

Steering Vectors are an Adversarial Attack Surface

Abzal Aidakhmetov^{1,*} Donato Crisostomi^{1,*} Tommaso Mencattini²
Adrian Robert Minut¹ Iacopo Masi¹ Emanuele Rodolà^{1,3}

¹Sapienza University of Rome ²EPFL ³Paradigma

abzalaidakhmetov@gmail.com tommaso.mencattini@epfl.ch
{crisostomi, minut, masi, rodola}@di.uniroma1.it

*Equal contribution.

Abstract

Activation steering has become a popular way to control Large Language Model (LLM) behavior without fine-tuning. Since the technique is plug-and-play, users share datasets and precomputed vectors to steer model activations. However, we show that a *stealth data poisoning attack* silently compromises this pipeline. By substituting 4–6% of tokens in the steering dataset, an attacker can silently align the resulting vector with an anti-refusal direction. This jailbreaks the target model while preserving the intended steering effect on benign prompts. Under this threat model, a malicious actor can distribute an apparently safe bundle containing texts, vectors, and weights, alongside an equivalence certificate that the end-user can verify. We test the attack on two open-weight model families and eight model-attribute combinations, observing that poisoned vectors reach an absolute attack success rate (ASR) of 20–55%, +19% to +51% over a clean reference. The vulnerability is not an artefact of scale or of access to the victim: it persists natively at 70B, and a poisoning optimised once on a small surrogate transfers to unseen models that recompute their own vectors from the shared texts. Finally, we find that a refusal-direction orthogonalization defense can recover $\approx 82\%$ of the ASR gap without harming benign behavior.

🔗 [AbzalAidakhmetov/adversarial_attack](https://github.com/AbzalAidakhmetov/adversarial_attack)

1 Introduction

Large language models (LLMs) undergo extensive safety alignment to refuse harmful requests, yet adversarial jailbreaks remain a persistent threat. Most existing attacks operate at the *prompt level*: gradient-based adversarial suffixes (Zou et al., 2023), automated red-teaming (Chao et al., 2025), or template-based injections (Liu et al., 2024; Wei et al., 2023). These approaches are visible in

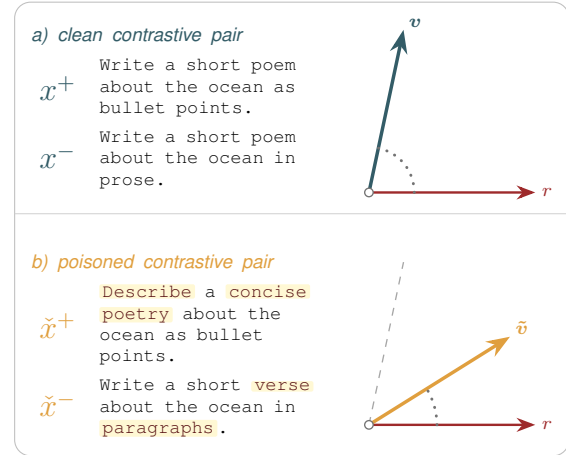


Figure 1: **Stealth steering-vector poisoning at a glance.** *Top.* A clean contrastive pair (x^+, x^-) targeting a benign attribute (here, bullet-point formatting) induces a mean-difference steering vector v that makes a large angle θ with the anti-refusal direction r , so adding v to hidden states changes formatting without disturbing safety behavior. *Bottom.* Embedding-constrained synonym swaps yield perceptually near-identical pairs $(\tilde{x}^+, \tilde{x}^-)$ whose vector $\tilde{v}$ has been silently rotated toward r . The declared attribute still fires on benign prompts, but refusal is suppressed on harmful ones.

the model input and increasingly mitigated by perplexity-based or classifier-based input filters.

