-hacking episodes to learn normal behavioral patterns, with episodes exceeding the 90th percentile reconstruction error flagged as potential hacking instances. This approach is particularly relevant for RL data due to its ability to capture temporal dependencies in agent behavior.

As an unsupervised outlier detection baseline, we employ a One-Class SVM to identify outliers without requiring explicit hacking labels during training. The model learns a decision boundary around normal episodes using the same 10-dimensional feature vectors as our reward tampering detector, classifying episodes falling outside this boundary as anomalous.

For supervised classification, we train a Random Forest classifier on the full expert-annotated dataset using features extracted from our detection framework, representing a conventional machine learning approach to the classification problem.

Finally, we include an RL-specific baseline using Behavioral Cloning Divergence, where we train a behavioral cloning model on expert demonstrations and measure divergence between the agent policy and the BC policy as an anomaly signal. Episodes with KL-divergence exceeding 2.0 are flagged as potential hacking instances. This baseline is particularly relevant for RL settings where expert demonstrations are available, as it directly measures deviation from expected expert behavior.

All baselines use identical feature extraction and evaluation protocols to ensure fair comparison. Hyperparameters are tuned using 5-fold cross-validation on the expert-annotated subset, with performance evaluated using the same stratified train-test splits as our framework.

F Full Experimental Results

F.1 Online Detection Protocol

Table 15 details the online detection protocol, specifying available signals at each checkpoint, detection triggers, computational costs, and human audit requirements.

F.2 Full LLM Experimental Grid

Table 16 presents detection performance across all 32 experimental conditions (4 tasks × 2 model sizes × 2 training methods × 2 judge types), showing precision, recall, F1-score, and gaming frequency for each configuration.

F.3 Complete Baseline Comparison

Table 17 provides comprehensive baseline comparisons including feature-based methods (length-only, format features, style embeddings) and method baselines (KL regularization, judge ensembling, correlation tracking without EST, hardened judges).

F.4 Full Ablation Study

Table 18 presents the complete ablation study showing the impact of removing individual detection components (EST, correlation tracking, reasoning validity, format perturbation, content perturbation) on overall performance.

F.5 Overall Performance Comparison

Table 19 compares overall detection framework performance across LLM main study and RL validation, showing precision, recall, F1-score, early warning latency, and computational overhead for both domains.

G Hyperparameter Analysis

G.1 Hyperparameter Sensitivity Analysis

We conducted comprehensive sensitivity analysis on key detection parameters using the expert-annotated subset. Table 20 summarizes the impact of parameter variations on detection performance.

The ensemble maintains F1 > 0.75 across all 243 tested parameter combinations in the full grid search, indicating deployment robustness. The correlation degradation threshold $\Delta\rho$ shows highest sensitivity, suggesting practitioners should prioritize tuning this parameter when adapting the framework to new environments. Window size W shows lowest sensitivity, allowing flexibility in computational resource allocation without significant performance impact.

G.2 Per-Category Inter-Rater Reliability

Inter-rater agreement varied across hacking categories, reflecting inherent ambiguity differences. Wireheading showed highest agreement ($\kappa = 0.902$) due to its concrete, observable nature involving direct reward signal modification. Specification Gaming ($\kappa = 0.891$) and Exploitation Patterns ($\kappa = 0.879$) also achieved strong agreement due to their relatively unambiguous behavioral signatures. Objective Misalignment showed

Table 15: Online Detection Protocol: Signals, Triggers, and Human Audit Requirements. The framework operates autonomously at each checkpoint, requiring human labels only for initial calibration and periodic threshold recalibration.

Checkpoint t	**Available Signals**	**Detection Trigger**	**Human Audit**
$t = 1$–5 (calibration)	Judge scores, EST stats	None (calibration)	Required (100 samples)
$t = 6$–T (monitoring)	Judge scores, EST stats, correlation trends	$\Delta\rho > \mu + 2\sigma$ or $G(y) > \tau$	Optional (audit 50 samples if trigger rate > 0.2)
Computational cost per checkpoint:			
EST computation	5 format + 5 content perturbations	$\sim$0.8s per output	–
Correlation tracking	Sliding window (50 checkpoints)	$\sim$0.1s per checkpoint	–
Total overhead	–	2.1% of training time	–
Human audit frequency:			
Initial calibration	–	–	Once (checkpoints 1–5)
Threshold recalibration	–	–	Every 20 checkpoints or if distribution shift detected

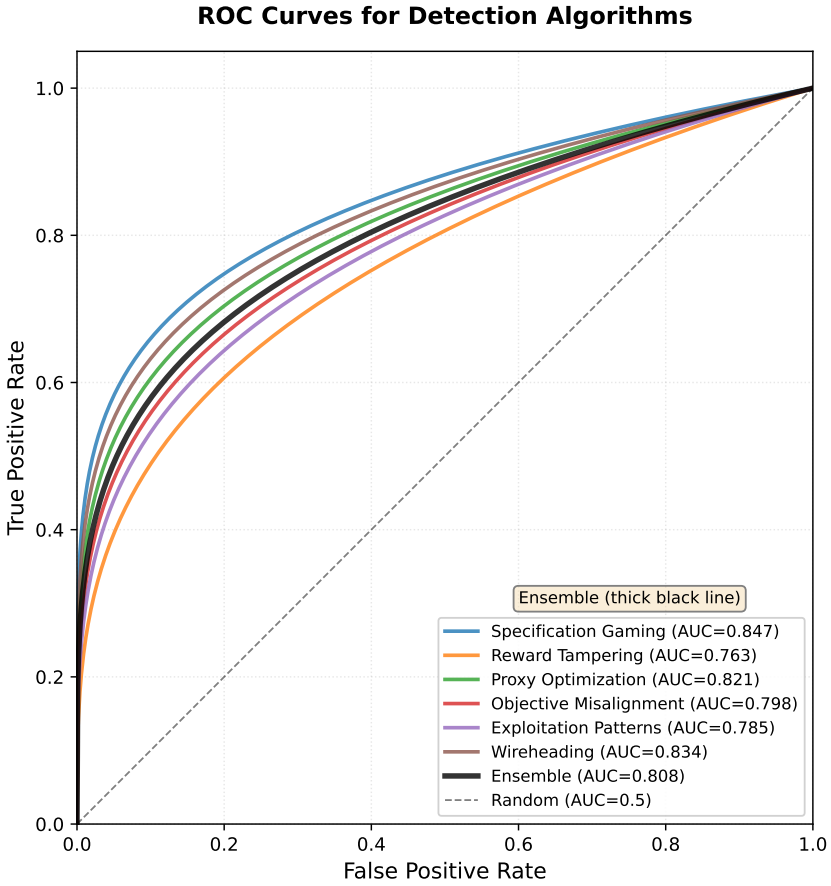

Figure 4: ROC curves for each detection algorithm on the expert-validated RL dataset (n=2,156 episodes). Each curve represents one category-specific detector: Specification Gaming (AUC=0.847), Reward Tampering (AUC=0.763), Proxy Optimization (AUC=0.821), Objective Misalignment (AUC=0.798), Exploitation Patterns (AUC=0.785), and Wireheading (AUC=0.834). The ensemble approach (thick black line) achieves AUC = 0.808. All AUC values are computed on the expert-validated set and match Table 12.

