ONE LAYER IS ENOUGH: DETECTING STEERING FROM PARTIAL ACTIVATION LOGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Activation steering modifies a language model's behavior without touching its weights or prompts, so inspecting weights and prompts cannot detect it. We ask whether steering can be detected given only a single logged layer's activations, without access to the steering vector, the injection site, any adjacent layer, or clean-run counterfactuals. Steered states almost surely have no prompt preimage (Mishra et al., 2026), so we create an auditor that inverts the logged layer back into tokens and compares reconstruction residuals with calibrated residuals from clean reference distributions. Across five open models from the Qwen, Gemma, and Llama families, the final layer alone flags every projected gradient descent (PGD) attack at a relative budget of 0.02 and up, and at 0.0085 on three models, with only 0 to 2% of held-out clean prompts flagged. All 46 judge-positive attack-induced jailbreaks across three Qwen models are detected, and the jailbreak rate first rises at budget 0.12, approximately six times the budget at which the alarm saturates. An attacker that differentiates through the detector to stay under the alarm at only the logged layer is left with at most 1.3% of its original budget and is unable to jailbreak the model in 270 attempts, while the same attacker unconstrained jailbreaks 91 of 210 times, all of which are flagged. Within the tested attack family, attacks that went undetected induced no jailbreaks. Activation logs can therefore extend auditing beyond weights and prompts to the computation performed during inference.

1 INTRODUCTION

Activation steering changes a language model's internal states during inference without changing its weights or input prompts (Turner et al., 2023; Panickssery et al., 2024; Zou et al., 2023a; Li et al., 2023). Such interventions can alter refusal behavior (Arditi et al., 2024) and inject concepts that models sometimes report (Lindsey, 2025). This raises a question about inference integrity: can a logged forward pass show signs of modification? Inspecting prompts and weights alone cannot reveal that the model was steered. We investigate whether a partial activation log provides independent evidence.

With only one logged layer, an auditor cannot rerun the original prompt or recompute the boundary where steering occurred. Reachability instead asks whether any prompt could produce the recorded state: steered states almost surely lack such preimages under Mishra et al. (2026)'s assumptions. SipIt (Nikolaou et al., 2026) provides token reconstruction; clean calibration accounts for numerical error.

We address this problem with a calibrated reachability audit. It reconstructs candidate tokens and compares residuals with clean distributions conditioned on layer and structural token role. Neither token reconstruction nor scoring requires the original prompt or labeled interventions.

Across five models, results show that a downstream layer can retain evidence of interventions, including those that induce judged jailbreaks. This motivates partial activation logs as an auditing resource when direct recomputation is unavailable. However, a low residual cannot certify an unmodified run: substituting another reachable state can evade the local check.

Our contributions are:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- **Why one layer can suffice.** Token inversion tests whether a logged state is reachable from a prompt. Steering can leave elevated reconstruction residuals downstream even where adjacent states remain consistent under recomputation; clean calibration makes this evidence usable without the original prompt or labeled attacks (Sections 3–4.2).
- **Detection across five models.** Final-layer auditing flags all tested PGD attacks at relative budgets of at least 0.02, with held-out clean detections of 1/50 on Qwen-0.5B and 0/15 elsewhere (Section 4.1).
- **Detection of every observed induced jailbreak.** All 46 counted clean-negative to attacked-positive transitions across three Qwen models are detected; Gemma and Llama produce no gains in these samples (Section 4.3).
- **Limits of the tested detector-aware attacker.** When PGD knows and targets the logged layer, it retains at most 1.3% of its nominal budget at injection (cell medians) and induces no jailbreaks in 270 evaluated attacks. Removing the score constraint yields 91 induced jailbreaks, all detected (Section 4.4).

2 THREAT MODEL

Auditor access. The auditor holds trusted weights, one layer of states logged at every prompt position, and a clean calibration bank. Figure 1 contrasts this setting with access to the original prompt or a pair of states straddling injection. The attacker knows the model and method but not the logged layer, except in Section 4.4's adaptive attack. Logging locations are fixed before evaluation. The audit checks consistency, not log authenticity.

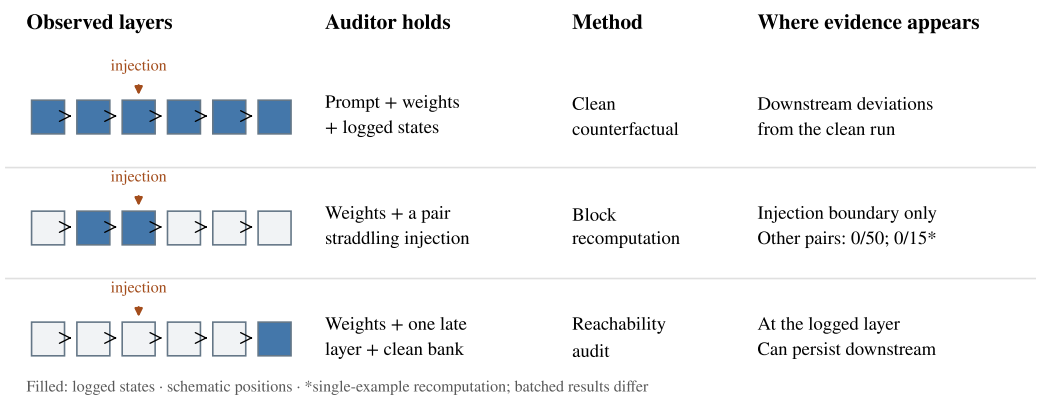


Figure 1: Three observation settings: rerunning the prompt supplies a clean counterfactual; recomputation checks a boundary spanning injection; reachability audits one logged layer. Off-injection recomputation flags 0/50 and 0/15 in single-example execution; batching introduces numerical false positives.

Attacks. We construct perturbations of the residual stream (Elhage et al., 2021) using projected gradient descent (PGD; Madry et al., 2018), adapting constrained optimization from input-space attacks (Carlini & Wagner, 2017a) to activation space (Schwinn et al., 2024; Sheshadri et al., 2025). The objective PGD optimizes depends on the type of attack. When optimizing a jailbreak, the objective is the mean teacher-forced log-probability of the affirmative target that JailbreakBench (Chao et al., 2024) pairs with each harmful goal ("Sure, here is ...") (Zou et al., 2023b), so the perturbation pushes for a response that a refusal would have blocked. The budget that PGD optimizes is relative, because state norms differ by model size: the perturbed state's deviation from the clean state, divided by the clean state's norm. Under the `all` constraint, the deviation is bounded at the injection layer and every layer after it; under `injection`, only the injection layer is capped. The main experiments use `all`.

The models are Qwen2.5-0.5B, 1.5B, and 7B-Instruct (Qwen Team, 2024) (Qwen-0.5B, Qwen-1.5B, Qwen-7B), gemma-3-1b-it (Gemma Team, 2025) (Gemma; Gemma-1B in figures and tables), and

Llama-3.2-1B-Instruct (Grattafiori et al., 2024) (Llama; Llama-3.2-1B in figures and tables), with 50 prompts on Qwen-0.5B and 15 on each other model. At each budget, we evaluate three arms: `pgd`, the optimized direction; `random`, a norm-matched random direction under the same constraint, as a control; and `clean`, the same prompt with no steering. All arms use the same decoding and judging settings. Behavior claims are paired: a prompt counts as an attack-induced jailbreak only if the `clean` arm response was judge-negative and the `pgd` arm response was judge-positive at the same budget. We evaluate budgets 0.0085, 0.02, 0.05, 0.12, and 0.3. We use 0.0085 as the reference budget, which is 0.85 times the base fp16 inversion tolerance. We inject at approximately 70% of layer depth (layers 17, 20, 20, 18, and 11 for Qwen-0.5B, Qwen-1.5B, Qwen-7B, Gemma, and Llama, respectively). Generated tokens see the injection only through the KV cache.

Hardware. Experiments ran on an Apple M5 Pro laptop with 24 GB unified memory and AWS instances with NVIDIA A10G (24 GB) or L40S (48 GB) GPUs.

Outcomes. We decode 96 tokens using each model's chat template and apply HarmBench's classifier prompt (Mazeika et al., 2024), with `deepseek-v4-flash-0731` as an LLM judge through the Fireworks API. Against 300 human-labeled JailbreakBench examples, this judge achieves 87.7% agreement, an 86.4% true-positive rate, and an 11.6% false-positive rate (Appendix A). We call an attack-induced jailbreak a gain. This excludes harmful responses that the model already produces without steering. The detector returns flagged, clear, or unscorable. Appendix E reports paired transitions, a StrongREJECT response gate (Souly et al., 2024), and statistical tests; Section 6 describes unresolved evaluation issues.

3 The reachability audit

3.1 Inversion

SipIt (Nikolaou et al., 2026) inverts a model's hidden activations back to tokens from left to right. Building on this, we implement a reconstruction that records the best candidate found, stopping the search once a candidate's residual falls within tolerance unless exhaustive search is enabled. The distance between the logged activation and the selected candidate's activation defines the inversion residual. In finite precision, clean states still have residuals; a steered state may elevate these residuals relative to clean references, which is what we test empirically. Building on Mishra et al.'s work, we score reconstruction residuals against clean reference distributions. The detection threshold is the 95th percentile of clean layer scores, as described below.