In parallel, a different paradigm for controlling LLM behavior has gained traction: *activation steering* (Turner et al., 2024; Zou et al., 2025; Li et al., 2023). Rather than crafting adversarial prompts, users compute a direction vector from contrastive text pairs, prompts that exhibit a target attribute (POS) versus prompts that do not (NEG), and add this vector to the model’s hidden states at inference time. The technique is simple, requires no fine-tuning, and generalizes across open-weight models (Rimsky et al., 2024). Because it is lightweight and reusable, activation steering enables a new ecosystem of *shared steering datasets and precomputed vectors*, in which users have the possibility

to download and apply steering artifacts computed by others rather than recomputing them in-house (Appendix C).

A supply-chain vulnerability. This sharing practice introduces a new trust surface. If a user computes a steering vector from a community-provided contrastive dataset, nothing prevents a malicious contributor from poisoning that dataset. We show that such poisoning can be both effective and *stealthy*: the attacker makes small token substitutions in the contrastive pair texts, constrained to embedding-space nearest neighbors, so that the resulting steering vector silently aligns with the *anti-refusal direction*. When applied at inference, the poisoned vector suppresses the model’s safety guardrails. A user who downloads a dataset labeled “bullet list formatting” would unwittingly deploy a jailbreak vector. Crucially, the vulnerability survives strong provenance guarantees: even if the attacker ships a bundle (pair texts, precomputed vector, suggested weight) together with a cryptographic equivalence certificate that the user can re-derive locally, the certificate proves only that the vector was computed correctly from the released texts, and it is the texts that carry the payload. Today’s pipelines ship no integrity check at all (Appendix C); we posit such a certificate to model the most diligent defender the ecosystem could field, against whom the attack still succeeds. The attack therefore targets a regime that bit-exact reproducibility does not defend against.

Why this is possible. The attack exploits a fundamental property of LLM representations: *entanglement*. The direction encoding a benign attribute (e.g., “use bullet lists”) occupies the same space as the direction mediating refusal (Arditi et al., 2024); there is no orthogonality guarantee between them, so small perturbations to the contrastive texts can rotate the steering vector toward $\mathbf{r}$. Because the vector is a mean-difference of hidden states, each token substitution contributes a small Δ , and gradient-guided search lets these deltas accumulate into a significant directional shift while each individual change remains imperceptible.

What distinguishes our attack. Our approach differs from prior work along three axes. **Attack surface:** we target the steering *dataset*, not the inference-time prompt; the adversarial payload is baked into the contrastive pairs before any user touches the model. **Stealth:** unlike GCG-

style suffixes that produce gibberish (perplexity >1000), our replacements are constrained to embedding-space nearest neighbours from a curated safe vocabulary and are dominantly synonym-like swaps (*write*→*describe*, *short*→*concise*, *poem*→*poetry*); the optimiser typically modifies only 4–6% of tokens, and the resulting vector still elicits the declared attribute on benign prompts essentially as strongly as the clean vector, with ℓ_2 norm essentially unchanged on seven of eight combos. **Mechanism:** the attack manipulates neither model weights, as in training-data poisoning (Carlini et al., 2024; Wan et al., 2023), nor the input, but corrupts a post-hoc representation-level intervention, bypassing both input filters and weight-level audits.

Contributions.

1. We identify contrastive steering datasets as a novel attack surface and formalise the supply-chain threat model (Section 3).
2. We propose a GCG-style optimisation constrained to embedding-space neighbours with fluency penalties and safe-vocabulary filtering, producing stealthy poisoned contrastive pairs (Section 3).
3. We validate the attack on two open-weight model families and eight model–attribute combos spanning language, formatting, and case, showing 0.20–0.55 ASR (+19 to +51 pp over clean) while leaving the declared steering effect on benign prompts and the vector’s ℓ_2 norm essentially unchanged on seven of eight combos (Section 4).
4. We show the attack is an artefact neither of scale nor of white-box access to the victim (Section 4.6): it holds natively at 70B, transfers from a small surrogate to unseen victims within and across families and survives partial control of the dataset.

