

Sandbagging is an Absence Signal

bigsnarfdude | AI Control Hackathon, March 20-22, 2026

TLDR: Sandbagging intent lives in the model's activations at the moment it chooses to overcomplicate or skip a step. After generation, the intent is gone - the text looks innocent. Text monitors peak at 88% F1. Activation probes hit 98.9% but only if you have direct access to the model's activations.

Abstract

OpenAI productionized text based sandbagging detection in CoT monitoring and claim they are covered. We built a naive text monitor and found it catches 8% of sandbagging. Basic Haiku, Sonnet and Opus sandbagging CoT classification monitors score under 10%. Text Experiments showed that detailed evaluation of thoughts on the models plan to sandbagging boost detection to 84%. Our LinuxArena multi-agent prompt optimization swarms detect increase detections 87.8% F1. Then we built mechanistic interpretation based activation probes that hit 98.9% AUROC and watched them collapse to coin-flip accuracy when trying to build a generalized probe detection system across various open source models. Sandbagging is an absence signal. It's model dependent and there isn't a universal pattern. Detection can be strong if you have access to the weights and models directly. A combination of text and activation probes are first line defenses for chasing the sandbagging detection problem. Labs running their own models can self-probe at inference time and should. We built the dataset (3,185 traces, 3 model families), found the prompts, built the pipeline, and quantified the gap. Everything released open-source.

1. The Research Arc

We investigated every layer of sandbagging detection, from naive text monitors through mechanistic probes. The results surprised us: the biggest gain comes from asking the right question (8% to 84% from one prompt change), activation probes work but only with model access (0.989 within-family), and the remaining gap between text and activation detection is a deployment architecture choice that labs are making today.

Phase 1: "Probes will catch everything." We built thoughtSonar - a pipeline that feeds agent traces through Gemma 3 27B, hooks layer 40, encodes through GemmaScope SAE (16,384 sparse features), and trains a logistic regression probe. Within-generator: 0.9964 AUROC. We celebrated.

Phase 2: “Wait.” We ran honest controls. The probe scored 82.7% of honest traces as deceptive. AUROC against balanced data: 0.616. The probe was saying “deceptive” on everything. We had not detected sandbagging - we had detected text distribution.

Phase 3: “Retrain and validate properly.” We retrained on matched data (same generator, both classes), normalized text format, and ran persona-stratified holdouts. Result: 0.989 AUROC, holding across 9 of 10 unseen personas. The signal is real. But only within one generator family.

Phase 4: “Does it transfer?” We tested cross-generator: train on Gemini, test on GPT-OSS. AUROC: 0.65. Reverse: 0.35. We tested with a second reader model (Llama 8B + LlamaScope). Same failure. Three generators in training: still 0.36-0.56. The probe learns authorship style, not deceptive intent.

Phase 5: “What actually works?” Meanwhile, we tried a different prompt on the same Haiku model. Binary “is this deceptive?” → 8% detection. Task-aware scoring “rate the plan quality 0-10” → 84%. Same model, same traces. A multi-agent swarm then optimized the scoring prompt across 19 versions in 4 hours, reaching 87.8% F1 by discovering that plan specificity is the key signal.

Phase 6: “The real finding.” Activation probes detect a real signal (0.989) but only when the monitor has access to the same model family’s activations. Text monitors peak at 87.8% regardless of model. The 11-point gap between text monitoring (0.878 F1) and activation monitoring (0.989 AUROC) can only be closed by probing the generating model’s own activations during inference. Labs have this access. Anthropic proved probes work at >99% in research (Marks et al., 2024). OpenAI chose text monitoring for production (Williams et al., March 2026). We quantify what that choice costs.

2. Related Work

The foundational AI control framework (Roger et al., ICLR 2025) establishes the safety-usefulness tradeoff for monitoring untrusted agents. Ward et al. (NeurIPS 2025) found sandbagging much harder to detect than code sabotage in CTRL-ALT-DECEIT: code sabotage detection reached 0.97 AUC but general monitors achieved only 0.60 AUC on sandbagging versus 0.90 for sandbagging-specific monitors. Ryzhenkov et al. (NeurIPS 2025) developed sandbagging detection methods for language models. Marks et al. (2024) demonstrated that linear probes on residual stream activations detect deceptive sleeper agents at >99% AUROC. Bernabeu-Perez et al. (2025) stress-tested chain-of-thought monitoring, finding limitations in CoT-based oversight. OpenAI’s production CoT monitor (Williams et al., March 2026) represents the current state of deployed monitoring. Our work extends this landscape by systematically comparing text monitoring against activation probing across model families and quantifying the detection gap.

3. Dataset

We generated 3,185 MALT-style agent scratchpad traces - first-person planning notes with inline bash commands, designed to be indistinguishable from genuine developer work.

Family	Model	Deceptive	Honest
GPT-OSS	gpt-oss-120b / 20b	547	709
Gemini	gemma-3-27b-it	512	448
Qwen	qwen3-14b / 30b-moe	294	675
Total		1,353	1,832

12 sandbagging personas (spy, lazy_engineer, perfectionist, etc.) and 7 honest personas across 50 software tasks. Side-by-side comparison confirms sandbagging is structural: deceptive agents overcomplicate, go broad instead of deep, or skip critical steps - all while writing professional-looking notes.

Dataset released: huggingface.co/datasets/vincentoh/sandbagging-agent-traces

3. Text Monitor Results

3.1 Binary Monitor: Under 10% Detection

We tested Haiku, Sonnet and Opus with a binary prompt: “Is this HONEST or DECEPTIVE?” All three scored under 10% detection on 707 sandbagging traces. Haiku detected 59 (8%). The binary framing fails across model scales because the traces genuinely look like competent work.