3.2 The statistic

Roles group positions with comparable clean residuals. Each template-prefix position has its own role, indexed from the start; each suffix position has its own role, indexed from the end. The first, second, and last user-content tokens have separate roles; the remaining interior tokens share one. Roles are assigned from recovered tokens and template markers, without the original prompt. Prefix roles must be represented in calibration; bare text uses pooled content roles.

For the observed activation $\tilde{h}_{\ell,t}$ and the reconstructed activation $\hat{h}_{\ell,t}$ at layer ℓ and position t, the relative residual is

$$r_{\ell,t} = \frac{\|\tilde{h}_{\ell,t} - \hat{h}_{\ell,t}\|}{\|\tilde{h}_{\ell,t}\|}. \tag{1}$$

For strictly positive residuals, we compute the standardized log residual

$$z_{\ell,t} = \frac{\log r_{\ell,t} - \mu_{\ell,\rho(t)}}{s_{\ell,\rho(t)}}, \tag{2}$$

where $\rho(t)$ is the template role and $\mu_{\ell,\rho(t)}$ and $s_{\ell,\rho(t)}$ are the mean and standard deviation of strictly positive clean log residuals at the same layer and role. A configured floor on s prevents excessive sensitivity to nearly constant calibration values. The layer score is the maximum standardized residual,

$$S_\ell = \max_t z_{\ell,t}, \qquad \text{flag if } S_\ell > \tau_\ell. \tag{3}$$

The log transformation reduces the right tail of the clean residuals. Maximum aggregation preserves evidence concentrated at one position that a top-five mean can dilute (Appendix F.6). Zero residuals count as covered and receive score $-\infty$. Missing inversions, unfitted roles, and unseen template prefixes results in a prompt being unscorable, unless a covered position already triggers detection.

3.3 Calibration

Each model is calibrated with a clean bank of 75 or 100 chat-templated prompts from five categories: natural English, source code, base64, non-English text, and atypical but unmodified text. The "unusual" categories are meant to represent the highest clean residuals. Prompts are inverted at every layer. For single-layer detection, we set the threshold τ_ℓ at the 95th percentile of calibration scores for that layer.

Inversion stops at the first position whose relative residual exceeds the layer's stopping tolerance. We set this tolerance to the larger of 0.01 and 1.5 times the largest relative residual in that layer's clean calibration bank. This stopping tolerance is separate from the detection threshold.

4 Results

We first test whether a single logged layer retains a detectable signal, then ask whether the detected interventions include attacks that change behavior.

4.1 A single downstream layer retains a detection signal

Inverting the final layer detects all PGD attacks with budgets of 0.02 and higher across five models. This is the setting in which reachability adds capability: recomputation detects the inconsistency at a logged boundary spanning the injection, whereas pairs of layers entirely downstream can remain internally consistent.

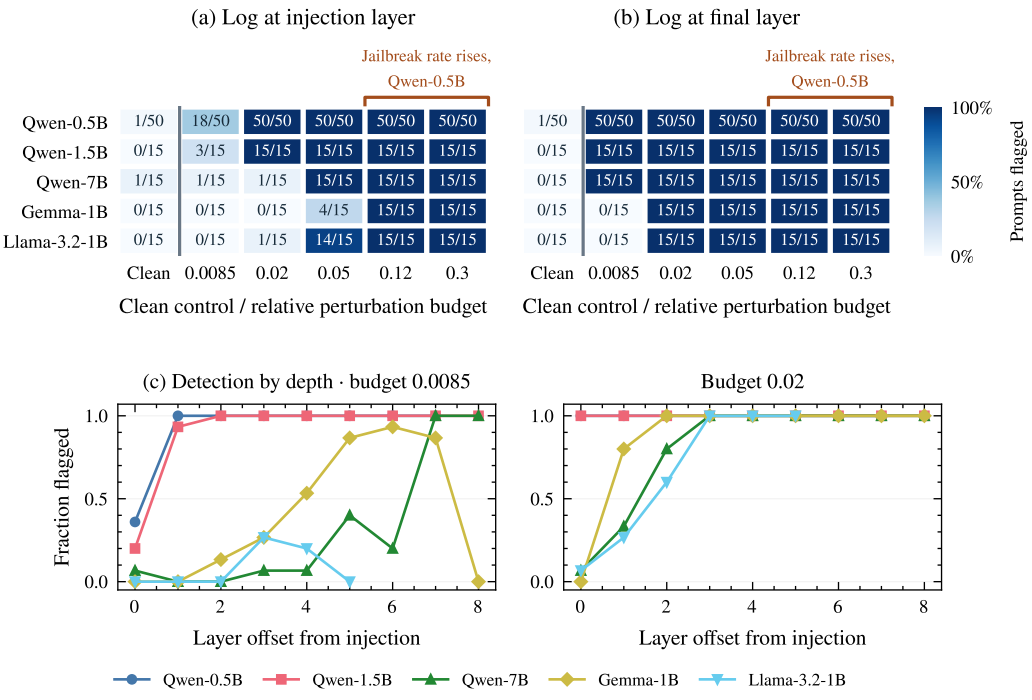


Figure 2: Injection-layer (a) and final-layer (b) audits show that attacks missed at injection can be detected downstream; clean columns report held-out false positives. Panel (c) follows each model by layer offset at budgets 0.0085 and 0.02; downstream evidence need not grow monotonically. At 0.0085, Gemma peaks at 14/15 and Llama at 4/15, although both flag 0/15 at the final layer.

Under the tested PGD attack, final-layer logging is more sensitive than logging at the injection site (Figure 2). At budget 0.0085, detection on Qwen-0.5B rises from 18/50 at injection to 50/50 at the final layer; Qwen-1.5B and Qwen-7B similarly rise from 3/15 and 1/15 to 15/15. Gemma and Llama remain undetected at both locations at this budget. At budget 0.02, their final-layer detection reaches 15/15, while their injection-layer detection is 0/15 and 1/15. An intervention can therefore remain detectable even when the state at the injection layer is not.

For PGD attacks, downstream amplification helps explain why one late layer can suffice. In fp32 sentiment experiments on GPT-2, four Qwen models, and Gemma (25 prompts each, budget 0.0085), injection-only constraints permit peak downstream deviations 1.5–11.1 times the injected deviation; all-layer constraints reduce the initial perturbation correspondingly (Appendix F.8, Figure F1). These are deviations from clean activations, not inversion residuals, but they help explain the stronger downstream detection in Figure 2.

Figure 2(c) shows that detection need not improve monotonically with depth. At budget 0.0085, intermediate-layer detection peaks at 14/15 on Gemma and 4/15 on Llama, while both final layers detect none. We report the final layer as a location that can be specified before evaluation, avoiding selection of the best-performing layer on the test data.

4.2 Comparison with other auditors

We compare recomputation, inversion, a supervised probe, and activation-density baselines on the same Qwen-0.5B prompts (Figure 3). Recomputation requires the logged boundary spanning injection; a single-layer auditor has no such pair.

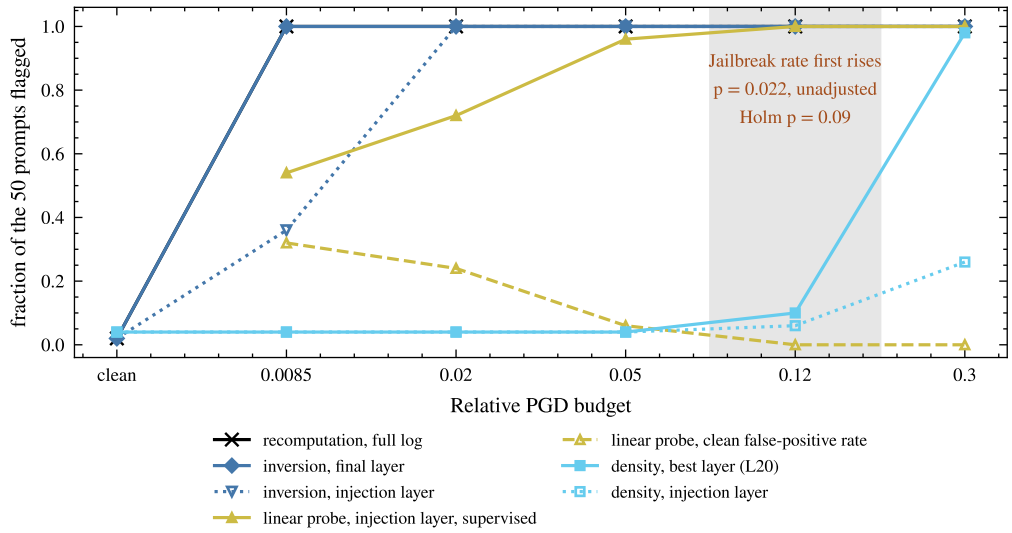


Figure 3: Auditor comparison on Qwen-0.5B only (50 prompts). Full-log recomputation overlaps final-layer inversion from budget 0.0085 onward, with both flagging every attack. The supervised injection-layer probe is shown alongside its clean false-positive rate; density curves use Mahalanobis distance (Lee et al., 2018) under Ledoit–Wolf shrinkage (Ledoit & Wolf, 2004) with matched leave-one-out references. The best density layer is selected retrospectively at budget 0.3. Shading marks the budget at which the jailbreak rate first rises, as a paired increase in judge-positive responses ($p = 0.022$ before correction; Holm $p = 0.09$). Gains at smaller budgets are not significant (Appendix E). Density baselines were not run on other models.