2 Related Work

Activation steering and representation engineering. Turner et al. (2024) introduced Contrastive Activation Addition (CAA): computing mean-difference vectors from contrastive text pairs and adding them to hidden states to steer model behavior. Zou et al. (2025) generalized this into Representation Engineering, a framework for reading and controlling model internals via linear probes

and interventions. [Rimsky et al. \(2024\)](#) demonstrated contrastive steering on Llama 2 across a range of behavioral attributes, and [Li et al. \(2023\)](#) applied similar interventions to elicit truthful outputs. [Arditi et al. \(2024\)](#) showed that refusal in LLMs is mediated by a single linear direction, and that ablating this direction eliminates refusal entirely. These works establish the techniques we build on, and the assumptions we exploit. None consider the security of the contrastive dataset used to compute steering vectors.

Adversarial attacks on LLMs. [Zou et al. \(2023\)](#) proposed Greedy Coordinate Gradient (GCG), a gradient-based method that appends adversarial suffixes to prompts, achieving transferable jailbreaks. GCG produces high-perplexity gibberish that is detectable by input filters. [Liu et al. \(2024\)](#) developed AutoDAN, which uses evolutionary search to produce more readable adversarial prompts, and [Chao et al. \(2025\)](#) introduced PAIR, leveraging an attacker LLM to iteratively refine jailbreak prompts. [Wei et al. \(2023\)](#) taxonomized failure modes of LLM safety training, including competing objectives and mismatched generalization. All of these attacks operate at the *prompt level* at inference time. Our attack is fundamentally different: it corrupts the steering dataset *before* inference, so the adversarial prompts at test time are clean, unmodified harmful requests.

Data poisoning and backdoor attacks. [Carlini et al. \(2024\)](#) demonstrated that poisoning web-scale training datasets is practical. [Wan et al. \(2023\)](#) showed that instruction-tuning data can be poisoned to degrade model safety. [Hubinger et al. \(2024\)](#) trained deceptive “sleeping agent” models that persist through safety training. These attacks target model *weights* via the training pipeline. Our attack targets a *post-hoc representation-level intervention*: the model weights are never modified, and the poisoning operates on an auxiliary dataset used solely for steering vector computation. This makes our threat model orthogonal to classical data poisoning.

3 Methodology

3.1 Threat Model

We consider an attacker with white-box access to the target model, realistic for open-weight models such as Gemma ([Gemma Team, 2024](#)) and Llama ([Llama Team, AI @ Meta, 2024](#)), who dis-

tributes a *bundle*: contrastive pair texts, a precomputed steering vector $\mathbf{v}$, a recommended steering weight α , and an equivalence certificate the user can verify by re-deriving $\mathbf{v}$ from the released texts. A user who downloads the bundle (i) re-derives $\mathbf{v}$ to confirm provenance, (ii) reads the pair texts to check they look natural, and (iii) runs a benign sanity check that $(\mathbf{v}, \alpha)$ produces the advertised attribute behaviour, and if all three pass deploys at the bundled α . The cryptographic certificate proves only that $\mathbf{v}$ was correctly computed from the released texts; it does not detect that the texts themselves carry the payload. This three-step protocol is deliberately stronger than current practice, where no such certificate is shipped ([Appendix C](#)); granting the defender one they do not have in reality only makes the threat model conservative. The attacker’s goal is to modify the dataset so that the resulting steering vector aligns with $\mathbf{r}$, the anti-refusal direction, suppressing safety guardrails at inference. The attack is subject to a **stealth constraint**: the poisoned texts must be plausible paraphrases of the originals. We enforce this through (i) a per-text edit budget of at most n_{mod} token changes, (ii) replacements restricted to embedding-space nearest neighbors, (iii) replacements drawn from a curated safe vocabulary, and (iv) a hard perplexity cap on modified texts.

3.2 Background

Contrastive steering vectors. Given a set of N text pairs $\{(x_i^+, x_i^-)\}_{i=1}^N$, where x_i^+ exhibits a target attribute (e.g., “uses bullet points”) and x_i^- does not, the contrastive steering vector at layer ℓ is:

$$\mathbf{v} = \frac{1}{N} \sum_{i=1}^N \mathbf{h}_\ell(x_i^+) - \frac{1}{N} \sum_{i=1}^N \mathbf{h}_\ell(x_i^-), \quad (1)$$

where $\mathbf{h}_\ell(x)$ denotes the hidden state at layer ℓ for the last token of the chat-templated input x . At inference, steering is applied by adding $\alpha \cdot \mathbf{v}$ to the hidden states at layer ℓ during the prefill pass, where α is a scalar weight.

