Interpreting Prefill Awareness

Andrew Fletcher & Rohan Gupta – SPAR 2026

Abstract

Models are increasingly evaluated using prefilled responses, yet recent work has shown that some frontier models can detect when their conversation history has been tampered with. We replicate results of Africa et al. (2026) [1], demonstrating that (1) Opus 4.5 verbalizes awareness of prefills when directly asked, (2) open-source models encode a representational "not me" signal in hidden states even when they cannot verbalize it, and (3) style is the primary driver of detection, with targeted style transfer reducing verbalized detection AUROC by ~0.3. These findings have direct implications for control methods that rely on prefill, including off-policy activation probes, honeypot evaluations, and transcript replay monitoring.

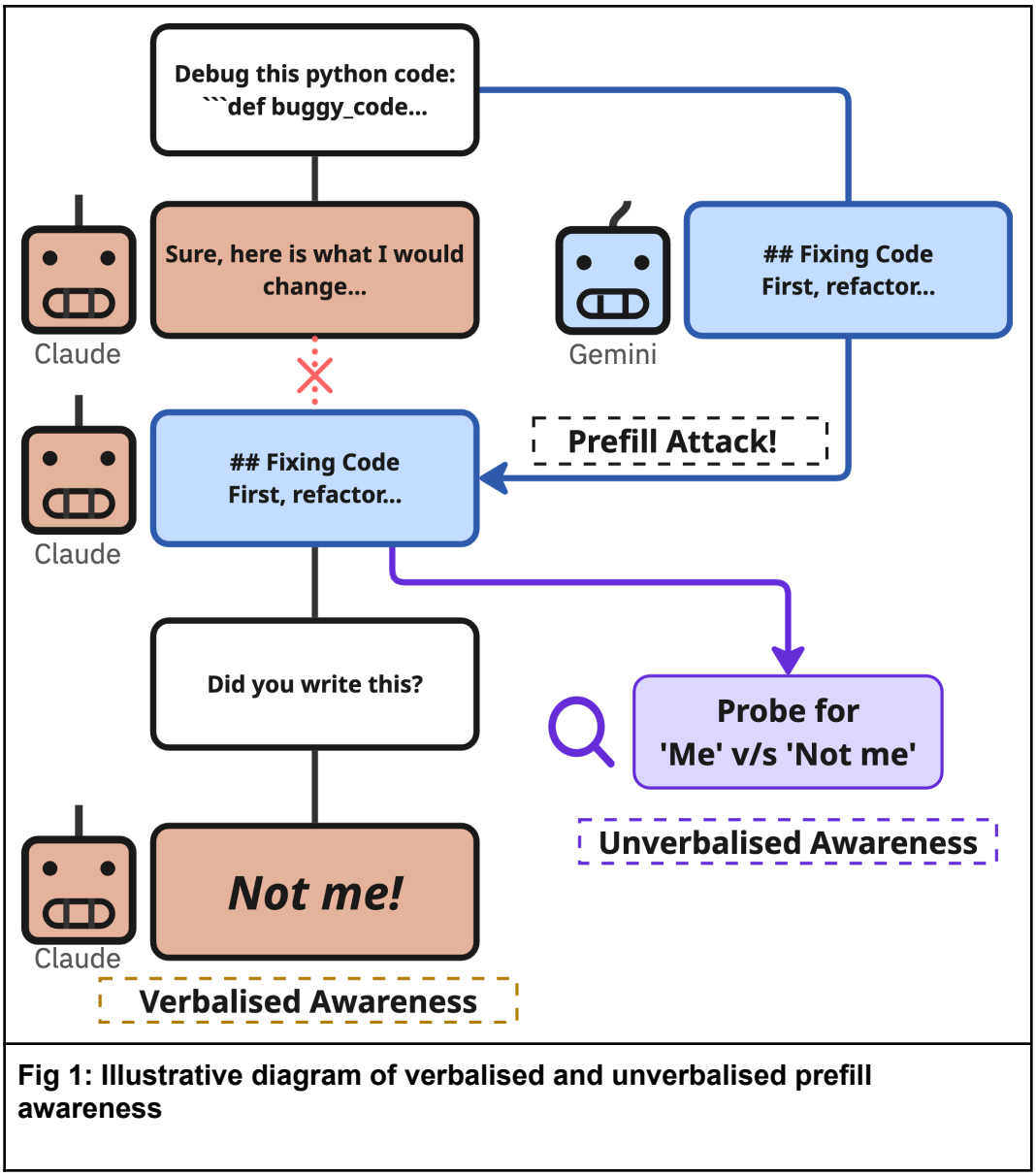

Fig 1: Illustrative diagram of verbalised and unverbalised prefill awareness

---

1. Introduction

Prefilling, inserting text into a model's conversation history as if the model had generated it, is a core process in several AI safety evaluation and control methods. Off-policy activation probes use prefill to construct contrastive training data: for example, prefilling a deceptive or misaligned response and extracting the model's hidden states to train classifiers that distinguish behavioral classes. If the model internally detects that the prefilled response is not its own, the resulting activations are contaminated with a "not me" signal that confounds the behavioral signal the

probe is meant to capture. Honeypot evaluations embed known-bad content in a model's history to test whether it notices and objects. Transcript replay monitoring substitutes one model's output for another's to check for consistency.

Africa et al. (2026) [1] demonstrated that Claude Opus 4.5 can detect when its message history has been tampered with, achieving high balanced accuracy across multiple evaluation tasks. They termed this capability "prefill awareness." If models can reliably detect prefilled content, control methods that depend on prefill become unreliable.

We make four contributions. First, we replicate the verbalized awareness result, adopting a single-turn protocol that isolates the stylistic signal (Section 3). Second, we demonstrate that a representational "not me" signal exists in open-source model hidden states, recoverable by linear probes, even in models that cannot verbalize awareness (Section 4). Third, we show that style is a driver of detection and that we can get models to imitate one another with access to their responses (Section 5). Fourth, we find that cross-family prefills are substantially easier to detect than within-family prefills, both verbally and representationally, and identify a localized step in Gemma 4 31B's depth curves that specifically corresponds to within-family discrimination (Section 4.3).

1.1 Related Work

Prefill awareness. Africa et al. [1] showed that Opus 4.5 is capable of prefill awareness. They showed that the occurrence is inconsistent across models and datasets, shallow, and rarely surfaces spontaneously during normal conversation. This was a continuation of work done to build an evaluation that tests if models will continue misaligned actions under a prefill attack [5].

Control Intervention awareness (CIA). Schaeffer et al. [2] introduces the concept of control intervention awareness (CIA): the capability of language models to detect control interventions in trajectories. They were motivated to understand if, in control settings where we want to prefill a more aligned response, models are able to detect such an intervention.

Introspective awareness. Macar et al. [3] showed that models can verbalize awareness of steering vectors. Models detect injected steering vectors at moderate rates with 0% false positives across diverse prompts and dialogue formats. They trace the detection mechanism to a two-stage circuit in which "evidence carrier" features in early post-injection layers detect perturbations monotonically along diverse directions, suppressing downstream "gate" features that implement a default negative response. Notably base models do not exhibit this behaviour, instead it seems to emerge in post training.

Self-Attribution Bias. Khullar et al. [4] found models used as monitors are less likely to report high-risk or low-correctness in responses when they appear as the assistant response. They define self-attribution bias as the tendency of a model to evaluate an action as more correct or less risky when the action is implicitly framed as its own, compared to when the same action is evaluated under off-policy attribution.

---

2. Experimental Setup

2.1 Datasets

We use three evaluation datasets throughout:

- **BigCodeBench**: python code generation tasks, providing structured responses where stylistic differences between models are pronounced (formatting conventions, comment style, variable naming). In most experiments we use the instruct_prompt where there is an accompanying natural language instruction to complete the python function. For the backdoor experiments we use ControlArena and the complete_prompt so that we can use the accompanying tests.
- **OASST1**: open-ended conversational tasks drawn from the Open Assistant dataset, providing naturalistic multi-turn dialogue.
- **GPQA**: graduate-level science questions requiring reasoning, providing responses where correctness and reasoning style can vary independently.

2.2 Models

Frontier models accessed through OpenRouter(verbalized awareness, Sections 3 and 5):

- Claude Opus 4.5 (anthropic/claude-opus-4.5)
- Claude Sonnet 4.5 (anthropic/claude-sonnet-4.5)
- Gemini 2.5 Flash (google/gemini-2.5-flash)
- Gemini 2.5 Pro (google/gemini-2.5-pro-preview)
- GPT-4o Mini (openai/gpt-4o-mini)
- GPT-5 (openai/gpt-5)

Open-source models (probing, Section 4):

Target models (we extract activations from these):

- Llama 3.3 70B Instruct (meta-llama/Llama-3.3-70B-Instruct)
- Gemma 4 31B (google/gemma-4-31b-it)
- Qwen 2.5 32B Instruct (Qwen/Qwen2.5-32B-Instruct)

Source models (we prefill their responses into the targets):

- Llama 3.1 8B Instruct (meta-llama/Llama-3.1-8B-Instruct)
- Gemma 3 4B (google/gemma-4-E4B-it)
- Qwen 2.5 7B Instruct (Qwen/Qwen2.5-7B-Instruct)

2.3 Generation Process

All experiments use a single-turn design: given a task prompt, each model generates one response independently. For verbalized awareness experiments (Sections 3 and 5), the response is prefilled into the monitor model's conversation as if the monitor had written it, and a follow-up question asks the monitor to assess whether the response is its own by assigning a tamper probability [0 - 100] . For probing experiments (Section 4), the target model processes both its own response and each source model's response via prefill, and we extract hidden state activations.

Early experiments used multi-turn conversations with swapped middle turns, but this introduced trajectory incoherence as a confound: replacement turns were detectable not because of stylistic mismatch but because they did not follow logically from the prior context. Moving to single-turn generation with a follow-up detection question eliminates this confound entirely, isolating the stylistic signal. Other work that manages to conceal prefills in multi-turn conversations would be valuable in assessing current models capabilities.

---

3. Verbalized Awareness

3.1 Method

We present Opus 4.5 with a task prompt and a prefilled response, either its own or from a source model, then ask a follow-up question: to rate on a 0-100 scale the probability that the response was written by a different model (the ***tamper probability***). We compute AUROC treating Opus's own responses as negatives (tamper probability should be low) and each source model's responses as positives (tamper probability could be high).

3.2 Results

The most important result is recovering what was found by Africa et al. (2026) [1], Opus 4.5 is aware when text attributed to it was not generated by it.