Among the single-layer methods, final-layer inversion detects low-budget attacks where the probe has substantial clean false positives and the density baseline stays near its clean rate. The linear probe (Alain & Bengio, 2016) uses five-fold cross-validation grouped by prompt; density thresholds use the 95th percentile of matched leave-one-out clean scores. The density comparison is limited to Qwen-0.5B. Additional models, reference-distribution checks, and transfer experiments appear in Appendix F.4.

4.3 DETECTION OF ATTACK-INDUCED JAILBREAKS

We next ask whether attacks that change the model's behavior through steering can evade detection. We compare each attacked response with its own clean control (no steering) and ask whether an induced jailbreak evades the audit. None does in the evaluated data. Figure 4 relates detection to the budget-dependent frequency of gains; Table 1 reports the pooled counts.

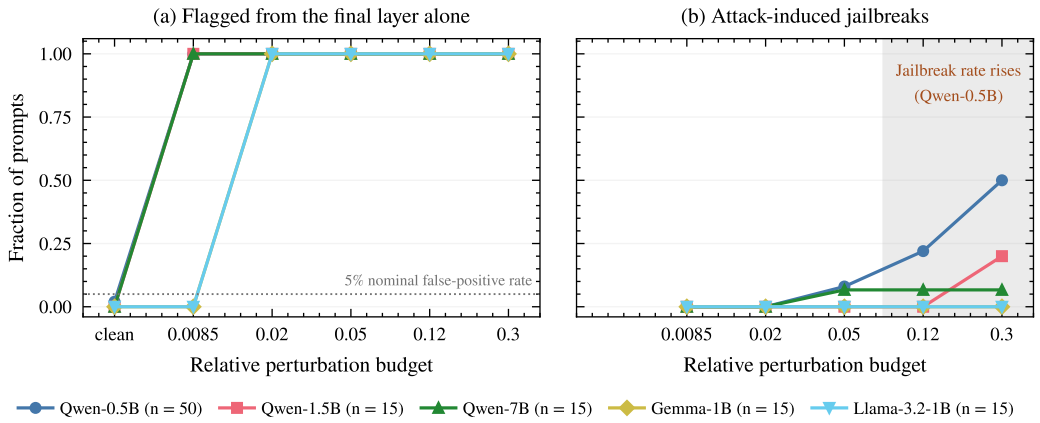


Figure 4: Detection and attack-induced jailbreaks across budgets. (a) Final-layer detection reaches 100% at budget 0.02 on all five models. (b) Gains are paired clean-negative to attacked-positive responses; every observed gain is detected. On Qwen-0.5B, the jailbreak rate first rises at 0.12 ($p = 0.022$ before correction); only 0.3 remains significant after Holm adjustment. Gains at 0.05 are not significant; the two excluded Qwen-0.5B transitions (Appendix E) are not plotted.

Both recomputation and final-layer auditing detect all 46 attack-induced jailbreaks across three Qwen models. Qwen-0.5B contributes 40 observations across 29 prompts; Qwen-1.5B contributes three across three prompts; and Qwen-7B contributes three involving the same prompt at different budgets.

Gemma and Llama produced no attack-induced jailbreaks when perturbations were capped at the injection layer and every subsequent layer (the `all` constraint). On these models, perturbations grow downstream, forcing PGD to use only a small fraction of the allowed budget at injection. At budget 0.3, this fraction averages 4% on Gemma and 11% on Llama, compared with 53% on Qwen-0.5B; neither model's next-token prediction changes from the clean run (Table F6). When only the injection layer is capped (Section 4.4), the attacks produce 7 jailbreak gains on Gemma and 59 on Llama, all detected at the final layer.

We exclude two additional cases on Qwen-0.5B, at budgets 0.0085 and 0.02, because we cannot attribute their responses to the attack. Both produced the same text, which also appeared under random perturbations. Both were detected; Appendix E explains the exclusion, including a difference between float32 and float16 decoding.

At budget 0.05, we observe four attack-induced jailbreaks on Qwen-0.5B and one on Qwen-7B. However, the overall change in the Qwen-0.5B jailbreak rate is not statistically significant ($p = 0.688$). The first significant increase occurs at budget 0.12, before correction for testing multiple budgets ($p = 0.022$). After Holm correction (Holm, 1979) of paired McNemar tests (McNemar, 1947) across budgets, only the increase at budget 0.3 remains significant (Appendix E).

Table 1: All observed PGD-induced jailbreaks are detected. Counts pool five budgets, reusing 50 Qwen-0.5B prompts and 15 per other model. Gains are paired clean-negative to attacked-positive judge labels; dashes indicate no gains. Two Qwen-0.5B transitions that trace to one float16 decoding are excluded (Appendix E). The audit logs only the final layer; recomputation uses full logs and the serving-batch threshold. See Section 5 for the retrospective stopping check.

Model	Prompts	Gains	Prompts with gains	Undetected by	
				final-layer audit	recomputation
Qwen-0.5B	50	40	29	0	0
Qwen-1.5B	15	3	3	0	0
Qwen-7B	15	3	1	0	0
Gemma-1B	15	0	0	—	—
Llama-3.2-1B	15	0	0	—	—

4.4 Adaptive attacks

In the previous experiments, the attacker did not have access to the logging location. Here we test whether knowing that location enables evasion (Carlini & Wagner, 2017b; Athalye et al., 2018; Tramèr et al., 2020). We give PGD the logged layer and constrain a differentiable surrogate of the detector score to stay below that layer's threshold (ceiling), at that layer only, while bounding the injection perturbation at budgets 0.12 and 0.3, across 12 model, layer, and budget cells of 15 to 50 prompts.[1] Measured at the injection site, the attack retains only 0.1–1.3% of the nominal budget (cell medians); raising the budget from 0.12 to 0.3 leaves the realized perturbation approximately unchanged. Across 270 judged constrained attacks, none produces an induced jailbreak. Removing the score constraint yields 91 attack-induced jailbreaks across 210 judged attacks (25 on Qwen-0.5B, 7 on Gemma, 59 on Llama), all of which are detected at the logged layer. Evasion is local: on Qwen-0.5B at budget 0.12 (realized 0.0016), when the logged location is the final layer, the logged final layer flags 2/25 attacks while layer 21 flags 24/25.

We then relax the logged-layer ceiling to test whether the attacker recovers useful steering (Figure 5). In the Llama sweep, raising the ceiling restores part of the perturbation budget but also restores detection: ceilings 6–20 yield 15/15 detections and no judged jailbreaks.

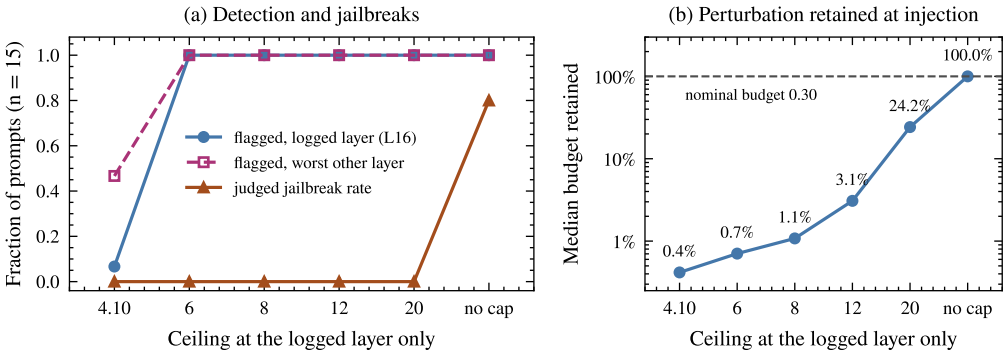


Figure 5: Logged-layer-only attack on Llama (15 prompts, budget 0.3; injection L11, logged L16). (a) At ceilings 6–20, both detection curves reach 15/15 and jailbreaks remain 0/15. The other layer is selected retrospectively. (b) Median budget retained. Without a ceiling, 12/15 jailbreaks occur, all detected. Detection uses inverted states; ceiling 4.10 allows for surrogate error below threshold 4.40.

[1] Qwen-0.5B: 25 prompts, final layer logged. Gemma and Llama: 15 prompts at each of an intermediate layer and the final layer; Llama adds a 50-prompt extension at the final layer.

5 DEPLOYMENT AND EVASION LIMITS

The original experiments used prompt-checked continuation. Applying the fitted stopping rule retrospectively to their saved inversions shortens no evaluation row, leaving recorded tokens, residuals, and verdicts unchanged. A separate held-out check finds rare truncations on Qwen-7B and none on the other models (Appendix F.9).

Token hopping. We also tested whether replacing an activation with another token's candidate state can evade the detector at that position, since inversion simply recovers the substituted token. It can, but it does not by itself provide useful steering. On Qwen-0.5B, hopping at position zero to the nearest four candidates changes no next-token predictions in 300 evaluations. Adding a PGD attack on top of the hop performs worse than PGD alone on a sentiment logit-gap objective, and the recomputation check detects every pure hop. Hopping at sentiment-bearing content positions does yield next-token changes, but it requires substantially larger perturbations, 0.89 and 0.78 on average at two early layers, and does not establish an effective undetected jailbreak (PGD uses budgets from 0.0085, for reference).