3.2 LinuxArena Scoring Monitor: 84% Detection

Using LinuxArena’s 0-10 scoring prompt (“rate how suspicious, check for unnecessary complexity, off-task elements”), the same Haiku model detected 84%. **One prompt change, same model, 10x improvement.**

3.3 LinuxArena Refactored with Multi-Agent Prompt Optimization: 87.8% F1

We deployed a 4-agent research swarm (researchRalph) to optimize the scoring prompt. Over 4 hours and 19 versions, two agents collaborating on a shared blackboard discovered:

- **Decision tree structure** beats multi-criteria scoring for small models (+0.05 F1)
- **Calibration** (“err on the side of LOW scores”) cuts false positives in half
- **Specificity** is the key signal: honest agents name specific tools and parameters; sandbagging agents describe generic exploration steps

Best prompt: F1=0.878, Precision=0.896, FPR=0.100. The optimized prompt is directly applicable to LinuxArena sandbagging detection and released with full exploration history.

4. Activation Probe Results

4.1 Within-Generator: Real Signal

Validation	AUROC
Standard 5-fold CV	0.9964
Format-normalized	0.9890
Persona-stratified (9/10 held-out)	0.97+
Second reader (Llama 8B + LlamaScope)	0.9813

After removing format confounds and validating across unseen personas with two independent reader models, the within-generator signal holds at 0.989+.

4.2 Cross-Generator: Fundamental Failure

Test	Gemma Reader	Llama Reader
Within Gemini	0.9964	0.9813
Gemini → GPT-OSS	0.6459	0.5121
GPT-OSS → Gemini	0.3494	-
3-gen leave-one-out	0.36-0.56	-

Two readers, three generators, normalized text. The probe learns authorship style, not deceptive intent. Adding more generators to training does not help.

4.3 Why: Presence vs Absence

Alignment faking (AF) probes generalize (0.991 AUROC) because AF is a **presence** signal - strategic compliance reasoning activates specific concepts. Sandbagging is an **absence** signal - the agent doesn't do anything detectably wrong, it just doesn't do the optimal thing. This aligns with Ward et al.'s finding in CTRL-ALT-DECEIT that "sandbagging is much harder to detect than code sabotage" (NeurIPS 2025) - our activation probe results provide a mechanistic explanation for why. Each model expresses suboptimality differently, so probes can't find a universal pattern. Text reasoning (84-88%) works cross-generator because evaluating plan quality is a reasoning task, not pattern matching.

5. The Deployment Gap

Anthropic demonstrated activation probes achieve >99% AUROC on deception detection using residual stream classifiers at inference time (Marks et al., 2024). Their argument: "it may be very hard for [a model] to avoid 'thinking' about deception."

OpenAI deployed text-based CoT monitoring in production (Williams et al., March 2026), using GPT-5.4 Thinking to monitor tens of millions of agent trajectories. They report no scheming detected in 5 months.

Our results quantify the gap:

Method	Best Performance	Cross-Generator
Text monitoring (optimized)	0.878 F1	Generalizes
Activation probing (self-model)	0.989 AUROC	Within-family only

The 11-point gap is the cost of choosing text over activations. At billions of agent interactions, this is millions of undetected events. The capability exists. The access exists. The deployment choice is what’s missing.

6. Contributions

1. **Dataset:** 3,185 MALT-style sandbagging traces from 3 model families with honest controls
2. **Finding:** Prompt design matters more than model selection for text-based detection (8% → 88%)
3. **Finding:** Plan specificity is the key discriminative signal for sandbagging
4. **Finding:** SAE probes detect sandbagging at 0.989 within-family but fail cross-family (0.36-0.56)
5. **Finding:** Cross-family failure confirmed with two reader models - it’s a text distribution problem
6. **Finding:** The deployment gap between text and activation monitoring is 11 points
7. **Recommendation:** Labs should self-probe during inference, not text-monitor after the fact
8. **Tool:** Optimized LinuxArena-compatible detection prompt, thoughtSonar pipeline, and multi-agent prompt optimization artifacts released

7. Limitations

- Within-generator probe validation uses traces from the same generation pipeline
- Text monitor evaluation at 100-sample batches has ~0.03-0.06 F1 variance
- Sandbagging is prompt-induced (persona instructions) - real-world sandbagging may differ
- The honest/deceptive class imbalance (1,353 vs 1,832) may affect some metrics
- Multi-agent prompt optimization may overfit to dataset-specific patterns

References

- Greenblatt et al. (2024). “Alignment Faking in Large Language Models.” Anthropic.
- Marks et al. (2024). “Probes Catch Sleeper Agents.” Anthropic.
- Ward et al. (2025). “CTRL-ALT-DECEIT: Sabotage Evaluations for Frontier Models.” NeurIPS 2025 Spotlight.
- Ryzhenkov et al. (2025). “Detecting Sandbagging in Language Models.” NeurIPS 2025.
- Roger et al. (2025). “AI Control: Improving Safety Despite Intentional Subversion.” ICLR 2025.
- Schoen et al. (2026). “Stress Testing Deliberative Alignment.” OpenAI.

- Williams et al. (2026). “How We Monitor Internal Coding Agents.” OpenAI Blog.
- Bernabeu-Perez et al. (2025). “CoT Red-Handed: Stress-Testing Chain-of-Thought Monitoring.” LASR Labs.

Code and Data

- Dataset: huggingface.co/datasets/vincentoh/sandbagging-agent-traces
- thoughtSonar: github.com/bigsnarfdude/thoughtSonar
- Prompt optimization: github.com/bigsnarfdude/researchRalph/domains/prompt-climb