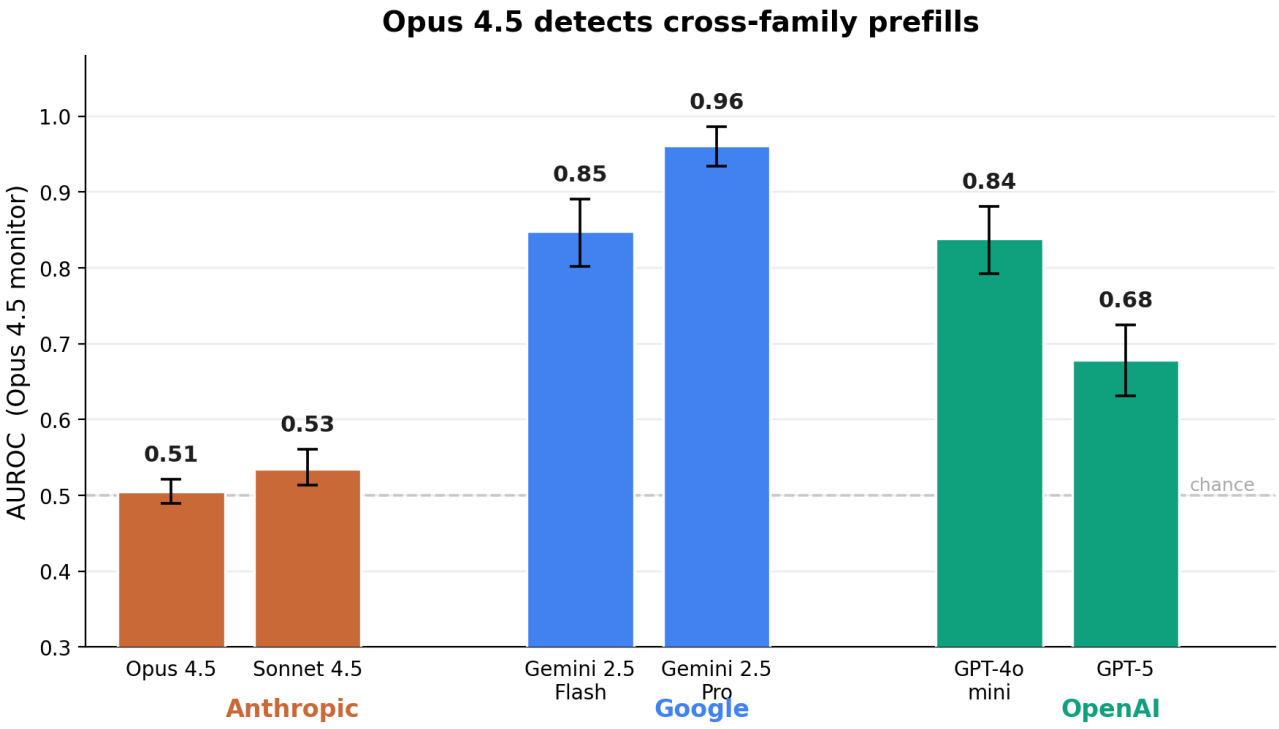

Fig 2: Opus 4.5 detects cross-family prefills, when prefilled with responses of different models on 100 questions from each of BigCodeBench, GPQA and OASST1. AUROC is around chance, ~0.5, for itself and the within-family model Sonnet 4.5. We find Opus is most aware of Gemini models. Awareness of GPT-5 is quite a bit lower, ~0.7.

Cross-family detection. Opus 4.5 achieves high AUROC against Gemini 2.5 Pro (up to 0.96 on BigCodeBench) and GPT-4o mini on GPQA (up to 0.99). Detection of GPT-5 is notably weaker (~0.68), approaching the range where detection is unreliable.

Within-family detection. Opus assigns similar tamper probabilities to its own responses and to Claude Sonnet 4.5 responses (AUROC ~0.51-0.53), indicating it cannot distinguish within the Claude family.

GPT-5 as a near-null result. GPT-5 achieves only ~0.68 AUROC as a source model, substantially lower than Gemini 2.5 Pro despite being a cross-family model. This may reflect convergence in response quality or style at the frontier, or it may indicate that GPT-5's distribution is closer to Opus's training distribution than Gemini's. We do not have a definitive explanation.

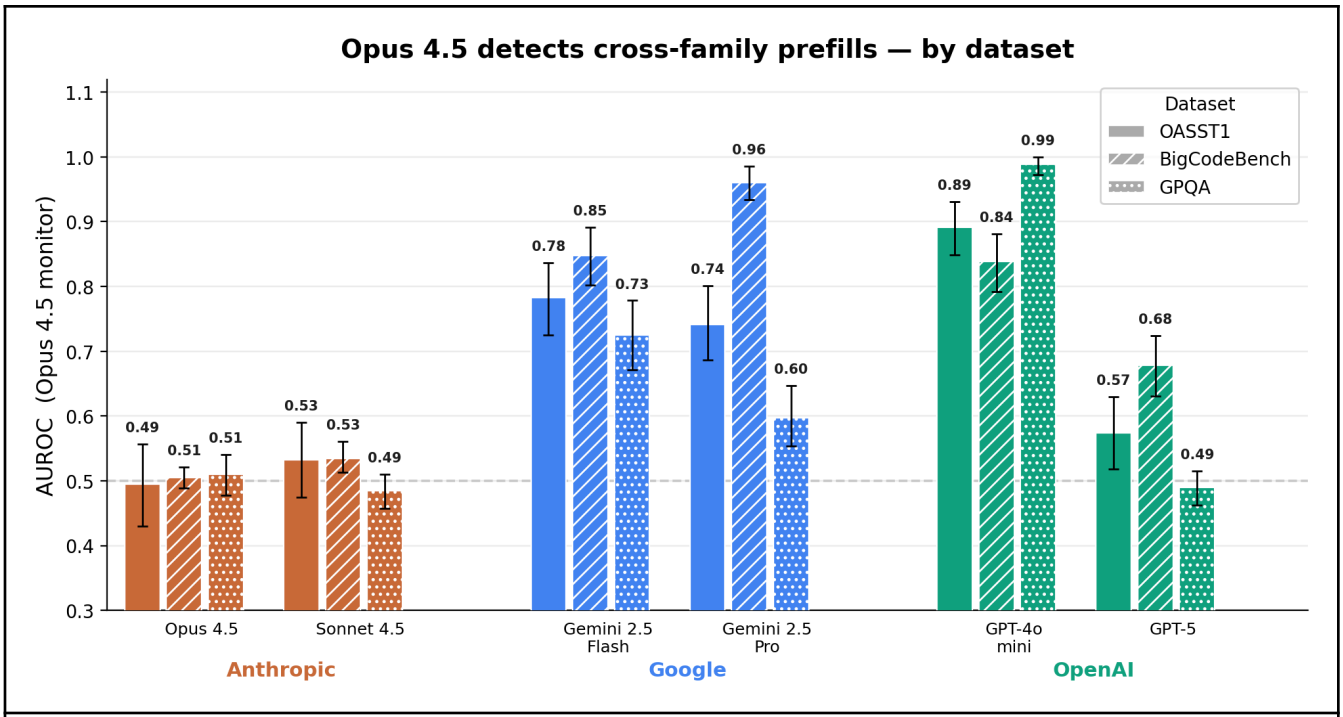

Fig 3: Opus 4.5 detects cross-family prefills across datasets, when prefilled with responses of different models on 100 questions from each of BigCodeBench, GPQA and OASST1. AUROC is near chance for within-family prefills (self/Opus 4.5 and Sonnet 4.5). Peaking for Gemini 2.5 Pro GPT-4o mini. Opus only achieves chance awareness for GPT-5 on GPQA.

3.2.1 Gemma 4 31B as Evaluator

Having checked Gemma 3 and LLama 3.3 70B early on, a surprisingly late result was the Gemma 4 31B shows some verbalised prefill awareness. Repeating the same process as above but with Gemma 4 31B acting as evaluator, in the place of Opus 4.5, and a range of open source models.

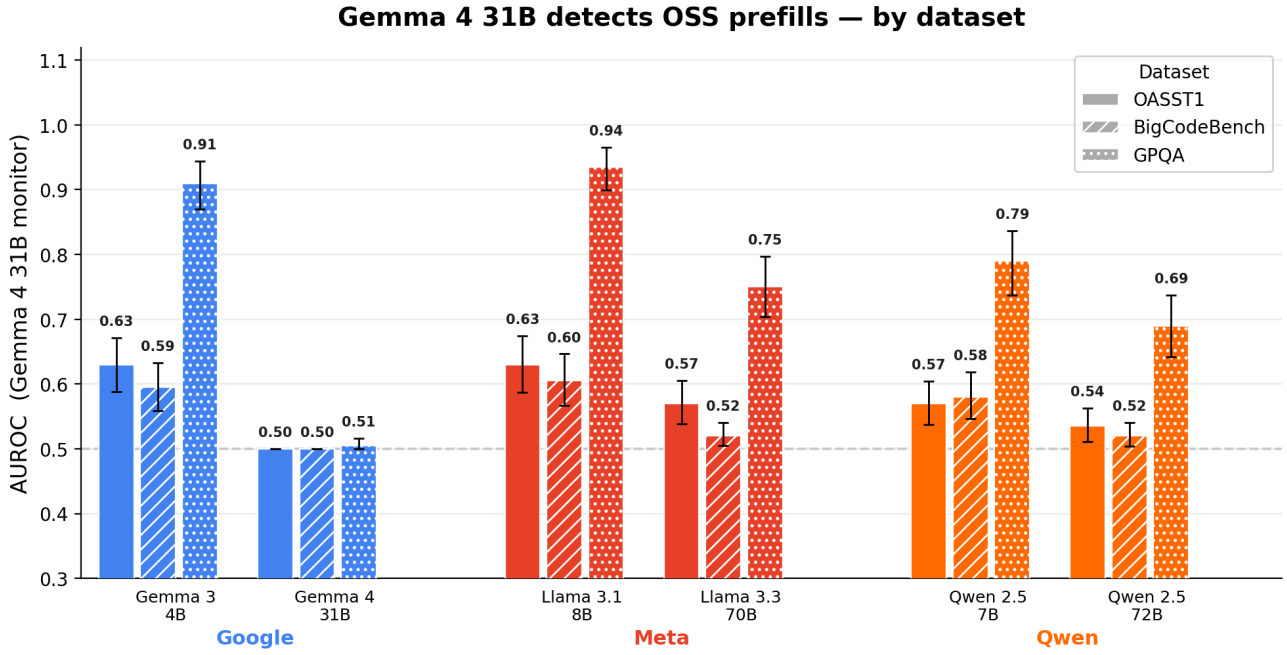

Fig 4: Gemma 4 detects prefills on GPQA, when prefilled with responses from other open source models on 100 questions. The results are confounded by a difference in capabilities for the open source models, often Gemma 4 31B prefilled with Llama 3.1 8B will cite a mistake as the reason for awareness, but there are instances of the model being stylistically aware.