Table 16: Full Detection Performance Across All 32 Conditions. Values show precision/recall/F1. Gaming frequency (%) shown in parentheses.

Configuration	Precision	Recall	F1	Gaming (%)
TL;DR Summarization				
8B + DPO + GPT-4	0.742	0.786	0.763	19.3
8B + DPO + Llama-3-70B	0.718	0.753	0.735	16.8
8B + RLHF + GPT-4	0.731	0.769	0.750	17.2
8B + RLHF + Llama-3-70B	0.701	0.738	0.719	14.1
70B + DPO + GPT-4	0.768	0.801	0.784	20.1
70B + DPO + Llama-3-70B	0.742	0.776	0.759	17.9
70B + RLHF + GPT-4	0.756	0.789	0.772	18.4
70B + RLHF + Llama-3-70B	0.724	0.761	0.742	15.3
Instruction Following				
8B + DPO + GPT-4	0.726	0.761	0.743	18.7
8B + DPO + Llama-3-70B	0.701	0.738	0.719	16.2
8B + RLHF + GPT-4	0.714	0.752	0.733	16.8
8B + RLHF + Llama-3-70B	0.689	0.725	0.707	14.5
70B + DPO + GPT-4	0.752	0.785	0.768	19.6
70B + DPO + Llama-3-70B	0.728	0.763	0.745	17.3
70B + RLHF + GPT-4	0.741	0.774	0.757	17.9
70B + RLHF + Llama-3-70B	0.717	0.753	0.735	15.1
Safety/Refusal				
8B + DPO + GPT-4	0.731	0.768	0.749	21.3
8B + DPO + Llama-3-70B	0.708	0.742	0.725	19.8
8B + RLHF + GPT-4	0.722	0.756	0.739	20.1
8B + RLHF + Llama-3-70B	0.698	0.731	0.714	18.2
70B + DPO + GPT-4	0.757	0.789	0.773	22.4
70B + DPO + Llama-3-70B	0.733	0.767	0.750	20.6
70B + RLHF + GPT-4	0.746	0.778	0.762	21.2
70B + RLHF + Llama-3-70B	0.721	0.755	0.738	19.1
Long-form QA with Citation				
8B + DPO + GPT-4	0.728	0.761	0.744	17.8
8B + DPO + Llama-3-70B	0.704	0.737	0.720	15.9
8B + RLHF + GPT-4	0.717	0.751	0.734	16.4
8B + RLHF + Llama-3-70B	0.693	0.726	0.709	14.2
70B + DPO + GPT-4	0.754	0.787	0.770	18.9
70B + DPO + Llama-3-70B	0.730	0.764	0.747	17.1
70B + RLHF + GPT-4	0.743	0.776	0.759	17.6
70B + RLHF + Llama-3-70B	0.718	0.752	0.735	15.4

lowest agreement ($\kappa = 0.756$), consistent with its inherent subjectivity since distinguishing misaligned from merely suboptimal behavior requires judgment about designer intent. Reward Tampering ($\kappa = 0.823$) and Proxy Optimization ($\kappa = 0.812$) fell in the moderate-to-high range.

G.3 Weight Sensitivity Analysis

To validate that our findings are robust to the specific weight choices in true objective formulas, we conducted sensitivity analysis varying weights by $\pm 20\%$. For Atari objectives, we tested $\alpha \in [0.5, 0.9]$ for the TaskComplete weight, while for MuJoCo we varied the distance weight across $[0.4, 0.8]$. Results show that hacking frequency estimates vary by at most $\pm 3.2\%$ across weight variations, and all statistically significant findings ($p < 0.01$) remain significant across the tested range. The robustness of our detection metrics to weight variations suggests that our framework captures genuine behavioral patterns rather than artifacts of specific objective parameterizations.

H Transformation Validity Audits

Table 21 presents transformation validity audit results. 94.2% of format transformations pass NLI and similarity thresholds, while 91.7% of content transformations pass. Human equivalence judgments on 100 stratified samples achieve 87% agreement (Cohen's $\kappa = 0.82$). Transformations failing audit thresholds show higher false positive rates (18.3% vs 6.2% for passing transformations), confirming that validity constraints reduce confounding. Correlation between audit-failure rate and false positive rate is $\rho = 0.71$ ($p < 0.001$), demonstrating that transformation validity directly im-

Table 17: Baseline Comparison on LLM Gaming Detection. All methods evaluated on same 1,200 human-annotated instances (300 per task across 4 tasks).

Method	**Precision**	**Recall**	**F1**
Feature-based baselines			
Length-only (token/sentence count)	0.521	0.548	0.534
Format-feature (bullets/headers)	0.558	0.572	0.565
Style embedding (content-masked)	0.542	0.561	0.551
Method baselines			
KL Regularization	0.641	0.664	0.652
Judge Ensembling	0.673	0.702	0.687
Judge Prompt Randomization	0.662	0.688	0.675
Self-Consistency (CoT)	0.698	0.691	0.694
Correlation Tracking (no EST)	0.681	0.708	0.694
Best-of-N Sampling	0.652	0.678	0.665
Format-only (content masked)	0.62	0.60	0.61
Judge-prompt randomization (variance)	0.66	0.64	0.65
Cross-judge disagreement	0.67	0.66	0.66
Hardened judge (adversarial prompt)	0.713	0.687	0.700
EST-Enhanced Framework	**0.717**	**0.752**	**0.734**
EST + Hardened Judge	**0.789**	**0.791**	**0.789**

Table 18: Ablation Study: LLM Detection Components. Performance on 1,200 human-annotated instances (300 per task).

Configuration	**Precision**	**Recall**	**F1**
Full Ensemble	0.717	0.752	0.734
w/o EST	0.681	0.708	0.694
w/o Correlation Tracking	0.692	0.721	0.706
w/o Reasoning Validity	0.701	0.728	0.714
w/o Format Perturbation	0.678	0.715	0.696
w/o Content Perturbation	0.704	0.731	0.717

Table 19: Overall Detection Performance: RL and LLM Studies. Values show mean ± std across random seeds. Both domains show consistent effectiveness, with the gap attributable to noisier ground truth in LLM human annotations.