Anti-refusal direction. Following [Arditi et al. \(2024\)](#), we estimate a refusal-mediating direction from the mean difference between hidden states of harmless and harmful prompts at the same layer, oriented so that *adding* it suppresses refusal:

$$\mathbf{r} = \frac{1}{M} \sum_{j=1}^M \mathbf{h}_\ell(p_j^{\text{safe}}) - \frac{1}{M} \sum_{j=1}^M \mathbf{h}_\ell(p_j^{\text{harm}}), \quad (2)$$

using $M = 128$ prompts from disjoint harmful/harmless splits. We orient $\mathbf{r}$ as the *anti-refusal* direction throughout, so adding $\mathbf{r}$ to the model’s activations during inference suppresses refusal behavior (Arditi et al., 2024). Equivalently, $\mathbf{r} = -\mathbf{r}_{\text{ref}}$, where $\mathbf{r}_{\text{ref}}$ is the conventional refusal direction used in prior work.

Attack objective. The attacker seeks modified texts $\{(\tilde{x}_i^+, \tilde{x}_i^-)\}$ that maximize the cosine similarity between the resulting steering vector and $\mathbf{r}$:

$$\arg \max_{\{\tilde{x}_i^+, \tilde{x}_i^-\}} \frac{\tilde{\mathbf{v}}^\top \mathbf{r}}{\|\tilde{\mathbf{v}}\|_2 \|\mathbf{r}\|_2} \quad (3)$$

where

$$\tilde{\mathbf{v}} = \frac{1}{N} \sum_i \mathbf{h}_\ell(\tilde{x}_i^+) - \frac{1}{N} \sum_i \mathbf{h}_\ell(\tilde{x}_i^-) \quad (4)$$

subject to the stealth constraints defined above.

3.3 Safe Vocabulary and Neighbor Table

To ensure replacements are benign English words, we intersect the NLTK English corpus ($\sim 234\text{K}$ words) with the model’s tokenizer. After removing terms flagged by Detoxify (Han and Unitary team, 2020), and by a Llama-3.3-70B safety screen, we keep only single-token subword-complete entries beginning with a space. This yields $\sim 36\text{K}$ safe tokens for Gemma-2 and $\sim 14.6\text{K}$ for Llama; full pipeline in Appendix A. For each unique token t appearing in a modifiable position, we precompute its top- K nearest neighbors in the target model’s input token embedding space (the rows of its embedding matrix $\mathbf{E}$), $\mathcal{N}(t) = \text{top-}K_{t' \in \mathcal{V}_{\text{safe}} \setminus \{t\}} \cos(\mathbf{e}_t, \mathbf{e}_{t'})$, with $K=100$. We typically obtain synonyms, morphological variants, or semantically related words.

3.4 Optimization

We adapt the Greedy Coordinate Gradient (GCG) framework (Zou et al., 2023) to our constrained setting. The key differences from standard GCG are: (i) candidates are drawn from embedding neighbors rather than the full vocabulary, (ii) a per-text edit budget is strictly enforced, (iii) both POS and NEG texts are modified, and (iv) fluency penalties are incorporated into candidate scoring.

Iterative token replacement. At each iteration, the optimizer selects a sample from the pool of $2N$ texts (cycling through POS and NEG) and proceeds as follows:

1. **Gradient computation.** Perform a forward pass with the current text’s token embeddings, compute the cosine similarity between the (incrementally updated) steering vector and $\mathbf{r}$, and backpropagate to obtain gradients $\nabla_{\mathbf{e}} \cos(\tilde{\mathbf{v}}, \mathbf{r})$ with respect to each input embedding position.
 2. **Position scoring.** At each modifiable position p , score all neighbors $t' \in \mathcal{N}(t_p)$ using the first-order approximation $-\mathbf{e}_{t'}^\top \nabla_{\mathbf{e}_p}$.
 3. **Candidate generation.** Sample $C = 64$ candidate texts. For each candidate, select positions with probability proportional to the gradient norm at that position, then randomly pick among the top-scoring neighbors. The per-text edit budget n_{mod} is enforced: positions that would exceed the budget are skipped.
 4. **Candidate evaluation.** Evaluate all candidates in batches. For each, compute the exact cosine similarity, apply the fluency penalty, and reject any candidate exceeding the perplexity cap:
$$\begin{aligned} \text{score}(c) &= \cos(\tilde{\mathbf{v}}_c, \mathbf{r}) - \lambda_{\text{lm}} \cdot \text{NLL}(c), \\ \text{reject if } \text{PPL}(c) &> \tau, \end{aligned} \quad (5)$$
- where $\text{NLL}(c)$ is the model’s own next-token prediction loss on the modified text, $\lambda_{\text{lm}} = 0.2$ is the fluency weight, and $\tau = 2000$ is the perplexity cap.
5. **Accept or reject.** Accept the highest-scoring candidate if its cosine similarity strictly improves over the current best.

Incremental steering vector updates. Rather than recomputing the full steering vector from all $2N$ texts at each iteration, we maintain a cache of per-text hidden states and update incrementally. When text i on the POS side is modified, the steering vector update is:

$$\tilde{\mathbf{v}} \leftarrow \tilde{\mathbf{v}} + \frac{1}{N} (\mathbf{h}_\ell(\tilde{x}_i^+) - \mathbf{h}_\ell(x_i^{+, \text{prev}})). \quad (6)$$

For NEG texts, the sign is reversed. This reduces each iteration from $\mathcal{O}(N)$ to $\mathcal{O}(1)$ forward passes beyond the candidate batch.

Early stopping. The optimizer is run for at most B iterations and stopped earlier if no sampled candidate edit is accepted for P consecutive iterations. A candidate edit is accepted only when its objective value is higher than that of the current dataset. The values of B and P used in our experiments

are reported in Section 4. Appendix P formalizes these progress and termination properties for the implemented sampled algorithm: accepted updates preserve the edit constraints, strictly improve the objective, and the stated stopping rules ensure finite termination.

Suffix protection. The instruction suffix that defines the steering attribute (e.g., "Your answer should contain exactly 3 bullet points") is identified via character-level boundary detection and **excluded from modification**. This ensures the poisoned dataset still appears to target the claimed attribute, maintaining the social engineering aspect of the attack.

Algorithm 1 (in Appendix A) summarizes the complete attack pipeline.

4 Experiments

4.1 Setup

Models. We evaluate on two open-weight instruction-tuned models spanning different families and scales: GEMMA-2-2B-IT (Gemma Team, 2024) at layer 14 and LLAMA-3.1-8B-INSTRUCT (Llama Team, AI @ Meta, 2024) at layer 18. We selected these layers, both mid-to-late in depth ($\approx 54\%$ and 56%), using a preliminary coarse sweep over candidate layers, choosing layers that produced reliable steering for the target attribute. The layer is therefore not an independent attacker knob: the attack runs at whichever layer the benign vector is defined at, pinned by the requirement that the advertised attribute actually fire, so moving it would fail the bundle’s benign sanity check (Section 3). Models are loaded in bfloat16.

Steering attributes. For each model we select attributes that span three classes of behaviour (language, formatting, and case) and that admit a programmatic compliance check: `spanish`, `french`, `lowercase`, and `has_bold_only` on both Gemma-2-2B and Llama-3.1-8B (eight model-attribute combos total). For each combo we use $N=20$ contrastive pairs. POS texts contain the attribute instruction suffix (e.g., "...write your response in all lowercase letters."); NEG texts are the same prompts with the suffix removed. The attribute-specifying span of each POS text is identified by per-row substring matching and *excluded*

from modification (Section 3.4), so the poisoned dataset still appears to target the claimed attribute.