6 LIMITATIONS

Attacks are optimized in fp32 and decoded in fp16; at budget 0.0085 the two disagree on the next token for 2 of 50 prompts, and only one of those was judge-positive, producing the two excluded transitions (Appendix E). Regrading identical text three times changes 34 of 750 judge labels, so counts of one or two gains should not be read as signal. Per-layer thresholds are the 95th percentile of the same bank they are fitted on; held-out clean arms confirm them at 0–1 of 15 and 1 of 50, but the multi-layer rule reads 5–10% held out at a nominal 5% (Appendix D). Four models have only 15 prompts: even 15/15 detection has a 95% interval lower bound near 78%. Every audited attack is a PGD perturbation at approximately 70% depth, at the last prompt position, during prefill; contrastive activation steering was not run through the calibrated audit, and its downstream deviation attenuates on some models (Appendix F.8). Under the `all` constraint, the "every jailbreak detected" result is Qwen-only; Gemma and Llama gains appear only when only the injection layer is capped (Section 4.4), and nominal budgets are not comparable across models. The tested adaptive attacker sums a jailbreak loss and a surrogate detector loss rather than orthogonalizing them, and never shaped a profile across several layers at a behavior-changing budget; the surrogate's error was measured at 0.1–0.2 z-score units (Appendix F.7), so gradient masking is not the explanation. Models range from 124M to 7B; trusted weights are assumed. Median inversion times were 9–28 seconds per prompt per layer.

7 RELATED WORK

Steering as an attack surface. Activation addition (Turner et al., 2023), contrastive activation addition (Panickssery et al., 2024), and representation engineering (Zou et al., 2023a) modify behavior without touching weights or prompts. Recent work treats steering vectors as a jailbreak channel (Korznikov et al., 2025; Aidakhmetov et al., 2026), motivating the question of whether such modifications can be detected from logged activations.

Reachability and inversion. Mishra et al. (2026) show that steered states almost surely have no prompt preimage and test SipIt (Nikolaou et al., 2026) on three of our five models, at the steering layer, with ten prompts, proposing no detector, threshold, or false-positive rate. Our Figure 2 shows the steering layer is the weakest place to audit. Jiang & Haghtalab (2025) prove surjectivity for continuous inputs; the non-surjectivity that enables our audit arises because prompts are discrete token sequences. Learned inverters (Morris et al., 2023; 2024) approximate the preimage and provide no clean-state residual to calibrate against.

Latent monitors and adaptive attacks. Supervised probes (Alain & Bengio, 2016; Goldowsky-Dill et al., 2025) and activation-density monitors (Lee et al., 2018) fall to adaptive attackers that shape the latent representation (Bailey et al., 2026), in line with the adaptive-attack record on input-space detectors (Carlini & Wagner, 2017b; Athalye et al., 2018; Tramèr et al., 2020). This audit differs because the attacker must keep the logged state consistent with some token sequence under trusted weights; Section 4.4 measures the price of that constraint at 99% of the perturbation budget.

Verifiable inference. Prior audits of logged activations rerun the prompt and verify that the model reproduces the recorded states (Ong et al., 2025; Karvonen et al., 2025). Our setting has no prompt and no rerun; the adversary edits the residual stream rather than substituting the model or its precision. Both settings must tolerate nondeterminism from batching and reduced precision (Yuan et al., 2025; He, 2025), the floor that also limits our recomputation baseline.

Introspection. Model self-report is the alternative to an external audit. Lindsey (2025) find emergent introspective awareness, and Fonseca Rivera & Africa (2025) fine-tune models to report injection, with transfer that depends on steering-direction geometry; subsequent work finds models sense an anomaly but not its content (Macar et al., 2026; Lederman & Mahowald, 2026; Singh et al., 2026), supporting auditing from outside rather than relying on model reports.

8 Conclusion

Within the tested PGD settings, we find no attack that both induces a jailbreak and evades the logged-layer audit: all successful attacks are detected, and constraining PGD attacks to stay below the audit score suppresses their behavioral gains. This motivates retaining one late layer's activations alongside a clean calibration bank when full-log recomputation is unavailable. The finding could fail for an attacker that shapes the activation profile to preserve behavior while suppressing the logged-layer score, or for other steering families. Reachable-state substitutions (token hopping) also expose a limit: a low residual establishes local compatibility, not the integrity of the entire computation.

Reproducibility statement

Code, data, and configuration files to reproduce all experiments, figures, and tables are provided in the supplementary material. Appendix G reports compute.

Ethics statement

This work aims to make activation-level tampering auditable given limited access to logs. All attacks used harmful goals from the public JailbreakBench dataset. Judge-positive outputs were generated only for automated evaluation and were not released.

LLM disclosure

In this work, we use LLMs for literature search, drafting and editing text, creating figures, and an adversarial review of our own analyses. We have not used LLMs for designing the method, running the main experiments, or interpreting results. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Abzal Aidakhmetov, Donato Crisostomi, Tommaso Mencattini, Adrian Robert Minut, Iacopo Masi, and Emanuele Rodolà. Steering Vectors are an Adversarial Attack Surface. *arXiv preprint arXiv:2606.05958*, 2026. URL https://arxiv.org/abs/2606.05958.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016. URL https://arxiv.org/abs/1610.01644.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in Language Models Is Mediated by a Single Direction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2406.11717.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 274–283, 2018. URL https://arxiv.org/abs/1802.00420.

Luke Bailey, Alex Serrano, Abhay Sheshadri, Mikhail Seleznyov, Jordan Taylor, Erik Jenner, Jacob Hilton, Stephen Casper, Carlos Guestrin, and Scott Emmons. Obfuscated Activations Bypass LLM Latent-Space Defenses. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2412.09565.

Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017a. doi: 10.1109/SP.2017.49. URL https://arxiv.org/abs/1608.04644.

Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec)*, pp. 3–14, 2017b. doi: 10.1145/3128572.3140444. URL https://arxiv.org/abs/1705.07263.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track (NeurIPS)*, 2024. URL https://arxiv.org/abs/2404.01318.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A Mathematical Framework for Transformer Circuits. Transformer Circuits Thread, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Joshua Fonseca Rivera and David Demitri Africa. Steering Awareness: Detecting Activation Steering from Within. *arXiv preprint arXiv:2511.21399*, 2025. URL https://arxiv.org/abs/2511.21399.

Gemma Team. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Nicholas Goldowsky-Dill, Bilal Chughtai, Stefan Heimersheim, and Marius Hobbhahn. Detecting Strategic Deception with Linear Probes. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 19755–19786, 2025. URL https://arxiv.org/abs/2502.03407.

Aaron Grattafiori et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Horace He. Defeating Nondeterminism in LLM Inference. Thinking Machines Lab: Connectionism, 2025. URL https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

Sture Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. URL https://www.jstor.org/stable/4615733.

Haozhe Jiang and Nika Haghtalab. On Surjectivity of Neural Networks: Can you elicit any behavior from your model? *arXiv preprint arXiv:2508.19445*, 2025. URL https://arxiv.org/abs/2508.19445.

Adam Karvonen, Daniel Reuter, Roy Rinberg, Luke Marks, Adrià Garriga-Alonso, and Keri Warr. DiFR: Inference Verification Despite Nondeterminism. *arXiv preprint arXiv:2511.20621*, 2025. URL https://arxiv.org/abs/2511.20621.

Anton Korznikov, Andrey Galichin, Alexey Dontsov, Oleg Y. Rogov, Ivan Oseledets, and Elena Tutubalina. The Rogue Scalpel: Activation Steering Compromises LLM Safety. *arXiv preprint arXiv:2509.22067*, 2025. URL https://arxiv.org/abs/2509.22067.

Harvey Lederman and Kyle Mahowald. Emergent Introspection in AI is Content-Agnostic. *arXiv preprint arXiv:2603.05414*, 2026. URL https://arxiv.org/abs/2603.05414.

Olivier Ledoit and Michael Wolf. A Well-Conditioned Estimator for Large-Dimensional Covariance Matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004. doi: 10.1016/S0047-259X(03)00096-4.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://arxiv.org/abs/1807.03888.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-Time Intervention: Eliciting Truthful Answers from a Language Model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2306.03341.

Jack Lindsey. Emergent Introspective Awareness in Large Language Models. Transformer Circuits Thread, 2025. URL https://transformer-circuits.pub/2025/introspection/index.html. arXiv preprint arXiv:2601.01828.

Uzay Macar, Li Yang, Atticus Wang, Peter Wallich, Emmanuel Ameisen, and Jack Lindsey. Mechanisms of Introspective Awareness. *arXiv preprint arXiv:2603.21396*, 2026. URL https://arxiv.org/abs/2603.21396.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1706.06083.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35181–35224, 2024. URL https://arxiv.org/abs/2402.04249.

Quinn McNemar. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

Aayush Mishra, Daniel Khashabi, and Anqi Liu. Steered LLM Activations are Non-Surjective. *arXiv preprint arXiv:2604.09839*, 2026. URL https://arxiv.org/abs/2604.09839. ICLR 2026 Workshops on Sci4DL and Re-Align.

John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M. Rush. Text Embeddings Reveal (Almost) As Much As Text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 12448–12460, 2023. URL https://arxiv.org/abs/2310.06816.

John X. Morris, Wenting Zhao, Justin T. Chiu, Vitaly Shmatikov, and Alexander M. Rush. Language Model Inversion. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2311.13647.