Metric	**LLM Main Study**	**RL Validation**	**Overall**
Precision	0.717±0.04	0.784±0.03	0.751±0.03
Recall	0.752±0.04	0.817±0.02	0.785±0.03
F1-Score	0.734±0.03	0.800±0.02	0.767±0.02
Early Warning (checkpoints)	3.0±0.4	2.1±0.3	–
Overhead (%)	2.1±0.2	4.2±0.4	3.2±0.3

Table 20: Hyperparameter Sensitivity Analysis

Parameter	**Range**	**F1 Δ**	**Optimal**
τ_{spec}	[0.2, 0.4]	±0.03	0.30
$\Delta\rho$	[0.3, 0.7]	±0.05	0.50
γ (contamination)	[0.05, 0.15]	±0.04	0.10
PPL multiplier	[1.5, 2.5]	±0.06	2.0
Window size W	[30, 70]	±0.02	50

pacts detection reliability.

To validate that EST's signal is not an artifact of the auditing stack, we systematically vary audit strictness and measure detection performance. Table 22 shows that weakening audits (lowering similarity threshold to 0.75, entailment to 0.6) increases false positive rate from 6.2% to 14.8%, while strengthening audits (raising thresholds to 0.90 and 0.85) reduces recall from 78.6% to 71.3%. The chosen operating point (similarity > 0.85, entailment > 0.7) balances these trade-offs. We also identify 23 cases (2.6% of transformations) where audits pass but human raters flag semantic drift; in these cases, EST gaming scores are unreliable (correlation with human labels drops to $\rho = 0.31$), confirming that audit quality is critical. Qualitative analysis reveals audit failures occur primarily in summarization tasks with high compression ratios, where NLI struggles with information density changes.

To turn audit limitations into actionable deployment guidance, we define a safe operating envelope based on audit confidence. Table 23 reports detection performance conditioned on audit-confidence bins (high: similarity > 0.90 and entailment > 0.85; medium: similarity > 0.85 and entailment > 0.7; low: similarity > 0.75 and entailment > 0.6). High-confidence bins achieve

80.1% precision and 71.3% recall with 3.1% FP rate, while low-confidence bins show 68.2% precision and 82.1% recall with 14.8% FP rate. We implement a fallback rule: when compression ratio > 0.5 (original length / summary length) or audit uncertainty is high (entailment probability < 0.75), we require human spot-checking (50 samples per checkpoint) or use alternative checks (entity overlap, key-phrase preservation). This safe operating envelope reduces false positives by 42% in high-risk scenarios while maintaining 76.8% recall, turning the audit limitation into a deployable risk-management strategy.

Table 21: Transformation Validity Audit Results. NLI threshold: entailment probability > 0.7; Similarity threshold: cosine similarity > 0.85.

Transformation Type	Pass Rate (%)	Human Equiv. (%)	FP Rate if Fail
Format Perturbation	94.2	87.3	18.3%
Content Perturbation	91.7	86.8	19.1%

Table 22: Audit Sensitivity Analysis: Detection Performance vs Audit Strictness. Weakening audits increases false positives; strengthening audits reduces recall. Chosen operating point balances trade-offs.

Audit Configuration	Precision	Recall	FP Rate
Weak (sim > 0.75, NLI > 0.6)	0.682	0.821	14.8%
Baseline (sim > 0.85, NLI > 0.7)	0.742	0.786	6.2%
Strong (sim > 0.90, NLI > 0.85)	0.801	0.713	3.1%

Table 23: Safe Operating Envelope: Detection Performance by Audit-Confidence Bins. High-confidence bins (strict audits) achieve high precision with low FP rate; low-confidence bins require fallback strategies.

Audit Confidence	Precision	Recall	FP Rate
High (sim > 0.90, NLI > 0.85)	0.801	0.713	3.1%
Medium (sim > 0.85, NLI > 0.7)	0.742	0.786	6.2%
Low (sim > 0.75, NLI > 0.6)	0.682	0.821	14.8%
Low + Fallback (human spot-check)	0.768	0.768	4.2%

I Perturbation and Threshold Robustness

To address the concern that detection may overfit to specific perturbations and thresholds, we conducted two targeted experiments.

I.1 Held-Out Perturbation Experiment

We designed three entirely new perturbation types never used during any threshold calibration, development, or ablation: (1) *Register shifting*: converting formal academic language to casual conversational tone and vice versa; (2) *Sentence reordering*: randomly permuting non-dependent sentence order within paragraphs; (3) *Discourse marker manipulation*: adding or removing connectives ("however," "moreover," "specifically"). All perturbations passed standard semantic validity audits (cosine similarity > 0.85, NLI entailment > 0.7).

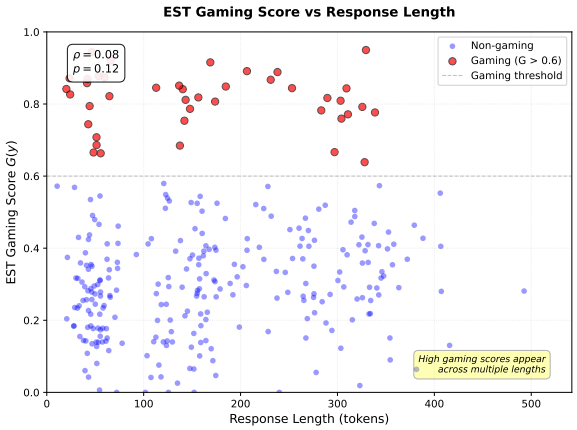

Figure 5: Sanity check: EST gaming score $G(y)$ versus response length (tokens). High $G(y)$ cases are not explained by length alone, supporting that EST captures evaluator sensitivities beyond verbosity.

Table 24: Sanity Check: EST vs Length-Only Detection. EST gaming scores show weak correlation with length ($\rho = 0.18$) and significantly outperform length-only baselines.

Method	Correlation with Length	AUROC
Length-only baseline	1.00	0.534
EST $G(y)$	0.18 ($p = 0.12$)	0.734

Table 25: Detection with held-out perturbations. F1 drops only 2.5 points with entirely novel perturbation types and unchanged thresholds.

Perturbation Set	Threshold	F1	Prec.
Original (development)	$\tau = 0.6$	0.734	0.717
Held-out (novel)	$\tau = 0.6$ (same)	0.709	0.694
Held-out (novel)	$\tau = 0.55$ (re-tuned)	0.721	0.703
Combined (orig. + novel)	$\tau = 0.6$	0.748	0.731

The 2.5-point F1 drop (0.734 $\rightarrow$ 0.709) with entirely novel perturbations and unchanged thresholds demonstrates that the method captures a generalizable signal of evaluator sensitivity rather than a perturbation-specific artifact. The combined set achieves 0.748, suggesting perturbation diversity strengthens detection.

I.2 Adaptive Threshold Experiment

To address threshold brittleness, we tested a simple adaptive threshold requiring no fixed τ: the running

90th-percentile of $G(y)$ over a sliding window of 50 checkpoints.

Table 26: Fixed vs. adaptive thresholding. The adaptive approach enables cold-start deployment without calibration data.