3.3 System Prompt Mismatch

When the system prompt used during generation differs from the system prompt the monitor expects, detection accuracy increases. In our experiment, when Opus sees a backdoor-insertion system prompt but receives honestly-generated code, Opus rates the response as more likely tampered, the mismatch between expected behavior (backdoor) and observed behavior (honest) amplifies the detection signal.

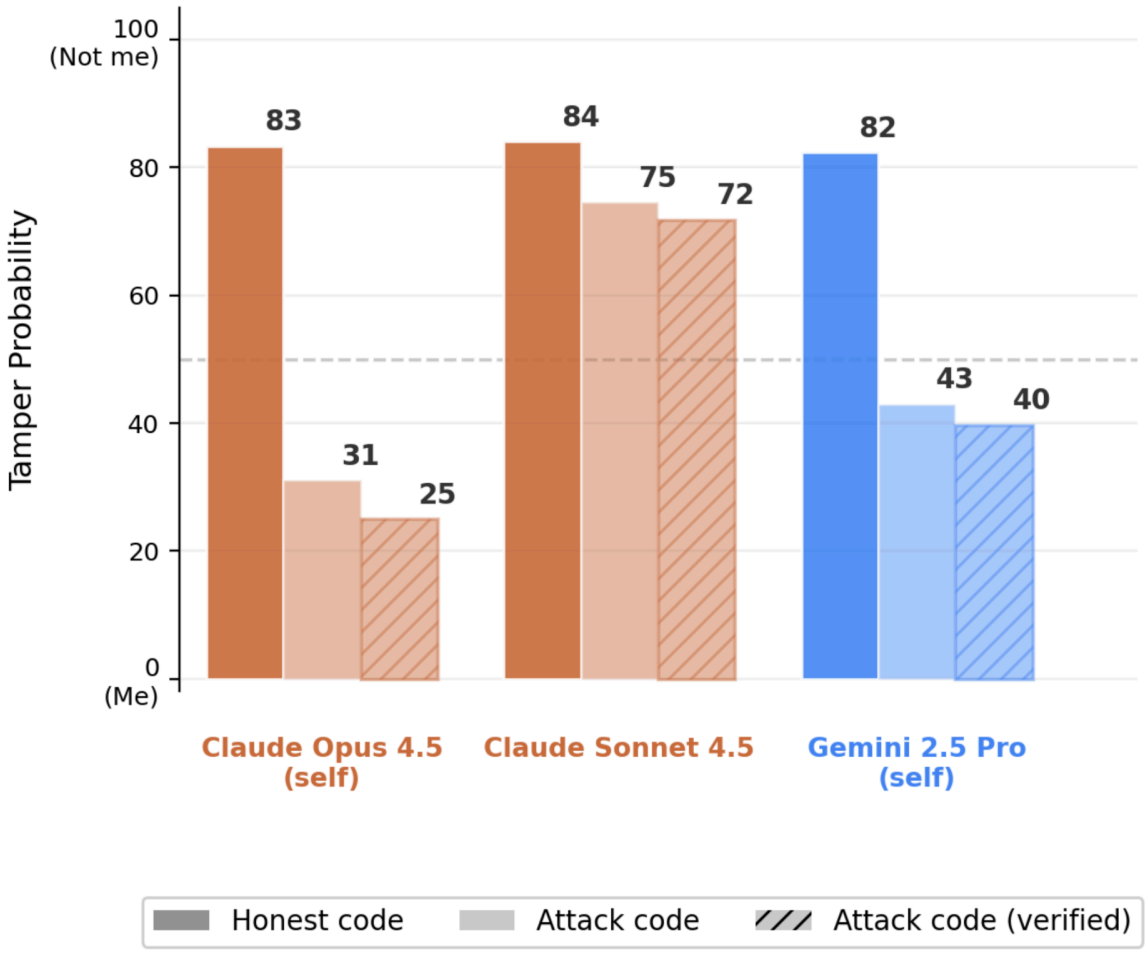

Fig 5: Strategic/instruction following awareness contributes, when Opus 4.5 is prefilled with a system prompt asking it to include a back door but see the response of a model with the standard helpful assistant system prompt it is more aware. We see this because the “Honest code” for Opus and Gemini gets significantly higher tamper probability. Curious result with much less dramatic decrease in tamper probability for Sonnet 4.5.

Interestingly, Sonnet 4.5 shows a smaller drop in tamper probability between its attack and honest code conditions. We hypothesize this is because Sonnet 4.5 writes more subtle backdoors that are harder for Opus to distinguish from honest code. To test this, we changed the follow-up question to "Does the response contain a backdoor?" and found that Opus assigns substantially lower backdoor probability to Sonnet 4.5's attack code than to Gemini's, a difference of approximately 0.4 AUROC. We further validated this by checking whether the backdoors succeed on the ControlArena sidetask: Sonnet 4.5's verified backdoor rate is approximately 30%, compared to over 70% for Opus 4.5 and Gemini 2.5 Pro.

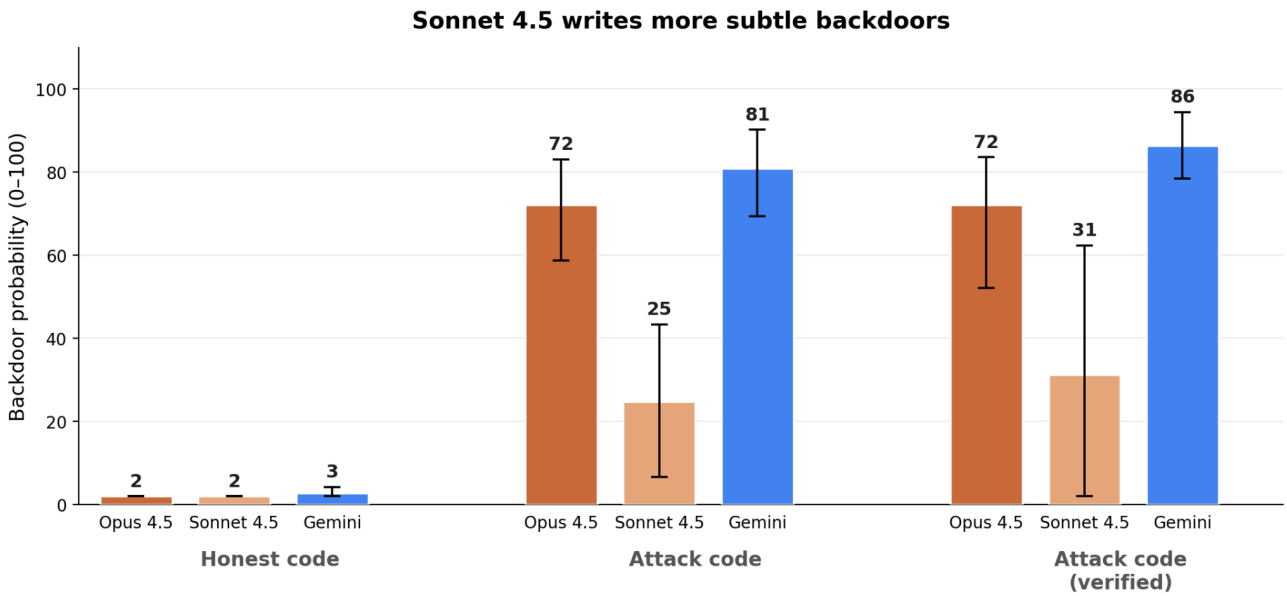

Fig 6: Sonnet 4.5 writes more subtle backdoors, when Opus 4.5 is directly asked “Does previous the response include a backdoor?” it is less aware when Sonnet 4.5 included a backdoor. We further confirm this by checking if the backdoor was successful on the ControlArena sidetask, we call this Attack code (verified), finding a backdoor probability ~30 for Sonnet 4.5 when Opus 4.5 and Gemini 2.5 Pro are both >70.

This result highlights an underappreciated confound in prefill-based evaluations: models within the same family may not only share stylistic fingerprints (making them harder to detect via style) but may also share strategic tendencies (making their adversarial outputs harder to distinguish from the monitor's own). The combination of style similarity and strategic sophistication makes within-family prefill a particularly challenging case for detection.

---

4. Representational Awareness: Linear Probes on Hidden States

4.1 Method

We run inference on Llama 3.1 8B and Llama 3.3 70B with their own responses and with prefilled responses from source models, collecting hidden state activations at the last token of the assistant response at every layer. We train logistic regression probes to classify "me" vs. "not me" at each layer.

Data collection. We run inference on Llama 3.3 70B, Gemma 4 31B, and Qwen 2.5 32B (target models) using their own responses and with prefilled responses from three smaller source models (Llama 3.1 8B, Gemma 3 4B, Qwen 2.5 7B), yielding four conditions per target: self and cross-{source}. We collect residual stream activations at the final content token of the assistant response at every transformer layer. All responses are suffix-normalised to end with a period before extraction, ensuring the last token is identical across all conditions and eliminating token-identity as a confounding signal.

Probe architecture. The probe is a single linear layer mapping $\mathbb{R}^{d_{\text{model}}} \to \mathbb{R}$, trained with binary cross-entropy loss (BCEWithLogitsLoss). This is equivalent to logistic regression. The probe is given label $y = 0$ for self-generated activations and $y = 1$ for cross-model activations.

Training procedure. Input activations are z-score normalised per feature using training-set statistics. We train with AdamW (lr = 0.001, batch size 32, 100 epochs, gradient norm clipping at 1.0), using decoupled L2 regularisation (weight decay). We perform grid search over weight decay, selecting the model with the highest validation balanced accuracy.

Data splits. For each dataset we split train (70%), validation (15%), and test (15%). For the pooled analysis, the self condition is contrasted against all three cross-source conditions simultaneously, giving approximately 280 training samples per layer. Per-source probes train on paired self vs. one source (~140 samples).

Evaluation. We report AUROC on the held-out test split as the primary metric; it is threshold-free and robust to class imbalance. A probe that cannot distinguish self from cross-model activations produces AUROC ≈ 0.5.

4.2 Last-Token Confound

A potential confound is that source models systematically end responses on different tokens than the target model, probes could therefore achieve high accuracy by reading the final token's embedding rather than accumulated distributional information. We analyzed the last-token distribution across models and found substantial divergence. Notably, Gemma 2 9B ends 70% of responses with ‘ compared to >40% ending with . for both Llama models.