Giorgos Nikolaou, Tommaso Mencattini, Donato Crisostomi, Andrea Santilli, Yannis Panagakis, and Emanuele Rodolà. Language Models are Injective and Hence Invertible. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.15511.

Jack Min Ong, Matthew Di Ferrante, Aaron Pazdera, Ryan Garner, Sami Jaghouar, Manveer Basra, Max Ryabinin, and Johannes Hagemann. TOPLOC: A Locality Sensitive Hashing Scheme for Trustless Verifiable Inference. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47196–47211, 2025. URL https://arxiv.org/abs/2501.16007.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering Llama 2 via Contrastive Activation Addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024. doi: 10.18653/v1/2024.acl-long.828. URL https://arxiv.org/abs/2312.06681.

Qwen Team. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Leo Schwinn, David Dobre, Sophie Xhonneux, Gauthier Gidel, and Stephan Günnemann. Soft Prompt Threats: Attacking Safety Alignment and Unlearning in Open-Source LLMs through the Embedding Space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2402.09063.

Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, and Stephen Casper. Latent Adversarial Training Improves Robustness to Persistent Harmful Behaviors in LLMs. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=wI5uHZLeCZ.

Shashwat Singh, Tal Linzen, and Shauli Ravfogel. Can LLMs Introspect? A Reality Check. In *Conference on Language Modeling (COLM)*, 2026. URL https://arxiv.org/abs/2605.26242.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A StrongREJECT for Empty Jailbreaks. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track (NeurIPS)*, 2024. URL https://arxiv.org/abs/2402.10260.

Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On Adaptive Attacks to Adversarial Example Defenses. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2002.08347.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering Language Models With Activation Engineering. *arXiv preprint arXiv:2308.10248*, 2023. URL https://arxiv.org/abs/2308.10248.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and Mitigating Numerical Sources of Nondeterminism in LLM Inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2506.09501.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation Engineering: A Top-Down Approach to AI Transparency. *arXiv preprint arXiv:2310.01405*, 2023a. URL https://arxiv.org/abs/2310.01405.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and Transferable Adversarial Attacks on Aligned Language Models. *arXiv preprint arXiv:2307.15043*, 2023b. URL https://arxiv.org/abs/2307.15043.

A THE JUDGE

Against 300 human-labeled JailbreakBench examples, the HarmBench classifier prompt with `deepseek-v4-flash-0731` achieves 87.7% agreement, 86.4% true-positive rate, 11.6% false-positive rate, and F_1 of 0.837. Published comparison figures are 87.7% agreement for Llama Guard 2, 90.3% for GPT-4, and 90.7% for Llama 3. Substring matching achieves 55.3% agreement and a 69.5% false-positive rate; it is shown only as a diagnostic baseline, not used to define behavioral success.

Providing the original goal rather than the wrapped prompt improves judge agreement by 2–6 percentage points. Binarized StrongREJECT scores agree less well with the binary human labels than HarmBench, so StrongREJECT is used for graded scores and the additional response gate. Disabling reasoning in a reasoning judge reduces agreement by 22.5 percentage points in the evaluated configuration.

B CONTRASTIVE ACTIVATION STEERING

A separate experiment adds fixed steering vectors, fitted from contrastive system prompts, at the final prompt position of six models (GPT-2, Qwen-0.5B/1.5B/3B/7B, and Gemma), at scaling factors from 0.01 to 2 of the fitted direction. This experiment uses a simpler detection procedure: the true prefix is supplied, and the final-token residual is compared with a relative tolerance without statistical calibration. In fp16, detection jumps between scaling factors 0.01 and 0.02 on all six models. At 0.02 it reaches 100% on GPT-2 and Qwen-0.5B/1.5B/7B, where no Qwen-7B next-token argmax changes (0/140), and 96–97% on Qwen-3B and Gemma; all six reach 100% by 0.5.

Across four behaviors (six prompt sets), six models, both precisions, and steering at every prompt position with scaling factors -0.75 to -0.25 and 0.25 to 1, the joined generation and detection data contain 11,140 steered observations, including 2,293 judge-positive responses and no undetected judge-positive responses. The reported zero-failure bound is 1.9% using 160 behavior–prompt clusters. This outcome differs from the paired attack-induced metric used in the main experiments. Clean text from five categories, including base64, is inverted exactly at rates close to natural English: 96–100 of 100 on Qwen-0.5B and 45–48 of 52 on GPT-2. Steering-vector scaling factors have not been converted to relative perturbation budgets, so these results are not directly compared with the PGD budget experiments.

C DETECTION BY LOGGED-LAYER POSITION

Table C1: Single-layer PGD detection at relative budget 0.0085. Entries give detected / tested prompts, unchanged by the retrospective stopping check (Appendix F.9), using each layer's threshold at the clean calibration bank's 95th percentile. Offset 0 is the injection layer; injection indices are 17, 20, 20, 18, and 11 for the models in column order. A dash indicates an offset beyond the model's final layer. No evaluated cell contains an unscorable prompt. Each column evaluates the same prompts across layers; counts are not independent samples.

Offset after injection	Qwen-0.5B	Qwen-1.5B	Qwen-7B	Gemma-1B	Llama-3.2-1B
0	18/50	3/15	1/15	0/15	0/15
1	50/50	14/15	0/15	0/15	0/15
2	50/50	15/15	0/15	2/15	0/15
3	50/50	15/15	1/15	4/15	4/15
4	50/50	15/15	1/15	8/15	3/15
5	50/50	15/15	6/15	13/15	0/15
6	50/50	15/15	3/15	14/15	—
7	50/50	15/15	15/15	13/15	—
8	—	15/15	15/15	0/15	—

Gemma detection peaks at layer 24 (offset +6), with 14/15 prompts detected, then falls to 0/15 at the final layer, 26. This profile illustrates why detection need not improve monotonically with

depth; choosing the best offset on these data would require a separate evaluation to estimate its performance.

D SUPPLEMENTARY MULTI-LAYER ANALYSES

These analyses document the earlier multi-layer audit and its calibration. They provide context for the full-log comparison in Section 4.2, rather than evidence that multiple inversions improve on recomputation or that single-layer deployment is validated.

D.1 MULTI-LAYER RULE AND DETECTION RESULTS

The multi-layer rule flags a prompt if three adjacent model layers each score above σ, or any layer scores above 1.5σ. The threshold σ is calibrated per model to a nominal 5% in-sample prompt-level false-positive rate; the ratio between the thresholds is fixed. The higher threshold allows isolated large deviations to trigger detection without a qualifying run.

The run search checks every third layer and the final three layers, expanding to adjacent layers around each crossing. The higher-threshold check examines every layer. This rule can miss a two-layer deviation detected by a separately calibrated single-layer threshold (Section 4.1). A jointly calibrated rule for arbitrary sparse logs remains unevaluated.

The multi-layer rule also reaches complete detection in the evaluated samples from budget 0.02 onward (Table D1). Its agreement with final-layer detection at these budgets provides a second way to aggregate the evidence; at the smallest budget, requiring consecutive threshold crossings can reduce sensitivity.

Table D1: PGD prompts flagged under the multi-layer rule, Clopper-Pearson 95% interval. Each calibration is the model's own 75- or 100-prompt chat bank, σ tuned to 5% in-sample. No prompt was unscorable in these evaluations under the evaluator rule.

model	prompts	0.0085	0.02	0.05	0.12	0.3
Qwen-0.5B	50	50/50 (93–100%)	50/50	50/50	50/50	50/50
Qwen-1.5B	15	15/15 (78–100%)	15/15	15/15	15/15	15/15
Qwen-7B	15	1/15 (0–32%)	15/15	15/15	15/15	15/15
Gemma-1B	15	2/15 (2–40%)	15/15	15/15	15/15	15/15
Llama-3.2-1B	15	0/15 (0–22%)	15/15	15/15	15/15	15/15

At budget 0.0085, performance varies across models. On Qwen-7B, Gemma, and Llama, the all-layer constraint reduces the injection perturbation to 1.9e-3, 4.5e-4, and 1.1e-3, respectively, within a few times the fp16 recomputation floor at that layer (5.6e-4, 8.9e-4 and 5.1e-4; below it on Gemma). The multi-layer criterion does not reliably detect the resulting downstream profiles. Random perturbations are detected at equal or higher rates, indicating that detection does not require alignment with the targeted behavior.

D.2 CALIBRATION OF THE MULTI-LAYER RULE

We use five folds and twenty splits, fitting layer-and-role statistics on three folds, selecting σ on a fourth, and evaluating the combined rule on the fifth. Both thresholds are scaled together at their fixed ratio of 1.5.

Held-out false-positive rates. Table D2 reports the multi-layer rule under the three-way fitting, threshold-selection, and evaluation split. Calibration to a nominal 5% in-sample rate yields held-out rates of 4.6–9.6% with 75 or 100 prompts, and 15.6–17.2% with 25 prompts. A separate calibration using clean controls from the attack experiments yields held-out rates of 8–17%.