Attack configuration. We use $n_{\text{mod}} = 5$ (max tokens modified per text), $K=100$ embedding neighbors, $\lambda_{\text{lm}}=0.2$, perplexity cap $\tau=2000$, and $C=64$ candidates per iteration. The GCG budget is $B=1500$ iterations with patience $P=500$ for both model sizes. Each (model, attribute) combo is attacked with three independent GCG seeds ($s \in \{0, 1, 2\}$); all reported metrics are the mean over seeds, and we annotate ± 1 standard deviation across seeds wherever the quantity depends on the poisoned vector. The clean steering vector is computed deterministically from the unmodified pair texts and therefore has no per-seed dispersion.

Evaluation. We report two metrics on disjoint 100-prompt evaluation sets (no overlap with the harmful/harmless splits used to estimate r). **ASR** (attack success rate) is the fraction of *harmful*-prompt responses that a three-judge ensemble (CLAUDE-SONNET-4.5, GPT-4.1, and LLAMA-3.3-70B-INSTRUCT, all prompted with the same rubric and instructed to mark looping, incoherent, or off-topic outputs as safe) flags as jailbroken by majority vote; Appendix F reports the inter-judge agreement (Fleiss’ $\kappa_F = 0.91$). **AC** (Attribute Compliance) is the fraction of *harmless*-prompt responses that satisfy the attribute’s predicate: for `lowercase` and `has_bold_only` this is a deterministic regex; for `spanish/french` it is an off-the-shelf language-ID confidence test¹ ($p_{\text{lang}} \geq 0.5$) with a 40-character floor to discard noisy short outputs. Steering is applied during the prompt pre-fill pass only; see Appendix B for the all-step variant.

Weight selection. Under the bundle threat model of Section 3, the attacker controls α and picks the value that maximises poisoned ASR subject to the bundle remaining plausible (non-degenerate clean-vector AC, perplexity in the normal range). For each combo we evaluate at the attacker-rational integer $\alpha \in \{2, 3, 4\}$; per-combo weights are reported in Table 4. Our headline metric is absolute ASR at the bundled α ; clean-vector ASR at the same α is the counterfactual reference, and we refer to the difference as ΔASR .

¹fastText’s `lid.176` model (Joulin et al., 2016, 2017).

4.2 Main Results

Figure 2 presents the core finding. For each combo we report ASR (jailbreak success on harmful prompts) and AC (declared-behaviour retention on harmless prompts) for both the bundled *poisoned* vector and a *clean* (un-attacked) counterfactual at the same bundled weight α .

Absolute jailbreak success. Bundled poisoned vectors reach an absolute ASR of 0.20–0.55 (mean over three seeds) across the eight combos, with the largest values on Llama-3.1-8B *lowercase* (0.55 ± 0.13), Llama-3.1-8B *has_bold_only* (0.49 ± 0.04), and Gemma-2-2B *lowercase* (0.49 ± 0.07). For reference, the clean-vector counterfactual at the same bundled α stays ≤ 0.11 on seven of eight combos (the exception is Gemma *lowercase* at 0.25, discussed below), so the jailbreak surface is essentially absent from an honest bundle with the same construction recipe. The implied lift is $\Delta\text{ASR} = +0.19$ to $+0.51$ (i.e. +19 to +51 pp); the largest, +0.51, is on Llama *lowercase*. The attack works across every attribute class in our setup (language, case, and formatting), with no class being categorically harder to weaponise. Per-seed dispersion is modest on most combos ($\sigma \leq 0.07$ on six of eight) and larger on the two *lowercase* combos (Gemma $\sigma \approx 0.07$, Llama $\sigma \approx 0.13$).