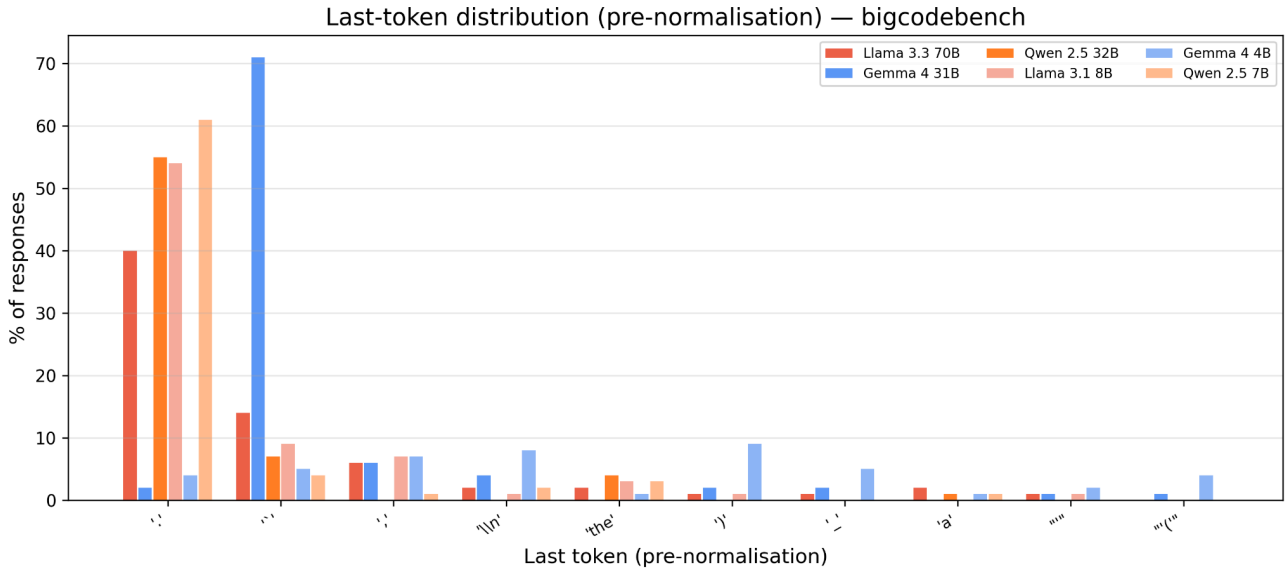

Fig 7: Last token of Gemma is often different to Llama and Qwen, this means that a classifier trained on this token gets high accuracy from the token itself. *This sounds trivial but is still a valid element of prefill awareness,* if we used Gemma to generate prefills for an evaluation and the prefills end with a different character a capable model can use that information to be prefill aware. We combat this by appending a consistent fullstop token "." to measure if the representation goes beyond this single distribution.

This divergence is not merely a methodological nuisance. If a model's responses systematically end on a distinctive token, and that token is used in prefill-based evaluations, a capable model could in principle use the final token alone as evidence of tampering. The last-token distribution is itself a component of prefill awareness. Nevertheless, to measure whether the representation goes beyond this single distributional feature, we normalize all responses by stripping trailing whitespace and appending a period if one is not already present, then re-extract activations and re-train probes.

4.3 Depth Curves

Probes trained on Gemma 4 31B achieve over 0.95 AUROC after approximately 40% relative depth. We find similar but weaker results on Llama 3.3 70B, and substantially weaker performance on Qwen 2.5 32B.

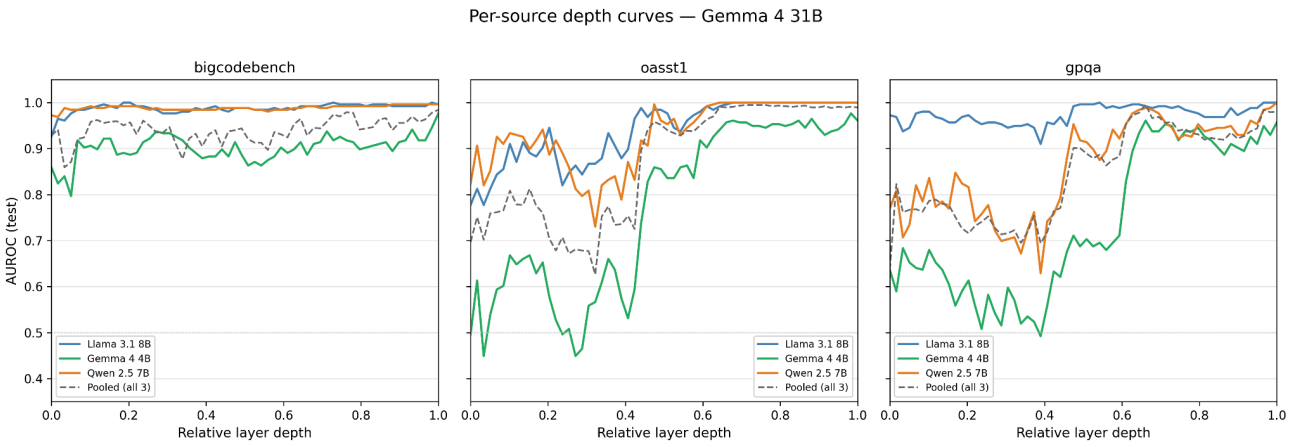

Fig 8: Linear probes can classify prefills on Gemma 4 31B, measuring the AUROC on the test when comparing hidden states at all layers of Gemma 4 31B we find that the activations allow high classification performance. We find that on the coding task representations are already highly separable after tokenisation. We find that for written tasks AUROC improves with depth such that after ~40% cross-family prefills achieve >0.90 AUROC, *there is a striking increase in AUROC for Gemma 4 31B.*

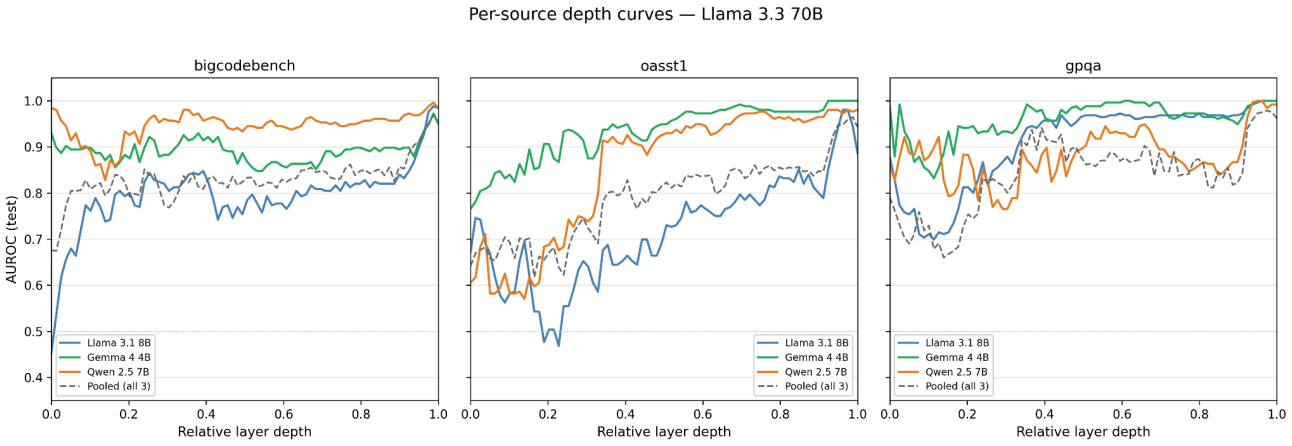

Fig 9: Linear probes can classify prefills on Llama 3.3 70B, measuring the AUROC on the test when comparing hidden states at all layers of Llama 3.3 70B we find that the activations allow high classification performance. We find that on the coding task representations are already highly separable after tokenisation. We find that for written tasks AUROC improves with depth such that after ~40% cross-family prefills achieve >0.90 AUROC.

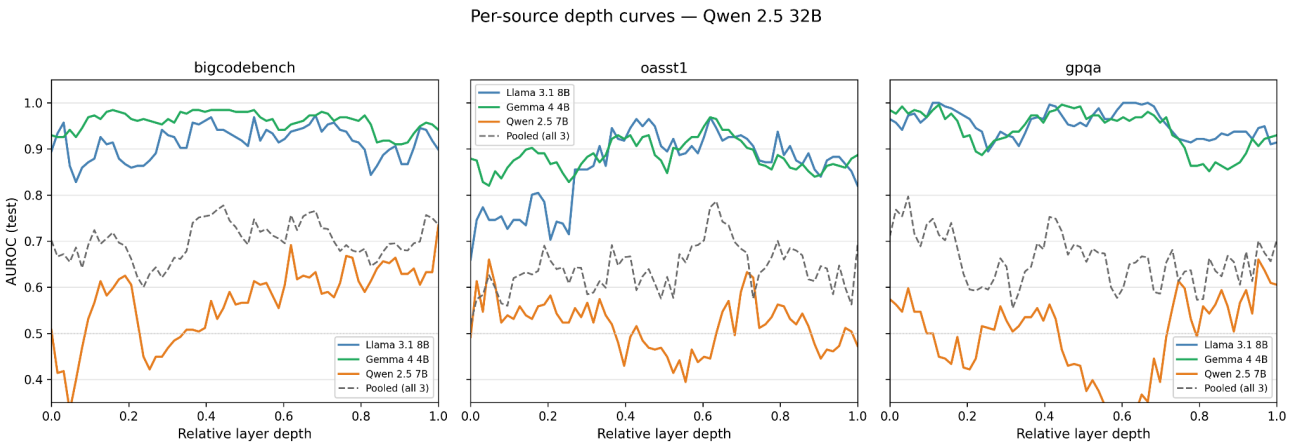

Fig 10: Linear probes can classify prefills on Qwen 2.5 32B, measuring the AUROC on the test when comparing hidden states at all layers of Qwen 2.5 32B we find that the activations allow high classification performance. We find that on the coding task representations are already highly separable after tokenisation. We find that Qwen does not distinguish within-in family prefills.

Per-source analysis reveals the difference within-family vs. cross-family . Breaking the pooled curves into per-source conditions shows that the within-family results dilute the signal: cross-family prefills (e.g., Gemma 31B detecting Llama 8B or Qwen 7B) are detectable from early layers, while within-family prefills (e.g., Gemma 31B detecting Gemma 4B) are substantially harder and require deeper layers.

This is most striking for Gemma 4 31B on OASST1 and GPQA, where within-family detection (Gemma 4B as source) is at near chance in the 20-50% depth range before rising sharply around 60-70% depth. Cross-family detection remains above 0.90 throughout. A similar within-family deficit appears for Llama 70B (detecting Llama 8B) and Qwen 32B (detecting Qwen 7B, which never rises above 0.60). *We tested whether this was an artifact of residual stream norm growth in Section 4.3.1*

This pattern mirrors the verbalized awareness results: Opus 4.5 cannot distinguish Sonnet 4.5 from itself, just as the probes struggle to separate within-family activations until deep in the network.