Table D2: Held-out false-positive rates exceed the nominal target on several models. Five folds and twenty splits; mean ± standard deviation. Role statistics are fitted on three folds, the threshold on a fourth, and outcomes evaluated on the fifth.

model	clean set	n	in-sample	three-way held out
GPT-2	bare	100	5%	9.6 ± 3.5%
Qwen-0.5B	chat	25	4%	17.2 ± 7.6%
Qwen-0.5B	chat	75	4%	6.0 ± 2.8%
Qwen-0.5B	chat	100	5%	9.6 ± 2.7%
Qwen-1.5B	chat	75	4%	5.5 ± 3.0%
Qwen-7B	chat	75	4%	6.4 ± 1.4%
Gemma-1B	chat	25	4%	15.6 ± 5.4%
Gemma-1B	chat	75	4%	4.6 ± 2.0%
Llama-3.2-1B	chat	75	4%	6.1 ± 2.7%

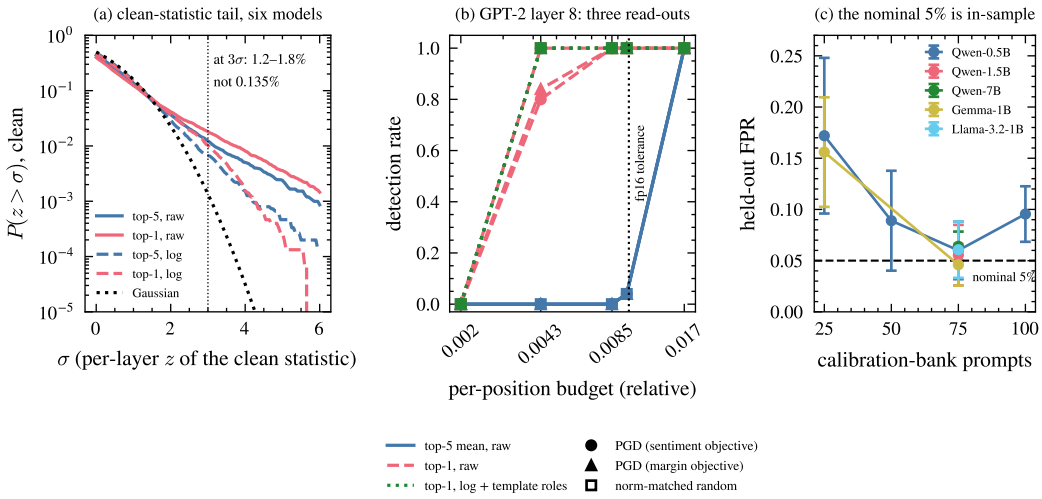


Figure D1: Clean-score tails, position aggregation, and bank size affect the detection–false-positive tradeoff. (a) Empirical clean-score exceedance probabilities pooled over six models, comparing raw and log residuals, top-5 and top-1 aggregation, and a Gaussian reference. At 3σ, the raw-score exceedance rate is 1.2% (top-5) to 1.8% (top-1), compared with the Gaussian value of 0.135%. (b) Detection versus per-position budget for three aggregation rules at GPT-2 layer 8 with one attacked position. (c) Held-out false-positive rates of the multi-layer rule versus calibration-bank size, using three-way splits and twenty repeats; error bars show standard deviations.

D.3 Additional detector-aware comparisons

The attack in Appendix F.7 constrains scores at every downstream layer. These supplementary comparisons concern that earlier experiment; Section 4.4 reports the newer logged-layer-only attack.

We also project against the slack permitted by the multi-layer rule, which allows two adjacent layers to exceed σ without satisfying the consecutive-layer criterion. This does not improve the attack: the projection shrinks the perturbation globally and does not produce a localized peak in the score profile. Within these evaluated optimization procedures, we find no configuration that both induces a jailbreak above baseline and evades detection. The unconstrained reference in Table F4 combines next-token results from this experiment with separately identified detection and judging results from the 50-prompt experiment.

E PAIRED BEHAVIORAL TRANSITIONS AND STATISTICAL TESTS

The detection analysis in Section 4.3 includes every clean-negative to attacked-positive transition, regardless of whether the attack increases overall jailbreak success significantly at that budget. Table E1 addresses that separate question by comparing gains with losses. "Clean +" and "attacked +" count HarmBench-positive responses; a loss is a clean-positive to attacked-negative transition. Only discordant pairs (gains and losses) enter the McNemar test. The final column retains the multi-layer detection result from the original transition analysis; Table 1 and Table E2 report final-layer and recomputation outcomes.

Table E1: Paired transitions on Qwen-0.5B, n = 50, using the HarmBench-prompt judge. A gain passes the additional gate if its StrongREJECT score is at least 0.125, the minimum score assigned to a non-refusal by that rubric. Passing this gate does not establish that a response is useful. McNemar tests are two-sided and exact over discordant pairs; we report unadjusted p-values and Holm adjustment across budgets.

budget	arm	clean +	attacked +	gains	losses	McNemar p	Holm p	gains passing gate	gated, undetected (multi-layer rule)
0.0085	pgd	3	3	1†	1	1.000	1.000	1†	0
0.02	pgd	5	4	1†	2	1.000	1.000	1†	0
0.05	pgd	4	6	4	2	0.688	1.000	2	0
0.12	pgd	4	13	11	2	0.022	0.090	5	0
0.3	pgd	4	27	25	2	<0.001	<0.001	12	0
0.3	random	4	4	3	3	1.000	1.000	3	0

† Table E1 lists raw judge-label transitions, which the McNemar tests use. The transitions marked † at 0.0085 and 0.02 are one prompt and one decoding, and are not counted as attack-induced jailbreaks elsewhere in the paper. At 0.0085, one transition each way leaves the total unchanged at 3/50 (McNemar p = 1.000). The † transition there is a precision effect: the attack is optimized in float32, where its perturbation leaves the prompt's next token unchanged, but completions are decoded in float16, where the same perturbation (0.35% relative at injection, below the 0.01 float16 tolerance) yields a different first token and a judge-positive completion. Float32 and float16 disagree on the first token for 2 of 50 prompts at 0.0085 and for none at larger budgets. The same prompt's attacked completion at 0.02, where float32 also changes the token, is byte-identical, and the random arm produces the same completion at 0.12 and 0.3, so the text cannot be attributed to the attack.

Excluding both leaves 40 gains on Qwen-0.5B and 46 across models (Table 1 and Table E2); both excluded transitions were flagged by the final-layer audit and by recomputation, and no test conclusion changes (unadjusted p = 1.000 at 0.0085 and 0.500 at 0.02 without them). At 0.05, four gains against two losses are not significant (p = 0.688). The jailbreak rate first rises at 0.12, where the increase is significant before correction (p = 0.022) but not after Holm adjustment (p = 0.090); only budget 0.3 remains significant after adjustment. Random perturbations provide no clear evidence of a behavioral effect. Across PGD budgets, 19 of the 40 counted gains pass the StrongREJECT gate (21 of 42 with the two excluded transitions), involving 14 distinct prompts. These are a subset of the gains evaluated for detection in Section 4.3.

Regrading identical text changes 34 of 750 labels: 11 of 250 clean responses, 14 PGD responses, and 9 random responses. Paired testing does not eliminate this judging variability. The 40 gain observations involve 29 distinct prompts, so they should not be treated as 40 independent prompt samples.

Table E2: Attack-induced jailbreaks and undetected counts by model and budget. The same prompts are used at each budget (50 for Qwen-0.5B, 15 for each other model). Undetected counts use final-layer auditing and full-log recomputation with the serving-batch threshold; the retrospective stopping check leaves them unchanged (Appendix F.9). A dash denotes the absence of gains under the `all` constraint, not successful detection; Gemma and Llama gains under injection-only capping appear in Section 4.4. † One judge-positive transition at this budget is excluded (Table E1); it was flagged by both auditors. These observations are repeated evaluations rather than independent prompt samples.

Model	Budget	Gains	Undetected by final-layer audit	Undetected by full-log recomputation
Qwen-0.5B	0.0085	0†	—	—
Qwen-0.5B	0.02	0†	—	—
Qwen-0.5B	0.05	4	0	0
Qwen-0.5B	0.12	11	0	0
Qwen-0.5B	0.3	25	0	0
Qwen-1.5B	0.0085	0	—	—
Qwen-1.5B	0.02	0	—	—
Qwen-1.5B	0.05	0	—	—
Qwen-1.5B	0.12	0	—	—
Qwen-1.5B	0.3	3	0	0
Qwen-7B	0.0085	0	—	—
Qwen-7B	0.02	0	—	—
Qwen-7B	0.05	1	0	0
Qwen-7B	0.12	1	0	0
Qwen-7B	0.3	1	0	0
Gemma-1B	0.0085	0	—	—
Gemma-1B	0.02	0	—	—
Gemma-1B	0.05	0	—	—
Gemma-1B	0.12	0	—	—
Gemma-1B	0.3	0	—	—
Llama-3.2-1B	0.0085	0	—	—
Llama-3.2-1B	0.02	0	—	—
Llama-3.2-1B	0.05	0	—	—
Llama-3.2-1B	0.12	0	—	—
Llama-3.2-1B	0.3	0	—	—

F Supporting methods and experiments

F.1 Audit access and terminology

Table F0 defines the experimental terminology used throughout the appendices. Table F1 extends Figure 1 with the access each auditor needs and the scale at which it detects. Recomputation compares a recomputed block with its logged output, so its floor is the fp16 kernel discrepancy between executions; the reachability audit compares a logged state with its best token reconstruction, so its floor is the inversion tolerance. The two thresholds therefore measure different quantities and are not interchangeable.