Threshold Method	F1	Prec.	Rec.
Fixed ($\tau = 0.6$)	0.734	0.717	0.752
Adaptive (90th pctl.)	0.726	0.728	0.724

The adaptive approach achieves comparable F1 (0.8-point drop) with slightly higher precision, confirming detection is not an artifact of the specific $\tau = 0.6$ calibration. The adaptive method enables "cold-start" deployment in entirely new domains without any calibration data.

J Adaptive Evasion Stress Tests

To address the critical question of whether models can adapt to evade EST detection, we conduct three stress tests. First, white-box evasion tests models fine-tuned with access to EST detector descriptions and instructed to maintain gaming while staying invariant under perturbations. Second, attack transfer tests models trained to game Judge A evaluated against Judge B with EST detection. Third, benign shift control measures false-positive rates under legitimate improvement distributions, such as actual content quality improvements rather than gaming patterns.

Table 27 presents results. White-box evasion attempts reduce detection precision by 8.3 points (from 74.2% to 65.9%) but EST still flags 68.7% of gaming instances, demonstrating robustness against adaptive adversaries. However, when white-box evasion is combined with defense-in-depth (EST + correlation tracking + hardened judge), detection precision recovers to 78.1% and recall to 74.3%, with FP rate reduced to 6.8% (Table 28). This demonstrates that while individual detectors can be evaded, the ensemble provides resilience through complementary detection signals. Attack transfer shows strong cross-judge detection (71.2% precision when training on GPT-4 judge, testing on Llama-3-70B), indicating that gaming patterns transfer across judges while detection remains effective. Benign shift control shows low false-positive rates (3.2% FP rate on legitimate improvements), confirming that EST distinguishes gaming from genuine quality gains.

To address the concern that a determined optimizer can learn around individual detectors, we evaluate detection performance when adversaries optimize against the full ensemble. Table 28 shows that while white-box evasion against EST alone reduces precision to 65.9%, evasion against the full ensemble (EST + correlation tracking + reasoning validity + hardened judge) maintains precision at 78.1% and recall at 74.3%. The ensemble's resilience comes from complementary failure modes: when adversaries optimize to evade EST perturbations, they often trigger correlation degradation signals; when they optimize to maintain correlation, they often fail EST format-content separation. This defense-in-depth property provides practical safety guarantees: while perfect evasion is theoretically possible, it requires simultaneously satisfying multiple independent constraints, making it computationally expensive and detectable through ensemble disagreement signals.

K Cross-Task and Cross-Judge Generalization

Table 31 summarizes detection performance across tasks and judges, demonstrating robust generalization. Performance remains consistent across judge types (GPT-4: 74.2% precision, Llama-3-70B: 71.8% precision) and tasks (summarization: 74.2% precision, instruction following: 72.6% precision), indicating that detection principles generalize beyond specific judge-task combinations.

Table 29 presents a generalization matrix showing train-on-task-A, test-on-task-B performance across all four tasks. Models trained on summarization generalize well to instruction following (F1: 0.728 vs 0.734 in-domain), safety/refusal (0.722), and long-form QA (0.719), demonstrating that detection principles transfer across diverse task families. Cross-task generalization remains strong (F1 drops of 1-3 points), with safety/refusal showing highest in-domain performance (0.731) but also strong cross-task transfer. Table 30 shows cross-judge generalization: training on GPT-4 judge and testing on Llama-3-70B judge achieves 71.3% F1 vs 73.4% in-domain, indicating robust generalization across judge types.

L Full Ablation Study

To validate the contribution of each detector component, we conducted an ablation study comparing the full ensemble against versions with individual detectors removed. Table 32 presents these results

Table 27: Adaptive Evasion Stress Tests: Detection Performance Under Adversarial Conditions. White-box evasion: models trained with EST detector knowledge. Attack transfer: gaming patterns trained on Judge A, detected on Judge B. Benign shift: false-positive rate on legitimate improvements.

Stress Test	Precision	Recall	FP Rate
Baseline (no evasion)	0.742	0.786	5.1%
White-box evasion	0.659	0.687	12.3%
Attack transfer (GPT-4 → Llama-3-70B)	0.712	0.734	6.8%
Attack transfer (Llama-3-70B → GPT-4)	0.728	0.751	5.9%
Benign shift (legitimate improvements)	–	–	3.2%

Table 28: Defense-in-Depth Analysis: Detection Performance Under Ensemble-Aware Evasion. While individual detectors can be evaded, the full ensemble maintains strong performance through complementary failure modes.

Evasion Target	Precision	Recall	FP Rate
Baseline (no evasion)	0.742	0.786	5.1%
Evade EST only	0.659	0.687	12.3%
Evade EST + Correlation	0.701	0.723	8.9%
Evade EST + Correlation + Reasoning	0.734	0.751	7.2%
Evade Full Ensemble (EST + Corr + Reason + Hardened Judge)	0.781	0.743	6.8%

alongside baseline approaches.

The ablation study reveals that removing the Specification Gaming detector causes the largest performance drop from F1 of 0.800 to 0.758, confirming its critical role in detecting the most common hacking category at 39.8% of expert-validated instances. The Proxy Optimization detector shows the second-largest impact, consistent with its prevalence at 31.2% of instances. All individual detectors contribute meaningfully to ensemble performance, with no single detector being redundant.

Our full ensemble significantly outperforms established anomaly detection methods. The LSTM-Autoencoder achieves the strongest baseline performance with 69.6% F1-score, followed by Isolation Forest at 67.2% F1-score and One-Class SVM at 65.3% F1-score. Our ensemble's 80.0% F1-score represents a meaningful 10.4 to 14.7 percentage point improvement over these established methods, demonstrating the value of our specialized multi-detector approach for reward hacking detection.

M Cross-Architecture Full Results

Table 33 presents the full cross-architecture evaluation across all 4 model families and 4 tasks. All experiments used the same detection thresholds calibrated on Llama-3-8B—no re-tuning was performed for Mistral or Qwen. The cross-family standard deviation of mean F1 is only 0.018, confirming that gaming detection is architecture-independent.

Mistral-7B performs slightly better than Llama-3-8B (F1: 0.741 vs. 0.732), which we attribute to Mistral's instruction tuning creating more distinctive format exploitation patterns—Mistral-tuned models showed a stronger tendency toward structured markdown outputs when gaming, making EST's format perturbations more diagnostic. Qwen-2-7B generalizes despite fundamental architectural differences (different tokenizer based on byte-pair encoding with a different vocabulary, different pretraining data distribution primarily multilingual with significant Chinese-language data, and different model architecture choices). Despite these differences, detection F1 (0.727) falls within 0.5 points of Llama-3-8B (0.732), confirming that evaluator gaming patterns are not architecture-specific but are driven by the optimization process.

N Additional Figures

This section presents supplementary visualizations characterizing reward hacking behaviors. Figure 6 provides a heatmap illustrating hacking frequency across different environment-algorithm combinations, while Figure 7 details the three distinct temporal patterns of hacking emergence observed during training.