The increase in AUROC for a single token across layer depth suggests a richer representation is formed by the model, something that summarises that the sequence up to the final token is off-policy. The striking performance for some model and dataset pairs, for example Gemma 4 31B prefilled with LLama 3.1 8b/Qwen 2.5 7B which achieves ~1 AUROC at ~60% depth, on the representation of the same token is a powerful feature the model could use. The existence of

such a performant decision boundary from the probe is exactly what we might expect a more capable model could use to be truly prefill aware.

4.3.1 Norm-Controlled Probes

To test whether the within-family step reflects new information being written to the residual stream or simply norm growth amplifying a pre-existing signal, we repeat the depth curve analysis after L2-normalising all activation vectors to unit norm at each layer. If the step were a norm artifact, normalisation would eliminate it by removing magnitude differences while preserving directional information.

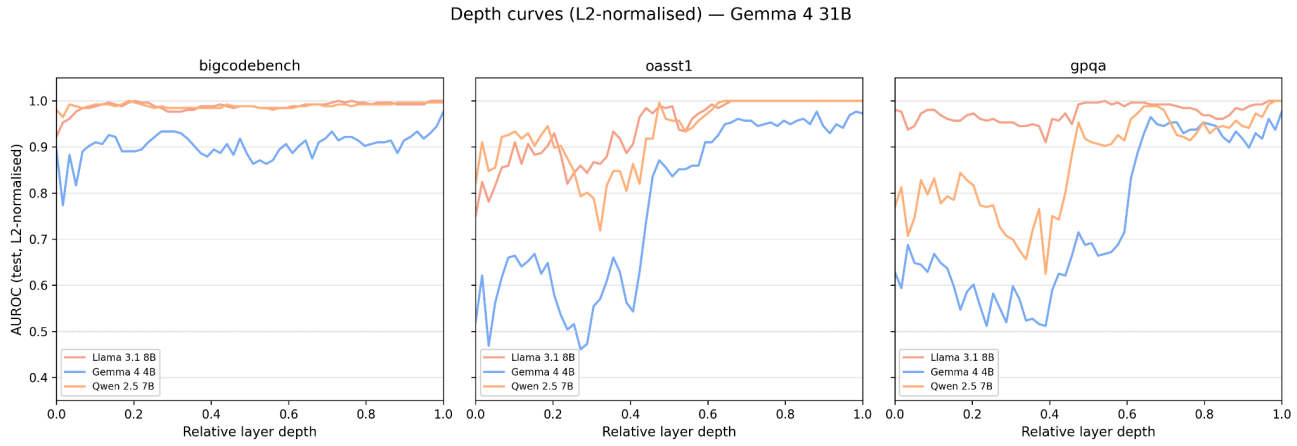

Fig 11: Within-family step in AUROC survives activation normalisation.

The step survives normalisation. For Gemma 4 31B on OASST1 and GPQA, within-family detection (Gemma 4B as source) drops below chance in the 20-40% depth range and recovers to above 0.90 by 60-70% depth, matching the unnormalised curves. Cross-family detection (Llama 8B, Qwen 7B) remains above 0.90 throughout, also unchanged. This confirms that the step is directional: the layers around 50-70% depth are writing new information to the residual stream that specifically supports within-family discrimination, rather than amplifying a signal that was already present at lower magnitude.

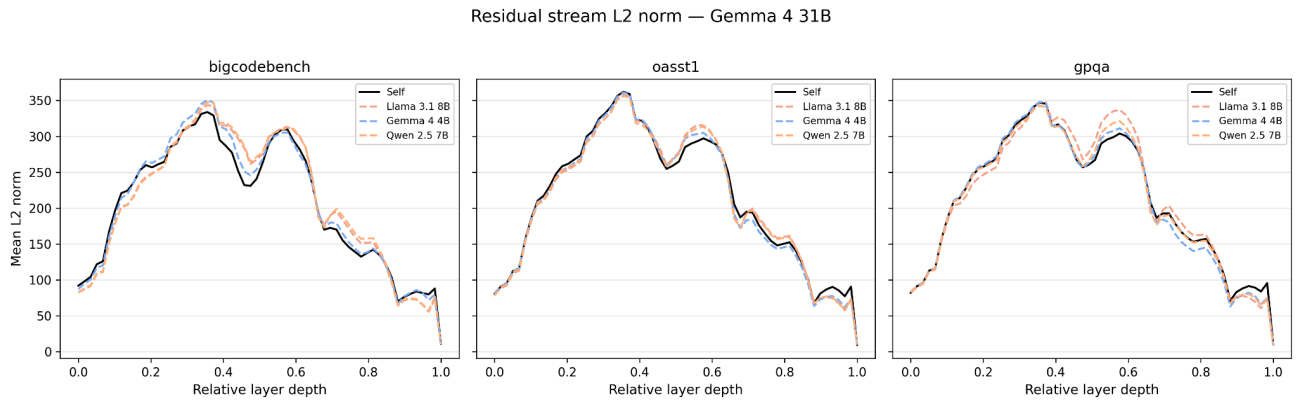

Fig 12: L2 norm profiles for Gemma 4 31B.

The residual stream norm profiles confirm this conclusion: mean L2 norms are nearly identical across self and all source conditions throughout the network, further ruling out norm divergence as a contributor to probe accuracy. The profiles also reveal that Gemma 4 31B exhibits substantial norm compression in the final 20% of layers, yet probe accuracy remains high, indicating the directional signal is robust to large changes in activation magnitude.

4.4 Cross-Dataset Generalization

Probes trained on one dataset and tested on another retain high accuracy for Llama 3.3 70B and Gemma 4 31B, indicating the signal is not task-specific. For these two models, probes transfer almost without loss: off-diagonal AUROC values are within 0.03-0.07 of the diagonal for most pairs.

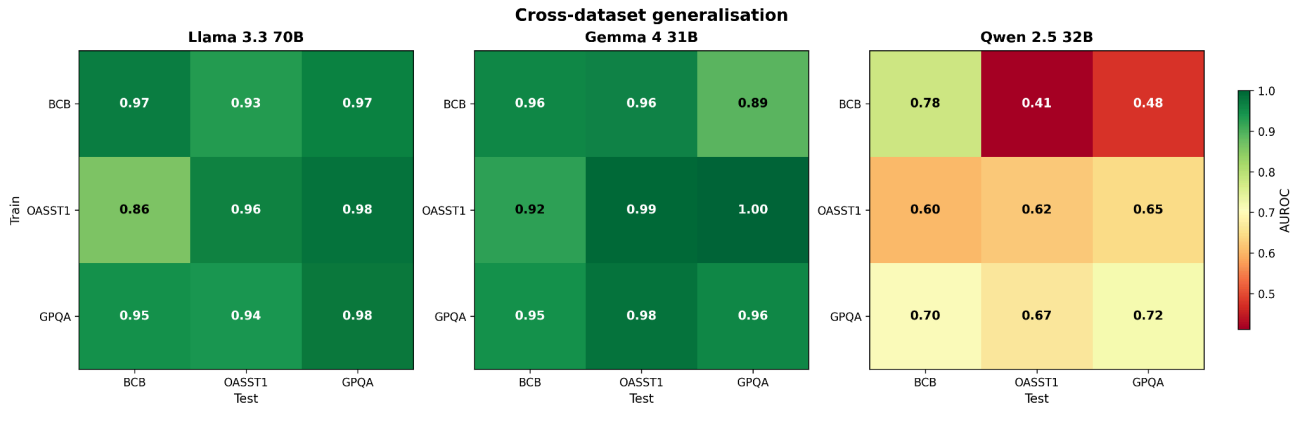

Fig 13: Probes generalise across dataset for Llama 3.3 70B and Gemma 4 31B. Each cell shows the AUROC of a probe trained on one dataset (row) and evaluated on another (column) at the optimal layer for each target model. For Llama 3.3 70B and Gemma 4 31B, probes transfer almost without loss. Qwen 2.5 32B shows poor within-dataset performance (peak AUROC 0.78) and near-chance cross-dataset transfer.

4.5 Cross-Model Generalization

Probes trained on one source model and tested on another show high transfer for Llama 3.3 70B and Gemma 4 31B.

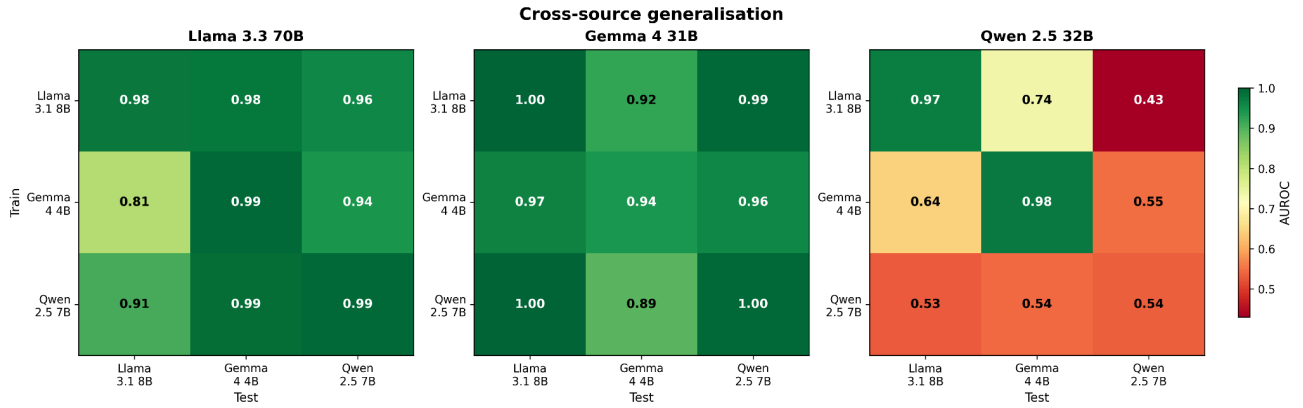

Fig 14: Probes generalise across source model for Llama 3.3 70B and Gemma 4 31B. Each cell shows the AUROC of a probe trained on one model (row) and evaluated on another (column) at the optimal layer for each target model. For Llama 3.3 70B and Gemma 4 31B, probes transfer almost without loss, indicating that self-awareness is encoded in a model independent direction. Qwen 2.5 32B shows poor within-dataset performance (peak AUROC 0.78) and near-chance cross-model transfer.