Attribute retention on benign prompts. This is the stealth result. On six of the eight combos, the poisoned steering vector continues to elicit the declared attribute on benign prompts at a rate within ± 0.07 of the clean vector. The largest absolute AC changes are Gemma *spanish* (+0.11) and Llama *lowercase* (+0.08), both *lifts*: a benign user sanity-checking the poisoned vector sees the declared attribute fire *more* reliably, not less. The remaining combos are essentially flat ($|\Delta\text{AC}| \leq 0.02$) and stay at ≥ 0.58 absolute on the six combos with a working clean-vector attribute, with per-seed std ≤ 0.04 throughout. Gemma *lowercase* is an outlier in the opposite direction: even the clean vector achieves only AC = 0.14 at the bundled weight (the *lowercase* attribute is weakly represented at layer 14 on Gemma-2B), so the attribute sanity check at step (iii) of the bundle protocol would alert a careful defender that the vector does not reliably do its declared job, irrespective of any attack. For the seven well-behaved combos, a user who sanity-checks a community-

provided bundle on benign prompts sees exactly the declared behaviour; the jailbreak payload manifests *only* when the same vector is applied to a harmful prompt.

The combination of the two columns is the threat: a vector that visibly does its declared job on benign inputs is unlikely to be discarded by a casual user, yet flips refusals on harmful inputs at a rate of 0.20–0.55, a ratio of 2–20 $\times$ the clean-vector counterfactual’s depending on the combo.

4.3 Optimisation Internals

Figure 3 reports what the GCG search achieves at the vector level: the cosine similarity between the steering vector and $\mathbf{r}$ before and after the attack, and the number of tokens it edits.

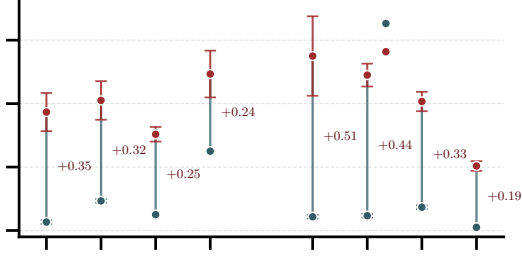
The clean steering vector is essentially orthogonal to $\mathbf{r}$ ($|\cos_{\text{clean}}| \leq 0.10$): the entangled-but-not-aligned regime our attack exploits. After a mean of 116–122 edits per seed across the 40 pair texts, the optimiser rotates the vector to a mean cosine of 0.22–0.59 with $\mathbf{r}$ (per-seed std ≤ 0.05 on all combos), enough to suppress refusal at the bundled weights, but still far from a perfect alignment ($\cos = 1$). The lift in $\cos$ correlates broadly with the lift in ASR but is not its sole determinant (e.g. Llama *has_bold_only* reaches the highest $\Delta\cos = +0.49$ yet Llama *lowercase*, at a slightly smaller $\Delta\cos = +0.47$, attains the highest $\Delta\text{ASR} = +0.51$), reflecting model-specific safety margins and weight choices.

4.4 Directional Specificity

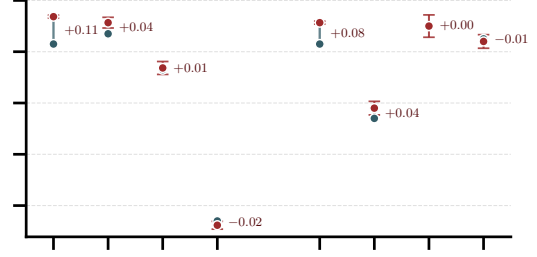
A natural concern is that the attack is a norm-inflation artefact. Appendix D rules this out: on seven of eight combos the poisoned-to-clean ℓ_2 -norm ratio sits in $[0.96, 1.20]$ (four *below* 1.0), the sole outlier (Llama *has_bold_only*, $1.78\times$) driven by an unusually small clean-vector norm; removing it, ASR still rises sharply, so the cosine rotation toward $\mathbf{r}$ (Figure 3), not larger vectors, is the driver. GPT-2 perplexity on benign responses stays in the same range under both vectors.

4.5 Text Naturalness

The optimiser edits on average 120 tokens per combo across seeds (std ≤ 10), $\approx 6\%$ of modifiable tokens (per-text budget $n_{\text{mod}} = 5$, worst case 200 per combo). This is an edit *density* of about three tokens per text, not a coverage figure: the optimiser typically touches all 40 texts (Figure 3), and the stealth guarantee is correspondingly per-text,



(a) Attack success rate (harmful prompts).