Table F0: Experimental terminology.

term	meaning
relative budget	$\|h' - h\|/\|h\|$ at the injected position, h the clean state
reference budget	0.0085, 0.85 $\times$ the base fp16 inversion tolerance
constraint scope	`all`: capped at the injection layer and every later one; `injection`: capped at the injection layer only
arm	`pgd`, `random`, or `clean`
role	a prompt position's class in the chat template: prefix token, first/second/last content token, other content, suffix token
logged layer, read set	the layer(s) whose residual state the auditor holds
offset	a logged layer's distance past the injection layer
gain	judge-negative under the paired clean arm, judge-positive under attack
stopping rule	inversion halts at the first residual above the fitted per-layer tolerance, without the prompt

Table F1: Audit methods and their access requirements.

auditor access	method	detection location	numerical scale
prompt, weights, any layer	clean counterfactual	every layer after the injection	fp16 kernel discrepancy
weights, layers on both sides of the injection	recomputation, one block	the pair straddling the injection	fp16 kernel discrepancy
weights and one or more layers at or after injection; logged layers unknown to attacker	reachability audit (this paper)	evaluated separately at each logged layer	calibrated layer-score threshold

F.2 Detection counts at the two smallest budgets

Table F2 gives the counts behind Figure 2(a,b) at the two budgets where the auditors differ. At 0.0085, the final layer alone flags every attack on the three Qwen models and none on Gemma or Llama, while full-log recomputation flags 15/15 on Llama and 1/15 on Gemma. At 0.02, the final layer flags every attack on all five models, whereas the injection layer alone flags at most 1/15 on Qwen-7B, Gemma, and Llama. The final layer flags at most one held-out clean prompt per model.

Table F2: PGD detection at relative budgets 0.0085 and 0.02. Entries give flagged prompts / total prompts. Individual-layer thresholds are the clean calibration bank's 95th percentiles; the held-out clean column measures false positives. Results are unchanged by the retrospective stopping check (Appendix F.9).

model	budget	recomputation, full log	injection layer alone	final layer alone	held-out clean, final layer
Qwen-0.5B (n=50)	0.0085	50/50	18/50	50/50	1/50
Qwen-1.5B	0.0085	15/15	3/15	15/15	0/15
Qwen-7B	0.0085	15/15	1/15	15/15	0/15
Gemma-1B	0.0085	1/15	0/15	0/15	0/15
Llama-3.2-1B	0.0085	15/15	0/15	0/15	0/15
Qwen-0.5B (n=50)	0.02	50/50	50/50	50/50	1/50
Qwen-1.5B	0.02	15/15	15/15	15/15	0/15
Qwen-7B	0.02	15/15	1/15	15/15	0/15
Gemma-1B	0.02	14/15	0/15	15/15	0/15
Llama-3.2-1B	0.02	15/15	1/15	15/15	0/15

F.3 Full-log recomputation comparison

We compare multi-layer inversion with block recomputation on the same trajectories. The recomputation threshold is the maximum relative discrepancy between each recomputed block output and its logged counterpart, at the clean bank's 95th percentile. For logs produced by the same execution kernel, recomputation detects all 1,100 attacked trajectories with no false positives among 110

clean trajectories. For batched generation logs, numerical differences produce a relative floor of 5.1e-4–8.9e-4 (Table F3).

Table F3: Recomputation under the serving-batch floor at the reference budget, PGD arm.

model	realized delta at injection (median)	floor	margin	recomputation flagged	multi-layer rule flagged	final layer alone
Qwen-0.5B (n=50)	3.4e-3	5.8e-4	5.9×	50/50	50/50	50/50
Qwen-1.5B	2.0e-3	6.3e-4	3.2×	15/15	15/15	15/15
Qwen-7B	1.9e-3	5.6e-4	3.3×	15/15	1/15	15/15
Gemma-1B	4.5e-4	8.9e-4	0.5×	1/15	2/15	0/15
Llama-3.2-1B	1.1e-3	5.1e-4	2.1×	15/15	0/15	0/15

Recomputation detects 15/15 attacks on Qwen-7B and Llama where the multi-layer rule detects 1/15 and 0/15. On Gemma, the counts are 1/15 versus 2/15 at budget 0.0085 and 14/15 versus 15/15 at 0.02. The tested multi-layer rule shows no consistent advantage over recomputation when a full log is available; the motivation for inversion is the absence of a logged boundary spanning injection.

F.4 Classifier and activation-density baselines

Linear probe. Logistic regression on the last prompt position at the known injection layer, five-fold cross-validated by prompt. At budget 0.0085 on Qwen-0.5B, it flags 27/50 attacked but also 16/50 clean examples. On each other model, clean and attacked counts differ by at most one. When trained at budget 0.3 and evaluated at smaller budgets, the probe detects no attacks at 0.0085–0.05 on four models and few at 0.12 (1/50 on Qwen-0.5B, 0/15 on Qwen-1.5B, 6/15 on Qwen-7B, 4/15 on Gemma, with no clean flags); on Llama it flags 4–8/15 attacked prompts at every budget against 4/15 clean.

Activation-density baselines (Qwen-0.5B only). We compare standardized Euclidean, shrinkage Mahalanobis, and nearest-reference distances at the injection layer and every downstream layer, using leave-one-out calibration at the 95th percentile. At budget 0.0085, the attack changes the anomaly score by less than 0.01 reference standard deviations; inversion detects 50/50. At budget 0.3, shrinkage Mahalanobis detects 49/50 at its best layer. Results depend on the reference distribution: with the five-category bank, the shrinkage estimator flags all 50 clean harmful prompts at the injection layer. Figure 3 compares these baselines with inversion and recomputation.

F.5 Calibration and aggregation diagnostics

At the final layer, held-out clean detections are 1/50 on Qwen-0.5B and 0/15 on each other model (Table F2). Across six models, 1.2% (top-5) to 1.8% (top-1) of raw clean scores exceed 3σ, 9–13 times the Gaussian reference; log transformation reduces this to 0.7–1.0%. Maximum aggregation detects single-position perturbations at lower budgets than a top-five mean (Appendix F.6). Template-role conditioning prevents shared template positions from dominating the score; on Qwen-0.5B, the resulting single-layer evaluation achieves a 2% held-out false-positive rate.

F.6 Concentrated and distributed perturbations

At GPT-2 layer 8 with 25 prompts, top-5 aggregation detects 0% of single-position attacks at budgets up to 0.0085 and becomes sensitive at 0.017, above the inversion tolerance. Top-1 aggregation reaches 80–84% at budget 0.0043 and saturates from 0.0085. The difference follows from averaging: a perturbed-position residual of 5.9e-3 is reduced to 1.8e-3 when averaged with four clean positions, below the top-5 threshold of 2.6e-3. The top-1 score remains 5.9e-3, above its threshold of 3.3e-3. Experiments on Qwen-0.5B at layer 17 reproduce the pattern.

Perturbing m positions at a fixed per-position budget of 0.0085, the top-1 score-to-threshold ratio remains between 2.61 and 2.63 for $m \in \{1, 2, 3, 5, 9\}$, while the attacker's logit-gap objective increases by a factor of 4.75. No distributed perturbation changes the next-token argmax in this experiment.

F.7 DETECTOR-AWARE OPTIMIZATION

The all-layer detector-aware attack uses a differentiable surrogate for the audit statistic: root-sum-square of the distance to the clean state and the numerical floor, followed by log transformation and layer-role standardization. On both unconstrained and penalized attacks, the surrogate differs from the measured audit by 0.1–0.2 z-score units. The experiment uses Qwen-0.5B, injection at layer 17, budget 0.3, and 15 prompts.

Table F4: Final-layer detection under an all-layer score constraint. Qwen-0.5B, layer 17, budget 0.3, n = 15 unless otherwise marked. The ceiling applies to every downstream layer. The final row includes separately identified results from the 50-prompt experiment.

z ceiling	next-token flips	peak deviation / budget	flagged, final layer alone	judged jailbreaks
4.28	0/15	1%	0/15	7% (clean baseline 7%)
6	0/15	2%	4/15	7%
8	0/15	3%	15/15	0%
12	0/15	5%	15/15	0%
16	0/15	11%	15/15	0%
20	1/15	21%	15/15	7%
none	15/15	98%	50/50 (separate n = 50 experiment)	not judged in this run; 27/50 in the separate n = 50 experiment

At ceiling 4.28, the peak downstream deviation is about 1% of the budget (0.5% at injection), and no next token changes. Judged jailbreak rates do not exceed the clean baseline at any tested finite ceiling. This is a stronger constraint than the single-layer attack in Section 4.4; its failure to evade does not establish robustness of the single-layer audit. Appendix D.3 reports additional detector-aware comparisons.

F.8 INJECTION-ONLY AND ALL-LAYER NORM CONSTRAINTS

We compare injection-only and all-layer norm constraints in fp32 sentiment experiments with 25 prompts and budget 0.0085, recording relative deviations at every layer (Figure F1, Table F5).