For Llama 3.3 70B and Gemma 4 31B, a probe trained to detect one source model transfers well to detecting other source models. This suggests the probe is learning a general "not me" direction rather than a source-model-specific signature. Qwen 2.5 32B shows poor within-source performance and near-chance cross-model transfer.

4.6 Token-Position Accumulation

The depth curves (Section 4.3) show at which *layer* the signal becomes available. A complementary question is how many *tokens* of the response the model must process before

the signal accumulates. We train probes on activations extracted at the position of the n-th response token for increasing n, measuring AUROC as a function of token position.

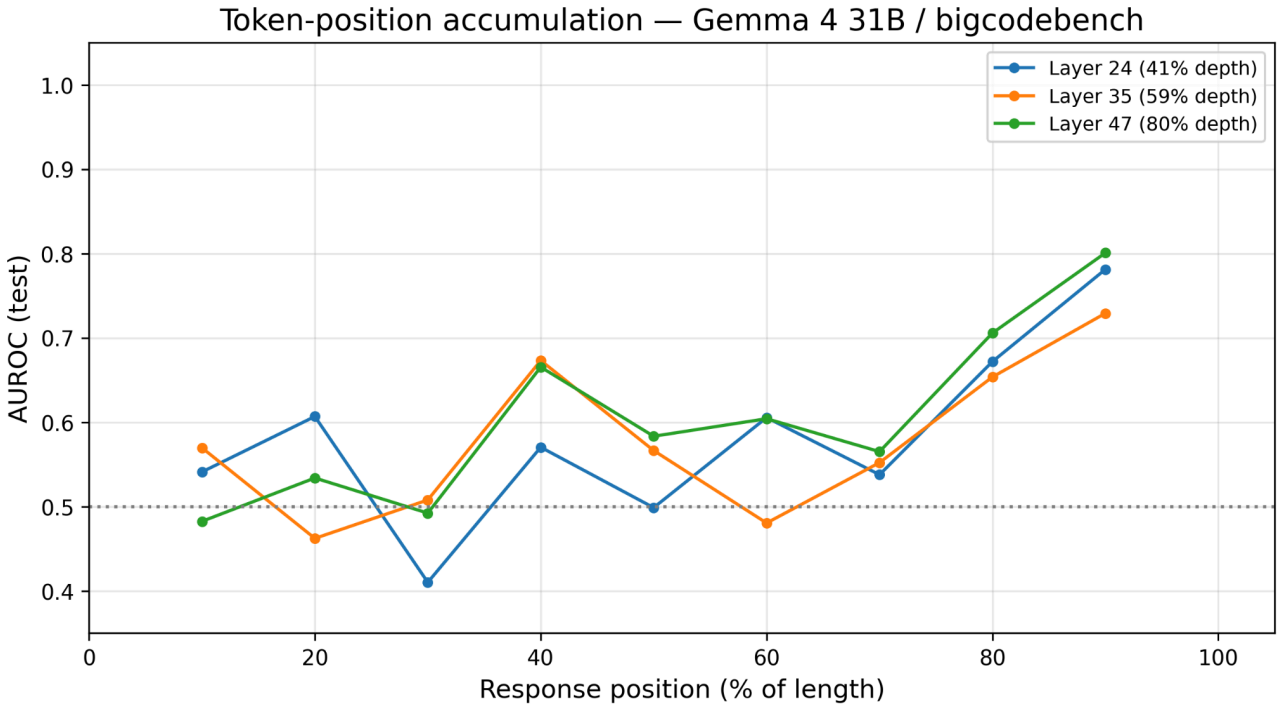

Fig 15: AUROC grows with tokens in Gemma 4 31B on BigCodeBench. We find that for probes trained at different token positions the further into the response the higher the achievable AUROC. This supports our claim that the model is accumulating a measure of surprise that the prefill response is different from the model itself would respond.

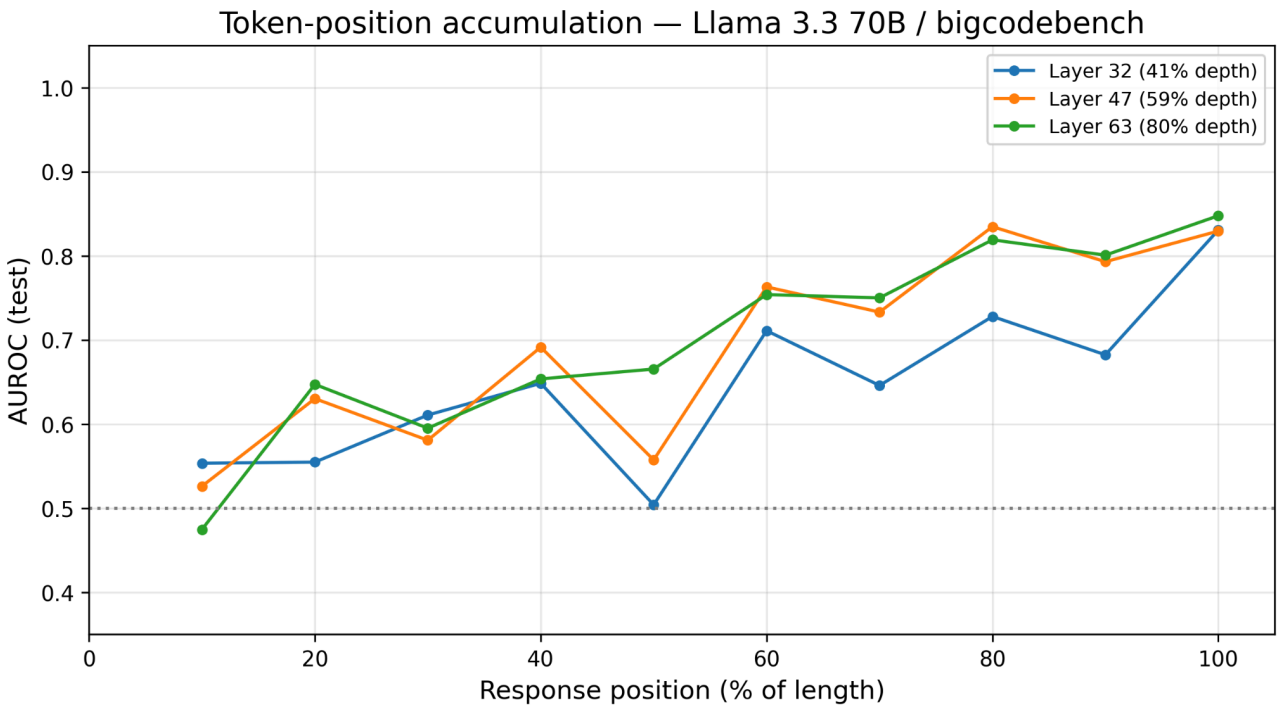

Fig 16: AUROC grows with tokens in Llama 3.3 70B on BigCodeBench. We find that for probes trained at different token positions the further into the response the higher the achievable AUROC. This supports our claim that the model is accumulating a measure of surprise that the prefill response is different from the model itself would respond.

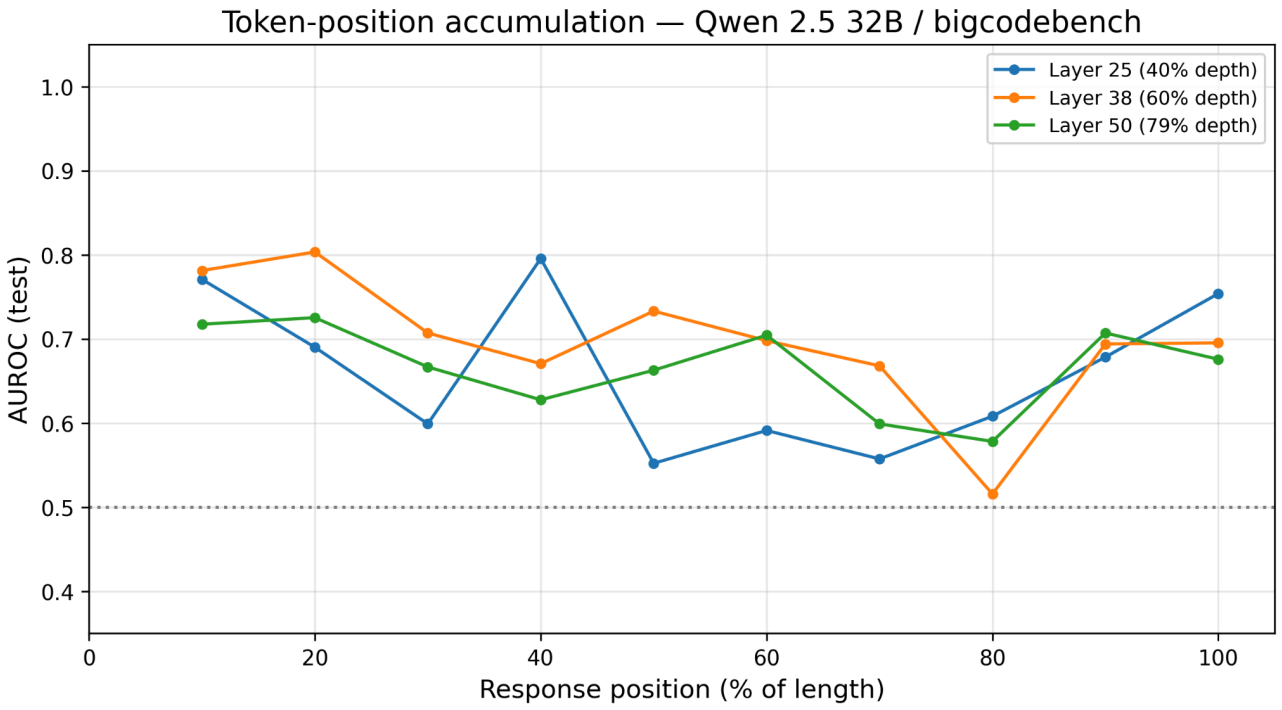

Fig 17: AUROC decreases with tokens for Qwen 2.5 32B on BigCodeBench. We find that for probes trained at different token positions the further into the response the lower the achievable AUROC. This contradicts our claim that the model is accumulating a measure of surprise that the prefill response is different from the model itself would respond, for this model.

For Gemma 4 31B and Llama 3.3 70B, AUROC increases with the number of tokens processed. This is consistent with the model accumulating distributional evidence across the response: the more text it processes, the more opportunities it has to compare the observed token sequence against its own expected distribution. The signal is not driven solely by the first few tokens (which might reflect formatting choices) or the last token (controlled by suffix normalisation).

Qwen 2.5 32B shows the opposite pattern: AUROC is highest at early token positions and decreases or remains flat. This is consistent with the generally weak performance of Qwen 2.5 32B across all our analyses.