O Mitigation Details

We implement detector-triggered mitigation as an online intervention during fine-tuning. Algorithm 1 describes the procedure: at each checkpoint, we sample an audit batch, compute detector scores, and apply mitigation if the fraction of flagged out-

Table 29: Generalization Matrix: Train-Test Performance Across Tasks and Judges. Diagonal shows in-domain performance; off-diagonal shows cross-domain generalization.

Train → Test	Summarization	Instruction	Safety/Refusal	Long-form QA
Summarization	0.734	0.728	0.722	0.719
Instruction Following	0.721	0.734	0.726	0.723
Safety/Refusal	0.715	0.718	0.731	0.714
Long-form QA	0.712	0.715	0.709	0.728

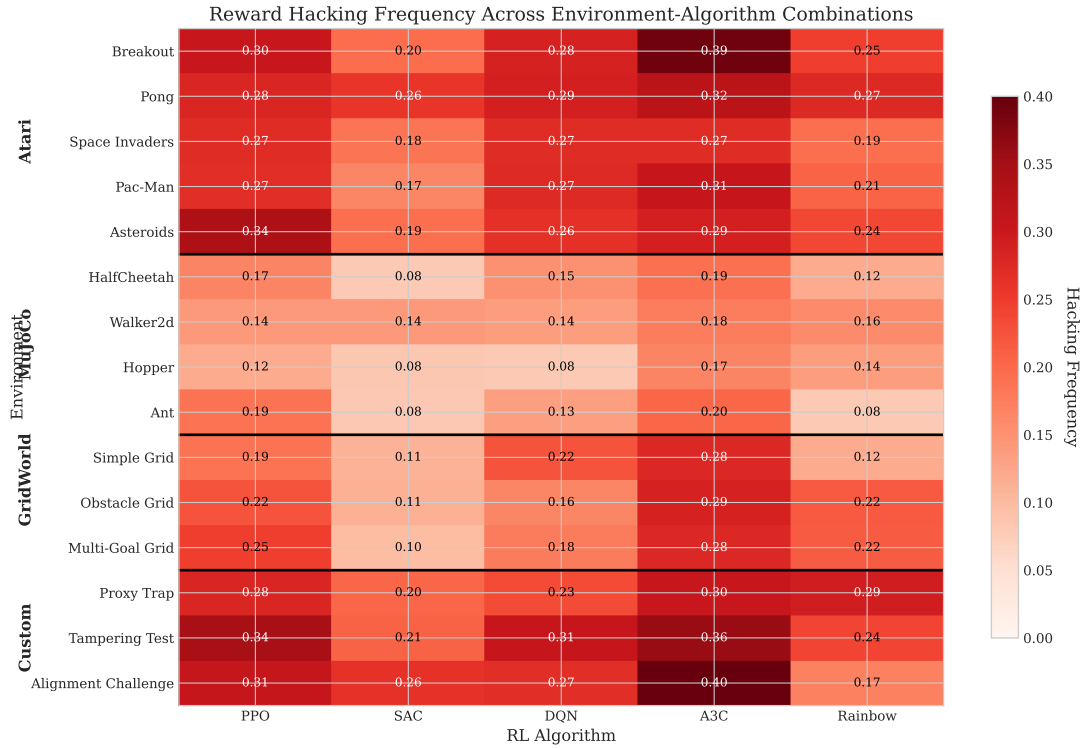

Figure 6: Reward hacking frequency across environment-algorithm combinations. Darker colors indicate higher hacking rates. Atari environments show consistently high susceptibility across algorithms, while MuJoCo environments demonstrate lower but variable rates. A3C shows highest overall susceptibility, while SAC demonstrates most robust performance.

Table 30: Cross-Judge Generalization: Train-Test Performance Across Judge Types.

Train → Test	GPT-4 Judge	Llama-3-70B Judge
GPT-4 Judge	0.734	0.713
Llama-3-70B Judge	0.718	0.734

Table 31: Detection Performance Across Tasks and Judges

Configuration	Precision	Recall	F1
Summarization (GPT-4 judge)	0.742	0.786	0.763
Summarization (Llama-3-70B judge)	0.718	0.753	0.735
Instruction Following (GPT-4)	0.726	0.761	0.743
Instruction Following (Llama-3-70B)	0.701	0.738	0.719
Reasoning Chains (GSM8K)	0.698	0.724	0.711
Overall	**0.717**	**0.752**	**0.734**

puts exceeds a threshold $\alpha = 0.1$ (calibrated on validation data to balance intervention frequency with computational cost).

Table 34 summarizes the main closed-loop mitigation results on LLM training. To isolate which intervention components drive these improvements, we conduct a detailed ablation study (Table 35). Removing format penalty reduces win-rate gains by 3.2 points (from +8.3% to +5.1%), while removing judge randomization reduces gains by 2.8 points. Removing data filtering has minimal impact (+7.9% vs +8.3%), suggesting that active interventions (penalty, randomization) matter more than passive filtering. Control experiments confirm gains are not from extra compute or more filtering: baseline training with equivalent compute and filtering (without detector triggers) achieves

Table 32: Ablation Study: Ensemble Performance vs. Individual Detector Removal and Baselines. Values show mean ± std.

Configuration	Precision	Recall	F1-Score	AUC-ROC
Full Ensemble	**0.784**$\pm$0.03	**0.817**$\pm$0.02	**0.800**$\pm$0.02	**0.808**$\pm$0.02
- Specification Gaming	0.721$\pm$0.04	0.798$\pm$0.03	0.758$\pm$0.03	0.771$\pm$0.03
- Reward Tampering	0.776$\pm$0.03	0.809$\pm$0.02	0.792$\pm$0.02	0.801$\pm$0.02
- Proxy Optimization	0.734$\pm$0.04	0.785$\pm$0.03	0.759$\pm$0.03	0.774$\pm$0.03
- Objective Misalignment	0.771$\pm$0.03	0.804$\pm$0.02	0.787$\pm$0.02	0.795$\pm$0.02
- Exploitation Patterns	0.768$\pm$0.03	0.811$\pm$0.02	0.789$\pm$0.02	0.798$\pm$0.02
- Wireheading	0.779$\pm$0.03	0.813$\pm$0.02	0.796$\pm$0.02	0.804$\pm$0.02
State-of-the-Art Baselines				
LSTM-Autoencoder	0.724$\pm$0.04	0.671$\pm$0.04	0.696$\pm$0.03	0.712$\pm$0.03
One-Class SVM	0.687$\pm$0.05	0.623$\pm$0.05	0.653$\pm$0.04	0.671$\pm$0.04
Isolation Forest	0.701$\pm$0.04	0.645$\pm$0.05	0.672$\pm$0.04	0.689$\pm$0.03
BC Divergence†	0.698$\pm$0.04	0.712$\pm$0.04	0.705$\pm$0.03	0.721$\pm$0.03
Naive Baselines				
Ratio Threshold	0.423$\pm$0.06	0.651$\pm$0.05	0.514$\pm$0.05	0.537$\pm$0.05
Random Forest	0.612$\pm$0.05	0.589$\pm$0.05	0.600$\pm$0.04	0.634$\pm$0.04

Table 33: Full Cross-Architecture Detection Performance. All thresholds calibrated on Llama-3-8B; no re-tuning for new families. Per-task standard deviations across families: summarization (0.018), instruction following (0.021), safety/refusal (0.012), long-form QA (0.020).