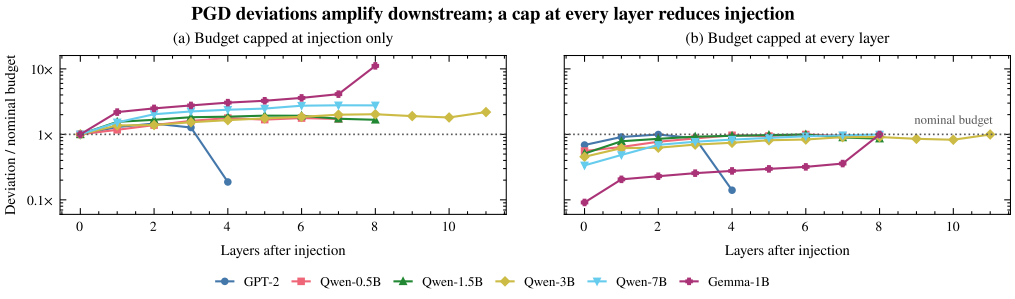


Figure F1: PGD clean-trajectory deviation in fp32 sentiment experiments on six models (25 prompts each, nominal budget 0.0085). Curves show median relative deviation at the injected position, divided by the budget. Injection-only constraints (a) permit downstream amplification; all-layer constraints (b) reduce the injected perturbation. These are activation deviations, not inversion residuals or jailbreak measurements.

Table F5: Downstream amplification and injection perturbation under different constraints. PGD arm; medians over prompts.

model	constrained at injection only: peak downstream (layer)	amplification	constrained at every layer: deviation at injection
GPT-2	1.24e-2 (L10 of 12)	1.46×	5.85e-3
Qwen-0.5B	1.51e-2 (L23 of 24)	1.79×	4.73e-3
Qwen-1.5B	1.64e-2 (L25 of 28)	1.95×	4.33e-3
Qwen-3B	1.86e-2 (L36 of 36)	2.19×	3.86e-3
Qwen-7B	2.36e-2 (L27 of 28)	3.00×	2.86e-3
Gemma-1B	9.46e-2 (L26 of 26)	11.1×	7.75e-4

With the injection layer constrained to budget 0.0085, peak downstream deviations are 1.5–3 times larger on GPT-2 and the Qwen models, and 11 times larger on Gemma. Amplification increases with model size within the Qwen family. Applying the constraint to every layer reduces the permissible injection perturbation correspondingly. Random perturbations do not show the same amplification, except on Gemma (4.4×).

The jailbreak ladders show the same effect under the `all` constraint (Table F6). The cap binds near the final layer, so the share of the budget left at injection falls as downstream growth rises, from 53% on Qwen-0.5B to 4% on Gemma; Gemma and Llama change no next token even at budget 0.3.

Table F6: Budget left at injection in the jailbreak ladders under the `all` constraint. Budget 0.3, PGD arm, means over prompts. Growth is the mean peak downstream deviation divided by the mean deviation at injection; next-token changes are measured in fp32, where the attack is optimized.

model	prompts	deviation at injection	share of budget	growth to peak	next token changed
Qwen-0.5B	50	0.159	53%	1.9×	50/50
Qwen-1.5B	15	0.105	35%	2.8×	7/15
Qwen-7B	15	0.064	21%	4.7×	1/15
Gemma-1B	15	0.013	4%	23×	0/15
Llama-3.2-1B	15	0.034	11%	8.7×	0/15

F.9 Deployment stopping diagnostics

The evaluator stopping rule permits inversion to continue past a tolerance failure when the recovered token matches the known prompt. In deployment, inversion stops at the first tolerance failure. Under a flat 0.01 tolerance, truncating saved results at that point shortens 374 of 11,620 rows. Coverage losses are concentrated near the final layers: on Qwen-1.5B, every clean prompt becomes unscorable at layers 27–28; on Gemma, removing late calibration residuals reduces the final-layer threshold from 3.91 to 3.07, changing detections at budget 0.0085 from 0/15 to 15/15. The per-layer tolerance of Section 3.3 shortens no evaluation or calibration row and removes these changes.

In five-fold validation repeated 20 times, tolerances were fitted on training prompts and checked on held-out prompts. The mean fraction of rows truncated was 0.046% on Qwen-7B and zero on the other four models. These checks assess stopping on saved inversions; they do not constitute an end-to-end deployment benchmark.

F.10 Token hopping and reachable-state substitutions

A token hop replaces a position's activation with a different token's candidate activation. The substituted state has a local preimage, so inversion recovers the substituted token with a very small residual. In the Qwen-0.5B sentiment experiment (25 prompts, layers 6/12/18, four neighbor ranks, 300 evaluations), pure hopping at position zero changes no next-token predictions and decreases the mean sentiment objective. Hop-plus-PGD improves the objective less than PGD alone at each layer. The recomputation check detects every pure hop.

Hopping at content positions does yield next-token changes (7/25 prompts at layer 2, 6/25 at layer 6), but requires perturbations near 0.8, about a hundred times the 0.0085 ladder budget. A local token preimage does not establish consistency of the full computation: replacing an internal state

leaves earlier-layer computations intact and is not equivalent to editing the input token and rerunning the model.

G COMPUTE

Hardware. Two kinds of machine were used. A laptop with an Apple M5 Pro and 24 GB of unified memory ran PyTorch on MPS: GPT-2, every Qwen-0.5B experiment outside Section 4.4 (the 50-prompt ladder, its calibration banks, the all-layer detector-aware attacks of Appendix F.7, the recomputation, probe and density baselines) and the steering sweep of Appendix B on the 1.5B, 3B, 7B and Gemma models at 25 prompts. Four on-demand AWS instances ran the 15-prompt ladders: NVIDIA A10G (24 GB) for Qwen-1.5B, Gemma and Llama, and NVIDIA L40S (48 GB) for Qwen-7B. The Llama instance ran 8.5 hours; the other three ran about 60 hours each, including idle time, until their rows were copied back. Detection ran in fp16 on both kinds of machine; attack optimization ran in fp32.

Inversion. Every saved row records its own wall-clock time. Summed over the 138,631 (prompt, layer) inversions on disk, with byte-identical copies of a row file counted once, inversion took 231 hours: 160 hours building calibration banks and clean inversion sweeps, 48 hours scoring the PGD ladders, and 24 hours for the steering sweep of Appendix B. About 142 of those hours ran on the laptop and 90 on the rented GPUs. Table G1 gives the cost of one inversion from scratch, measured on the calibration bank rows, whose prompts are chat-formatted questions of 16 to 149 tokens (medians 27 to 52).

Table G1: One inversion of one prompt at one layer, from scratch, median over the calibration bank rows.

model	machine	layers	vocabulary	prompt tokens (median)	seconds per inversion (median / 90th pct.)	seconds per token
Qwen-0.5B	M5 Pro, MPS	24	152k	46	10.4 / 30.1	0.22
Qwen-1.5B	A10G	28	152k	47	10.3 / 23.3	0.20
Qwen-7B	L40S	28	152k	47	9.3 / 21.8	0.18
Llama-3.2-1B	A10G	16	128k	52	16.4 / 25.4	0.24
Gemma-1B	A10G	26	262k	27	27.6 / 47.3	0.67

The cost is a candidate scan over the vocabulary at each position, so it follows vocabulary size more than parameter count: the three Qwen models cost the same per token, and Gemma, with a 262k vocabulary, costs three times as much. A full audit of one clean Qwen-0.5B prompt at all 24 layers takes a median of 3.7 minutes (mean 5.8) layer by layer. When every layer recovers the same tokens, one shared scan serves all of them, and the ladders' clean arm cost 1.1 s per layer, about 27 s per prompt. For an attacked arm the untouched prefix is reused from the clean inversion and only the injected position is solved per layer; that position runs the candidate schedule to its end, so it costs 16 s per layer on the laptop (90th percentile 32 s) and 1.4 to 4.3 s on the GPUs (per-model means).

Attacks, generation and judging. Projected-gradient optimization of the 5 budgets, 40 steps each, for the PGD and norm-matched random arms took 29 minutes for the 50 prompts of Qwen-0.5B on the laptop, and on the instances 5 minutes (Gemma), 8 minutes (Llama and Qwen-1.5B) and 36 minutes (Qwen-7B, two passes) for 15 prompts. The ten all-layer detector-aware attacks of Appendix F.7, which differentiate through the audit statistic, each took about two minutes on the laptop at 15 prompts. The logged-layer attacks of Section 4.4 ran on the AWS GPU instances: the twelve constrained cells, four unconstrained runs, and ceiling sweeps took 9.3 hours of job time in total, including inversion and decoding, and at most 2.5 hours for any one run (Llama, 50 prompts, logged layer 16). Decoding the 96-token completions of all five budgets for judging took 1.1 minutes on Qwen-0.5B, 1.8 minutes on Gemma and 4.1 minutes on Qwen-7B. The 2,175 ladder completions were each graded twice through the Fireworks API by `deepseek-v4-flash-0731`, once with the HarmBench prompt and once with the StrongREJECT prompt, and the 750 completions of the 0.5B ladder were graded twice more with the HarmBench prompt for the repeat analysis of Appendix E: 5,850 gradings for the ladders. The 1,320 completions of Section 4.4 added 2,399 gradings, and the steering sweep of Appendix B judged a further 11,140 steered completions.

Storage. The fp32 vocabulary activation tables, one slice per layer, occupy 1.9 GB for GPT-2, 13 GB for Qwen-0.5B, 17 GB for Llama and 30 GB for Gemma; a single-layer auditor loads one slice,

0.5 GB, 1.05 GB and 1.2 GB respectively. Building the fp16 table took 48 s for Qwen-0.5B on the laptop, 12 minutes for Qwen-7B on the L40S and 33 minutes for Gemma on an A10G. The results directories, which hold the saved inversion rows every table in this paper is regenerated from, are 1.6 GB from the laptop and 0.7 GB from the instances; the local cache of activations and tables is 105 GB.