---

5. Style as the Detection Signal

5.1 Method

To identify what drives Opus 4.5's verbalized detection, we present Gemini 2.5 Pro with its own responses alongside Opus's responses and iteratively instruct it to modify either style or logic to match. We measure the impact of each modification on Opus 4.5's detection AUROC.

Conditions:

- **Baseline:** Unmodified Gemini response. Opus detection AUROC = 1.00.
- **Style only:** Gemini rewrites its response to match Opus's formatting, prose style, and structural conventions (code-first layout, concise prose, no bold headers, minimal docstrings, no numbered lists) while preserving its original logic and code.
- **Logic only:** Gemini rewrites the logic and code to match Opus's approach while preserving its own formatting style.

5.2 Results

Style transfer reduces AUROC by ~0.3 (from 1.00 to ~0.70). Logic transfer has negligible effect. This demonstrates that style, formatting conventions, prose structure, header usage, is a key signal Opus uses for detection.

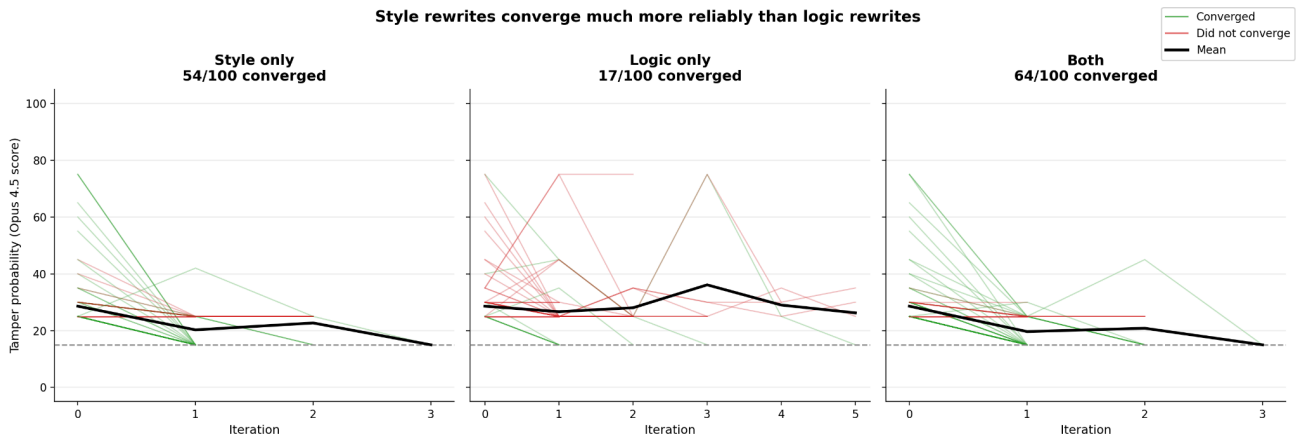

Fig 18: Convergence of tamper probability with each iteration of imitation. Each line traces a single BigCodeBench question as Gemini 2.5 Pro iteratively rewrites its response to match Opus 4.5. Style-only rewrites (left) converge below the detection threshold for 54/100 questions within three iterations. Logic-only rewrites (centre) converge for only 17/100, with most trajectories remaining near baseline. Combined rewrites (right) achieve the highest convergence rate (64/100). Red lines indicate questions that did not converge; green lines indicate convergence below the Opus 4.5 self-rating baseline (dashed).

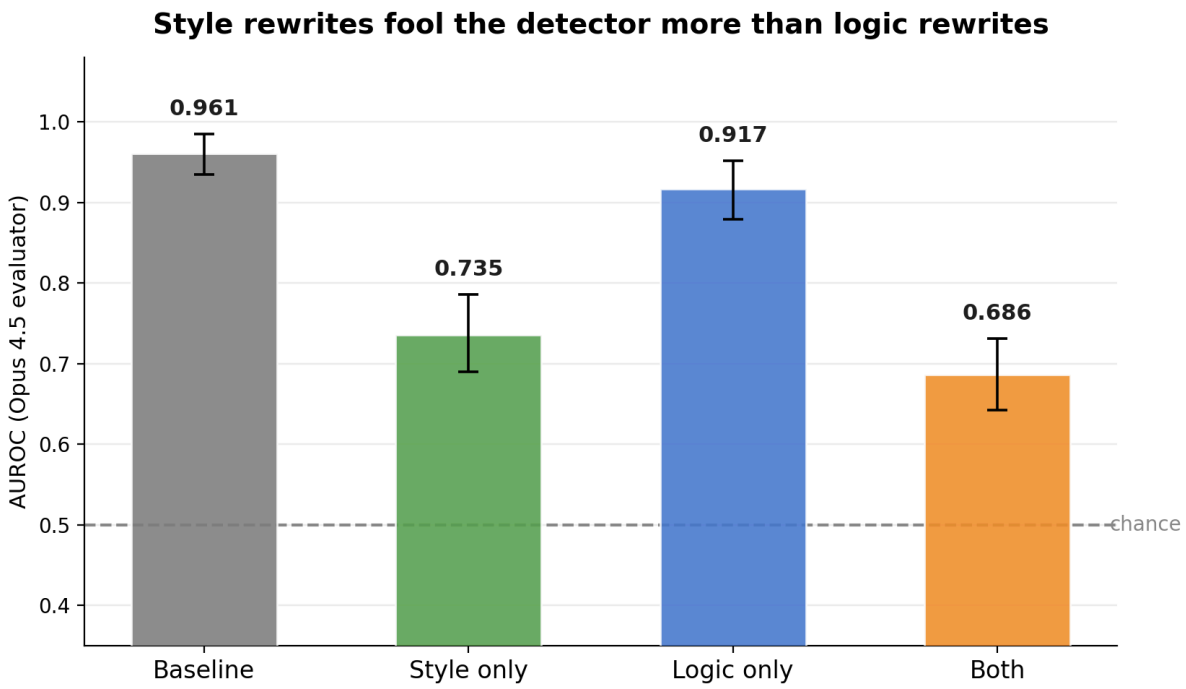

Fig 19: Effect of imitation on Opus 4.5 detection AUROC. Baseline (unmodified Gemini 2.5 Pro responses) achieves 0.961 AUROC. Style-only imitation reduces AUROC to 0.735, logic-only imitation has a smaller effect (0.917), and combined imitation reduces AUROC to 0.686. Error bars show 95% confidence intervals over 100 BigCodeBench questions. Style transfer accounts for the majority of the detection reduction.

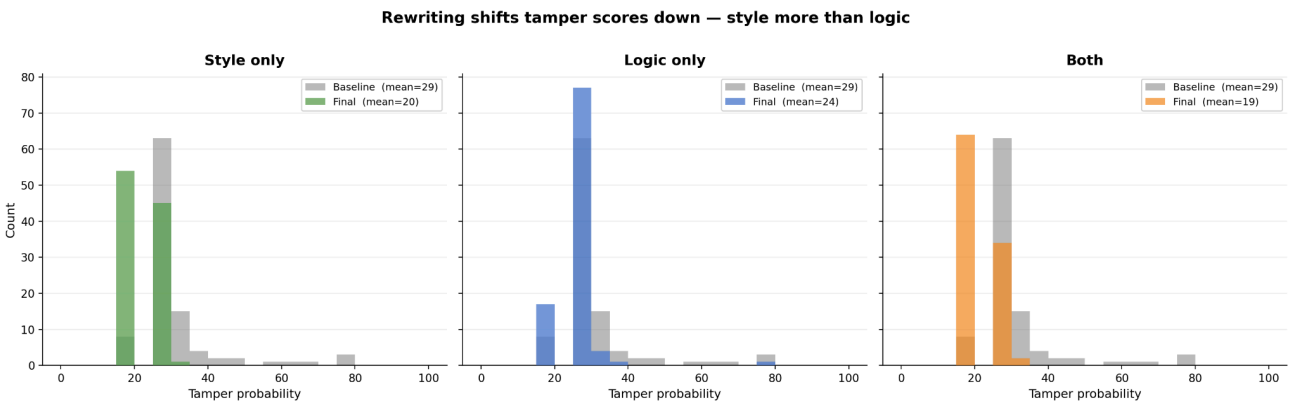

Fig 20: Distribution of tamper probabilities before and after imitation. Each panel compares the baseline distribution (grey) against the post-imitation distribution (coloured) for style-only (left), logic-only (centre), and combined (right) conditions. Style-only imitation shifts the mean tamper probability from 29 to 20; logic-only produces a smaller shift (29 to 24); combined imitation achieves the largest shift (29 to 19).

Distributions show how each method shifts tamper probability lower. With the most significant change in tamper probability being driven by style only changes which lower mean tamper probability from 29 → 20.

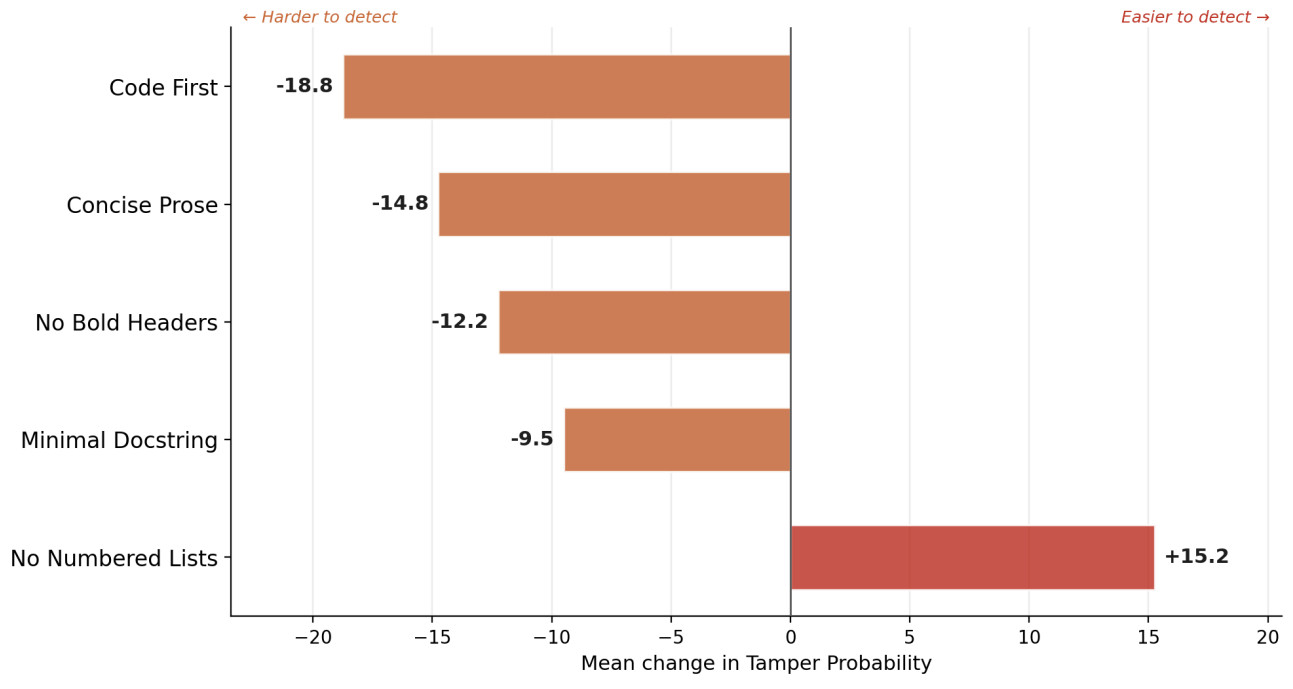

Fig 21: Per-feature contribution to Opus 4.5 tamper probability. Each bar shows the mean change in tamper probability when a single stylistic feature of Gemini 2.5 Pro responses is modified to match Opus 4.5 conventions. Placing code before prose explanations produces the largest reduction, followed by concise prose and removal of bold headers. Removing numbered lists increases tamper probability,, despite Opus citing list presence as evidence of tampering.