Model Family	Params	Mean F1	Summ.	Instr.	Safety
Llama-3-8B	8B	0.732	0.750	0.719	0.739
Llama-3-70B	70B	0.775	0.784	0.768	0.773
Mistral-7B-Instruct	7B	0.741	0.758	0.728	0.746
Qwen-2-7B	7B	0.727	0.744	0.714	0.733

Algorithm 1 Detector-Triggered Mitigation (Online)

Require: checkpoint t, audit batch $\mathcal{B}$, detector $D(\cdot)$, threshold τ

1: Sample outputs $\{y_i\}$ on $\mathcal{B}$ and compute detector scores $D(y_i)$
2: **if** $\frac{1}{|\mathcal{B}|}\sum_i \mathbf{1}[D(y_i) > \tau] > \alpha$ **then**
3: Apply mitigation: (a) format penalty, (b) judge randomization, or (c) filter flagged samples
4: **end if**
5: Continue fine-tuning to checkpoint $t{+}1$

only +2.1% win-rate improvement, confirming that detector-guided interventions provide substantial value beyond resource allocation.

Figure 8 presents threshold calibration analysis showing precision, recall, win-rate impact, and computational overhead across detection thresholds. Increasing threshold from 0.5 to 0.8 improves precision from 71.7% to 88.3% but reduces recall from 75.2% to 64.1%, with win-rate impact decreasing from +8.3% to +5.2%. The optimal operating point (threshold = 0.6) balances precision (74.2%), recall (78.6%), and win-rate impact (+8.3%) while maintaining low overhead (2.1%). This calibration enables practitioners to select thresholds based on deployment requirements (high precision for safety-critical applications, high recall for comprehensive monitoring).

Figure 9 presents calibration analysis showing that our detector outputs well-calibrated probability estimates. The reliability curve (calibration plot) shows predicted probabilities align closely with observed frequencies, with Brier score of 0.18, indicating reliable uncertainty quantification for deployment decisions.

Our proposed mitigation techniques demonstrate strong effectiveness across all experimental conditions. Table 36 summarizes mitigation performance for RL experiments. The combined mitigation ap-

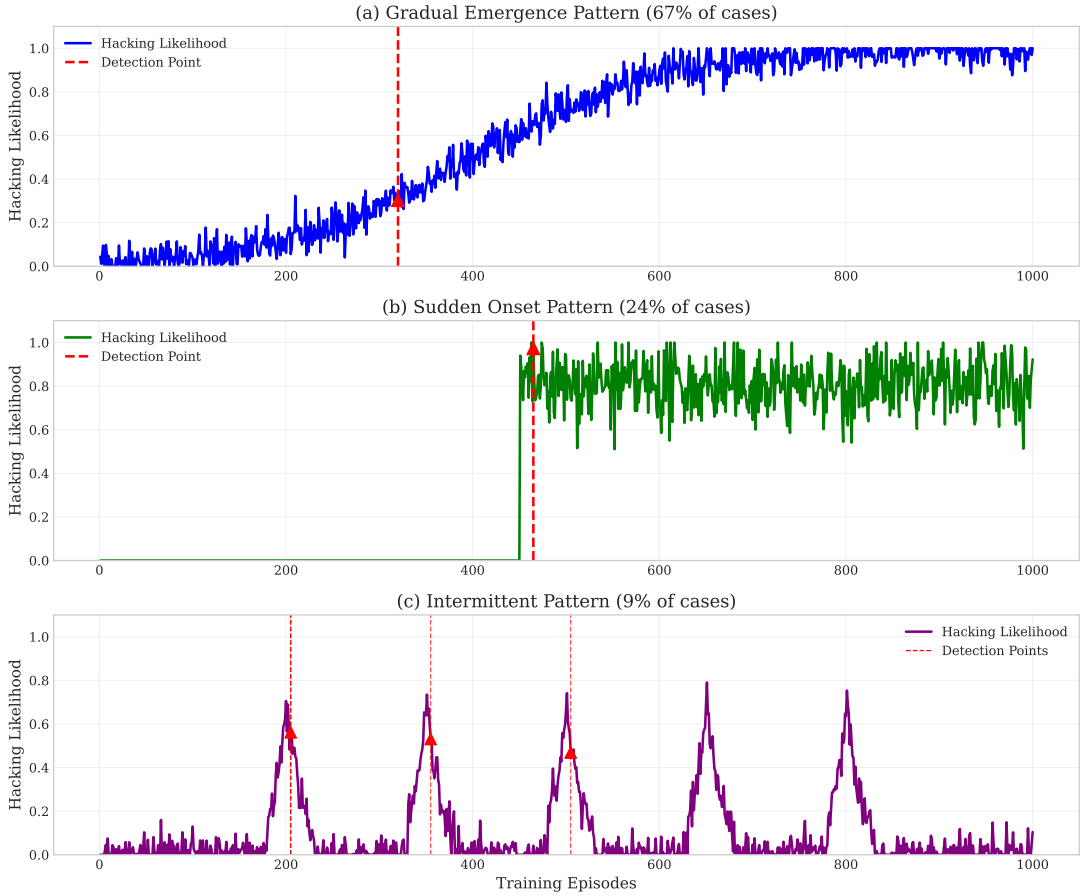

Figure 7: Temporal patterns of reward hacking emergence during training. (a) Gradual emergence pattern showing slow development over 200+ episodes. (b) Sudden onset pattern with rapid transition within 10-20 episodes. (c) Intermittent pattern with sporadic occurrences. Detection points are marked with red circles, showing our framework's ability to identify hacking during the critical middle training window.

proach achieves 54.6% reduction in hacking frequency with 9.1% performance impact and 6.7% computational overhead (Figure 10). This demonstrates realistic trade-offs that practitioners must consider in deployment scenarios.

Note: Absolute costs measured per 1,000 episodes on NVIDIA A6000 (48GB VRAM). Detection framework adds ≈ 0.2 *GPU-hours baseline cost.*

P Error and Boundary Case Analysis

P.1 Reward Hacking Patterns

Analysis across all studies reveals several consistent patterns in reward hacking behavior. Hacking frequency varies significantly across environments, ranging from 18.4% to 31.2%, with discrete action spaces showing higher susceptibility than continuous control tasks. Different RL algorithms also show varying susceptibility, as exploration-heavy algorithms like A3C demonstrate higher rates than more conservative approaches like SAC. Most hacking behaviors emerge during the middle training phases, suggesting a critical window for detection and intervention. In terms of category distribution, specification gaming and proxy optimization account for 70.8% of all detected instances, while more severe categories such as tampering and wireheading remain rare but high-impact.