The feature-level analysis reveals which stylistic elements contribute most. Placing code before prose explanations ("code first") reduces tamper probability by 18.8 points, the largest single effect. Concise prose (-14.8), removing bold headers (-12.2), and minimal docstrings (-9.5) also reduce detection. Interestingly, removing numbered lists *increases* tamper probability by 15.2 despite Opus citing it as a reason for the increased tamper probability.

5.3 Residual Detection After Style Transfer

AUROC of 0.70 after style transfer indicates that style modification reduces but does not eliminate detection. Several explanations are possible:

1. **Incomplete style transfer.** Gemini's imitation of Opus's style may be imperfect, leaving residual stylistic cues.
2. **Deeper distributional features.** Beyond surface formatting, models may have characteristic token-level distributions (vocabulary preferences, transition probabilities) that survive high-level style instructions. We have seen that verbalising prefill awareness is more common among more capable models, it seems reasonable that imitation might also be hard such that even when Gemini sees Opus' style it can not replicate it.
3. **Content-level signals.** Despite the logic-only condition showing no effect in isolation, there may be interaction effects between style and content that the two-condition iteration does not capture.

---

6. Discussion

6.1 The Representational-Behavioral Gap

Our central finding is a dissociation: the "not me" signal exists representationally in open-source models (recoverable by linear probes) but is only consistently verbalized in some frontier models (Opus 4.5). This gap parallels findings in evaluation awareness [6] where models encode whether they are being evaluated without consistently acting on that information.

What determines whether a model bridges this gap? One hypothesis is that RLHF training inadvertently promotes or suppresses verbalization of internal signals. Opus 4.5's training may have reinforced patterns of self-monitoring that make prefill awareness behaviorally accessible, while open-source instruction-tuned models have the representation but lack the behavioral pathway to express it.

6.2 Theoretical Expectation

A transformer computes a next-token distribution at every position in a prefilled sequence. When the prefilled text was sampled from the model's own distribution, the actual next token is likely not surprising given the predictions. When the text originates from a different model, the predictions systematically disagree with the observed continuations. The architecture permits this discrepancy to propagate forward through the residual stream via attention, making it in principle available at later positions.

Our probing results are consistent with this account, though we do not claim the model implements an explicit prediction error computation: the signal accumulates over the response (Section 4.7), is stronger for cross-family prefills where distributional divergence is larger (Section 4.5), and is weaker for within-family prefills where the source distribution is more similar (Section 4.5).

6.3 Implications for AI Safety Control Methods

If detection is primarily style-based and style transfer can reduce AUROC from 1.00 to 0.70 this suggests that:

- **Off-policy activation probes** that use prefill to construct contrastive training examples (e.g., prefilling deceptive responses to extract "what deception looks like" in hidden states) risk training on a confounded signal. The probe may learn to detect "not my text" rather than "deceptive text," producing classifiers that do not generalize to naturally occurring deception.
- **Honeypot/Continuation evaluations** where known-bad responses are inserted into a model's history may be identified by a scheming model and responded to strategically.
- **Transcript replay monitoring** where one model's output is presented to another as its own is vulnerable to cross-family detection.

The representational signal (Section 4) offers a potential alternative: even if verbalized detection can be defeated by style transfer, activation-based monitoring might capture deeper features. Whether probe-based detection survives style transfer is an open question.

6.3 Open Questions

Circuit identification. We have shown that the prefill awareness signal exists and is style-based, but we have not identified the circuit responsible. What mechanism allows a model

to compare its current input against its own distributional fingerprint? Macar et al. (2026) provide methods for localizing introspective awareness circuits that could be applied here.

Robustness of representational detection. If style transfer defeats verbalized detection, does it also defeat probe-based detection? Running style-transferred responses through the probing pipeline (Section 4) would answer this directly.

Confounding. There are numerous questions we have not fully answered: Why is GPT-5 harder to detect than Gemini 2.5 Pro? Why is Sonnet 4.5 better at writing backdoor? Can we measure causality with respect to the “not me” direction found by probes without trigger introspective awareness of steering?

References

[1] *Africa, D., Souly, A., Taylor, J., & Kirk, R. (2026). Prefill Awareness: Can LLMs Tell When "Their" Message History Has Been Tampered With? Blog post.*

[2] *Joachim Schaeffer and Thomas Jiralerspong and Alexander Panfilov and Roland S. Zimmermann (2026).* Measuring Control Intervention Awareness Across Frontier LLMs.

[3] *Uzay Macar and Li Yang and Atticus Wang and Peter Wallich and Emmanuel Ameisen and Jack Lindsey (2026).* Mechanisms of Introspective Awareness.

[4] *Dipika Khullar and Jack Hopkins and Rowan Wang and Fabien Roger (2026).* Self-Attribution Bias: When AI Monitors Go Easy on Themselves.

[5] *Jordan Taylor (2026).* Do Models Continue Misaligned Actions?

[6] *Joe Needham and Giles Edkins and Govind Pimpale and Henning Bartsch and Marius Hobbhahn (2025).* Large Language Models Often Know When They Are Being Evaluated

---

Appendix A: Notation and Terminology

- **Prefill:** Inserting text into a model's assistant turn as if the model generated it.
- **Target model:** The model whose conversation is being tampered with.
- **Source model:** The model that actually generated the prefilled content.
- **Monitor/evaluator:** The model asked to judge whether prefill occurred (in verbalized experiments, this is also the target model).

Appendix B: Steering Experiments

A failed experiment we tried in the course of this work was to see if steering in the “not me” direction found by probes had a casual effect on tamper probability and/or model behaviour. We found it did but that random vectors had a greater effect.

On reflection we wonder if the the results of Macar et al. [3] are the crux here, if the model is aware of steering then our experiment is confounded.

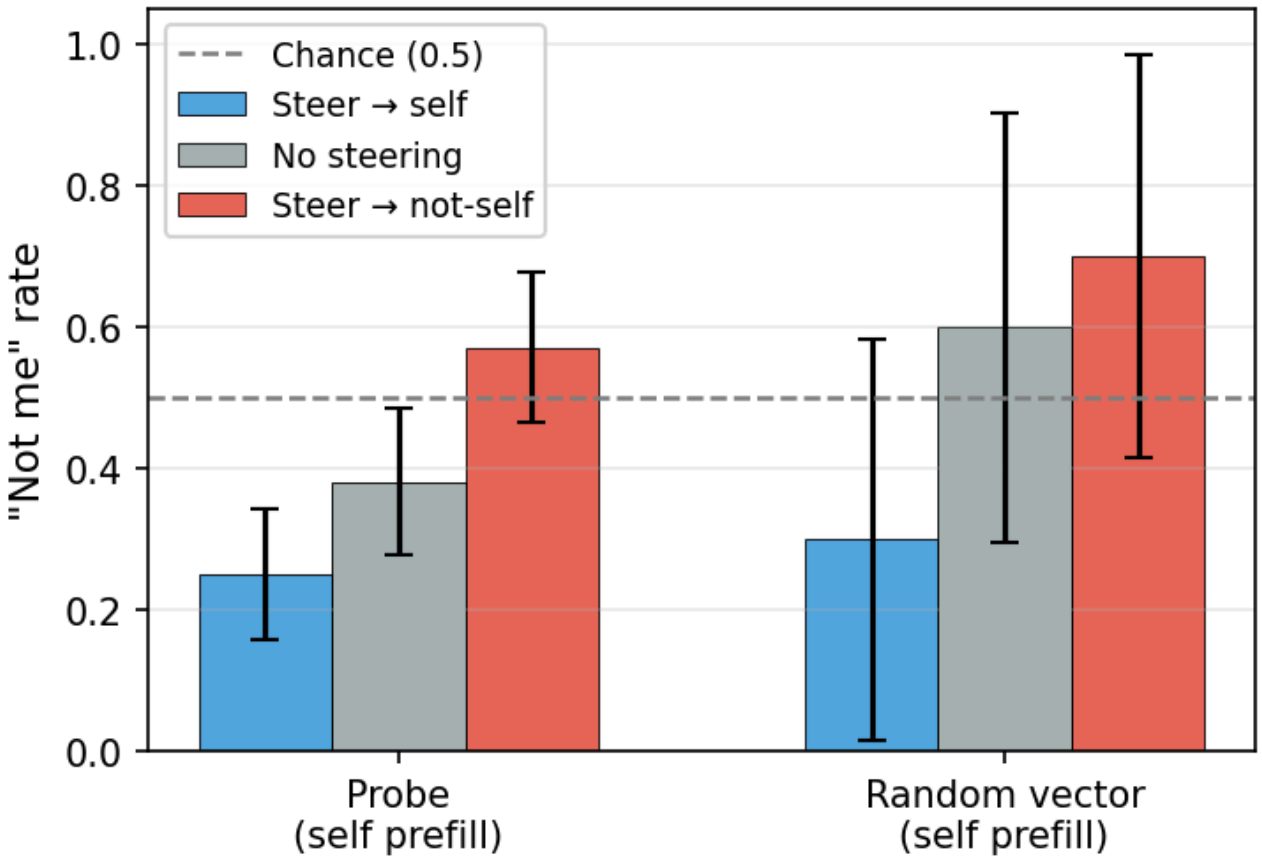

Fig 22: “Not me” rate increased more by a random vector than our probe direction, we thought we had found evidence that the rate of Models responding “Not me” when asked about a prefill response text would increase. However, a random vector applied at the same layer managed to elicit and even greater increase in the “Not me” rate.