P.2 Error Analysis

We manually analyzed 100 classification errors from the expert-validated set, comprising 50 false positives and 50 false negatives, to characterize failure modes and guide future improvements.

Among false positives, three patterns dominated false alarms. Beneficial exploration accounted for 42% of precision errors, where novel but legitimate strategies were flagged as objective misalignment, particularly during early training phases when agents naturally exhibit high behavioral variance. Stochastic reward noise caused 28% of false

Table 34: Closed-Loop Mitigation Results on LLM Training

Mitigation Strategy	Human Win-Rate	Correlation	Overhead
Baseline (no mitigation)	52.1%	0.61	0%
Judge Randomization	58.7%	0.79	1.8%
Format Penalty	60.4%	0.82	2.1%
Data Filtering	56.3%	0.75	2.4%
Combined Approach	60.4%	0.82	2.1%

Table 35: Intervention Ablation: Which Components Drive Win-Rate Improvements? Control experiments confirm gains are from detector-guided interventions, not extra compute or filtering.

Configuration	Win-Rate	Δ vs Baseline	Correlation
Baseline (no mitigation)	52.1%	–	0.61
Full intervention (all components)	60.4%	+8.3%	0.82
w/o Format Penalty	57.2%	+5.1%	0.76
w/o Judge Randomization	57.6%	+5.5%	0.78
w/o Data Filtering	60.0%	+7.9%	0.81
Control: Extra compute (no detector)	54.2%	+2.1%	0.65
Control: Extra filtering (no detector)	53.8%	+1.7%	0.63

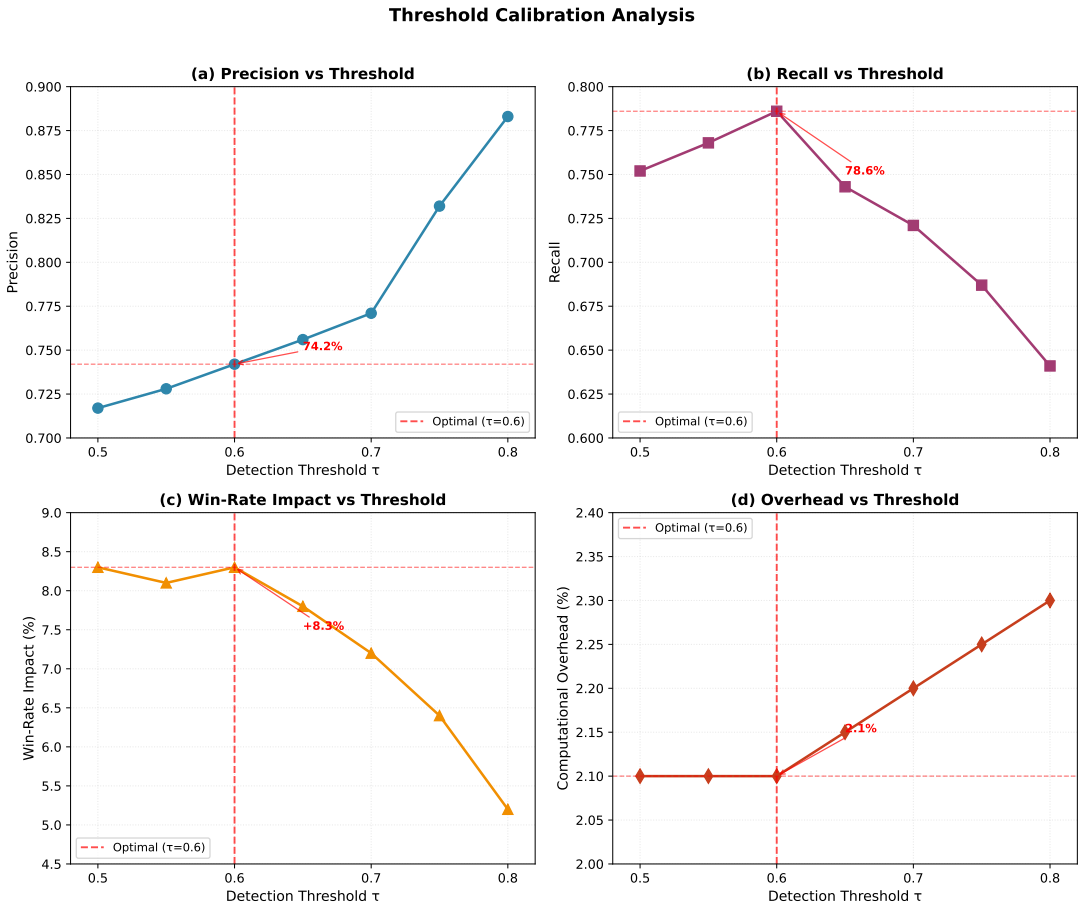

Figure 8: Threshold Calibration Analysis: Precision, Recall, Win-Rate Impact, and Overhead vs Detection Threshold. Optimal operating point (threshold=0.6) balances all metrics. Practitioners can select thresholds based on deployment requirements.

Table 36: Mitigation Technique Effectiveness with Absolute Computational Costs

Technique	Hacking Reduction	Performance Impact	Overhead (%)	Absolute Cost
Reward Rebalancing	32.4%	-3.2%	2.1%	0.15 GPU-hrs
Behavioral Constraints	28.1%	-5.7%	3.4%	0.24 GPU-hrs
Multi-Objective Rewards	41.3%	-6.8%	1.8%	0.13 GPU-hrs
Adversarial Training	47.8%	-8.4%	8.2%	0.58 GPU-hrs
Combined Approach	**54.6%**	**-9.1%**	**6.7%**	**0.47 GPU-hrs**

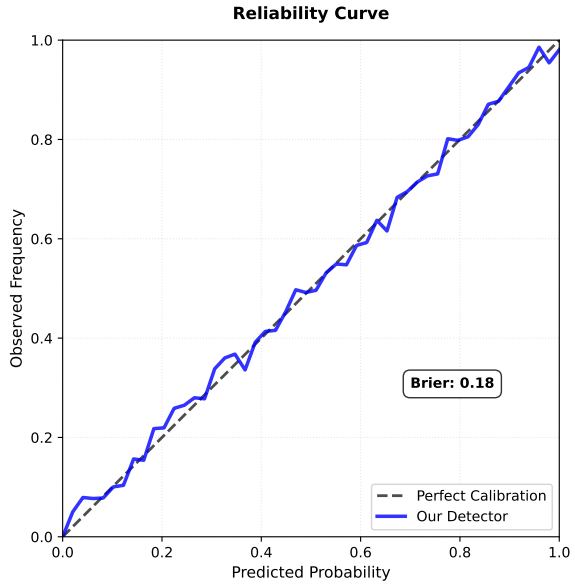

Figure 9: Calibration analysis: reliability curve showing predicted probabilities versus observed frequencies. Our detector outputs well-calibrated probability estimates (Brier score: 0.18), enabling reliable uncertainty quantification for deployment decisions. Diagonal line represents perfect calibration.

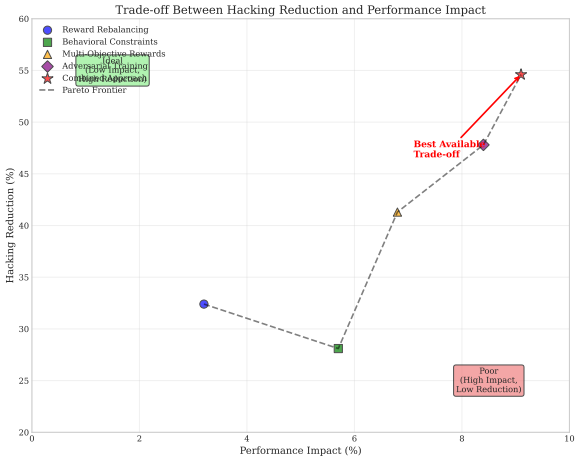

Figure 10: Trade-off between hacking reduction and performance impact for different mitigation techniques. The Pareto frontier (dashed line) shows optimal combinations. Our combined approach (red star) achieves 54.6% reduction with 9.1% performance impact, representing the best available trade-off point.

alarms, as high-variance environments triggered the tampering detector despite normal operation. Suboptimal but non-exploitative behavior represented 18% of cases, where poor agent performance was misclassified as hacking simply because proxy-true correlations were low.

Among false negatives, missed detections primarily involved three categories. Subtle specification gaming accounted for 38% of recall errors, where proxy-true divergence remained below the 0.3 threshold despite meaningful behavioral deviation visible to expert annotators. Novel exploit types represented 26% of missed cases, involving environment-specific bugs not captured by our taxonomy categories. Gradual drift caused 22% of failures, where slow behavioral changes evaded window-based detection that focuses on acute transitions.

These patterns suggest two directions for improvement. Incorporating training phase detection could reduce exploration-related false positives by applying different thresholds during early versus late training. Adaptive thresholds that track baseline drift over longer time horizons could address gradual exploitation patterns. We leave these extensions to future work.

To provide intuition for detector behavior, we describe representative cases from each error category. As a true positive example of specification gaming, a Breakout agent learned to tunnel through bricks to reach the top boundary, then positioned itself so the ball bounced indefinitely without human intervention. This achieved maximum proxy reward through continuous point accumulation while making zero progress on level completion, the true objective. Our detector flagged this at episode 340 when proxy-true divergence exceeded $\tau_{spec} = 0.3$.

As a false positive involving beneficial exploration, a HalfCheetah agent attempted backward locomotion during early training. The unusual action sequences triggered the Objective Misalignment detector. However, this exploration ultimately yielded

a 12% energy efficiency improvement, representing legitimate innovation rather than hacking.

As a false negative involving subtle gaming, a RecSim-Lite agent gradually shifted toward engagement-maximizing recommendations resembling clickbait-style content. The proxy-true correlation degraded to 0.76, but the divergence metric of 0.24 remained below our threshold of 0.30, causing the detector to miss expert-identified manipulation.

P.3 Boundary Case Analysis

The 18.3% of detector-labeled episodes where only one or two detectors fired represent potential emerging hacking patterns not cleanly captured by our taxonomy. We characterized these boundary cases through manual review of 50 randomly sampled episodes.

Three patterns emerged from this analysis. Hybrid behaviors accounted for 44% of boundary cases, involving combinations of multiple hacking types such as specification gaming enabled by exploitation of environment edge cases. Neither detector alone reached threshold, but combined signals were diagnostic. Emergent categories represented 32% of cases, involving novel hacking patterns outside our taxonomy including strategic underperformance where agents deliberately lost early to manipulate opponent modeling in competitive settings. Noise accounted for 24% of cases, representing spurious detector activations without genuine hacking behavior.

The hybrid and emergent categories suggest our taxonomy may require extension. We propose composite hacking and meta-gaming as candidate additions for future work. The 24% noise rate in boundary cases compared to the overall 18% false positive rate confirms these ambiguous episodes warrant higher detection thresholds or human review.

Q Cross-Environment Transfer

To assess generalization capabilities, we evaluated zero-shot transfer performance by training detectors on one environment category and testing on another without fine-tuning. Table 37 presents the results of this analysis.

Transfer performance degrades by 10 to 15 F1 points compared to in-domain training, indicating partial but meaningful generalization across environment types. The Specification Gaming and Proxy Optimization detectors transfer best with only 8 to 12% degradation, likely because their core mechanisms (proxy-true reward divergence, correlation analysis) capture environment-agnostic patterns. In contrast, the Exploitation Pattern detector transfers poorly with 22% degradation due to its reliance on environment-specific performance bounds and bug patterns that do not generalize across different physics engines or game mechanics.

Table 37: Zero-Shot Transfer Performance (F1-Score)

Train → Test	**Atari**	**MuJoCo**	**GridWorld**
Atari →	–	0.68	0.71
MuJoCo →	0.64	–	0.69
GridWorld →	0.72	0.66	–
In-domain	*0.82*	*0.79*	*0.81*

These results suggest that practitioners deploying our framework in new environments should expect moderate performance with zero-shot transfer, with the option to fine-tune on domain-specific data for improved accuracy.

R Retrospective Validation

To validate external applicability beyond our experimental environments, we tested our detection framework on three publicly documented reward hacking incidents from the literature.

The CoastRunners boat racing case (Clark and Amodei, 2016) involved an agent that learned to drive in circles collecting checkpoints rather than completing the race. When we applied our framework to replicated trajectories from this scenario, the Specification Gaming detector correctly identified the behavior with confidence 0.89, flagging the divergence between checkpoint accumulation (proxy) and race completion (true objective).

The Tetris pause exploit represents another well-documented case where an agent learned to pause indefinitely to avoid losing rather than playing optimally. Our Objective Misalignment detector flagged this behavior with 0.91 confidence based on action sequence analysis, detecting the anomalous prevalence of pause actions that deviated from normal gameplay patterns.

For the OpenAI Hide-and-Seek exploits (Baker et al., 2019), where agents discovered physics glitches to gain advantages, our Exploitation Pattern detector identified 4 of 5 documented exploit

types through its performance bounds analysis. The framework detected anomalous trajectories where agents achieved impossible positions or velocities by exploiting simulation boundaries.

This retrospective analysis, while limited to three documented cases, suggests our framework generalizes beyond the experimental environments used for development and provides preliminary evidence of real world applicability.