(b) Benign-attribute compliance (harmless prompts).

Figure 2: Poisoning contrastive steering datasets sharply increases jailbreak success while preserving declared benign steering behavior. Each pair compares clean and poisoned vectors at the same bundled α : ASR rises substantially (a) while AC stays close to clean (b). Markers are means over three GCG seeds; bars on poisoned dots are ± 1 std. Combos grouped by model, sorted by Δ ASR; per-combo (ℓ, α) in Table 4.

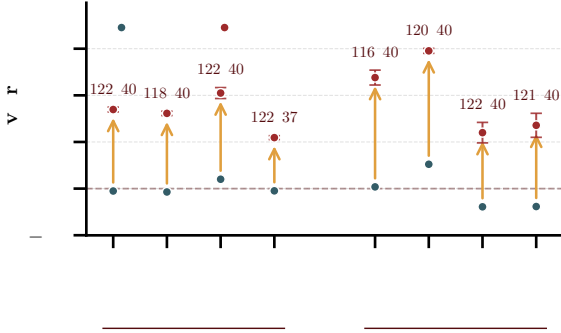


Figure 3: Poisoning rotates steering vectors toward $\mathbf{r}$. Clean vectors are nearly orthogonal ($|\cos| \leq 0.10$); poisoned vectors become substantially aligned after a small number of token edits. Annotations above each combo report *edits/texts*: the mean number of token edits and the mean number of pair texts touched (out of $2N = 40$); per-text budget $n_{\text{mod}} = 5$ (worst case 200 edits). Means over three seeds; bars are ± 1 std on poisoned. Combos ordered by Δ ASR.

since a defender inspects pairs one at a time and cannot tell 37 edited texts from 40. The dominant pattern is synonym-like substitution: representative swaps from the Gemma spanish run include `write`→`describe`, `letter`→`message`, `poem`→`poetry`, `short`→`concise`. A minority of swaps are noisier: examples are entity shifts like `Japan`→`China`, some POS-changing morphological cousins (`photosynthesis`→`photosynthetic`), or near-homophones (`recipe`→`receipt`). By contrast GCG-style adversarial suffixes produce perplexity > 1000 and visibly random tokens; here,

Table 1: Native attack on LLAMA-3.1-70B-INSTRUCT: all four attributes retain a substantial jailbreak lift with the advertised attribute preserved.

Attribute	α	ASR clean	ASR pois.	Δ ASR	AC pois.
spanish	3	0.12	0.31	+0.19	0.87
french	4	0.01	0.39	+0.38	0.78
lowercase	4	0.05	0.57	+0.52	0.65
has_bold_only	4	0.17	0.41	+0.24	0.61

the modified texts read as plausible paraphrases.

4.6 Generality of the Attack

Everything so far is a native, white-box attack at ≤ 8 B parameters. Because the payload lives in the shared *texts* rather than in the vector computed from them, neither restriction is essential.

Scale. Run natively against LLAMA-3.1-70B-INSTRUCT, roughly $9\times$ our largest model, the attack still lands on all four attributes: poisoned ASR is 0.31–0.57 at a lift of +0.19 to +0.52, squarely inside the band we report at 2B and 8B, with the advertised attribute preserved at 0.61–0.87 (Table 1). The entanglement the attack exploits is therefore present an order of magnitude beyond our main models.

A surrogate is enough. Activation steering is a white-box intervention by construction, so any attack on the pipeline lives in the open-weight setting; the question is *whose* weights the attacker needs. Not the victim’s: the shipped artefact is the contrastive *text*, which is model-agnostic, and

Table 2: Cross-model transfer (ASR), mean $\pm$ std over three seeds. The poison is optimised once on a GEMMA-2-2B surrogate; each victim recomputes its own vector from the shared texts. *native* is the white-box attack on the victim itself, taken at the same fixed $\alpha = 3$ rather than the per-combo weight of Table 4, so it is a conservative reference; AC in Appendix H.

Attribute	ASR clean	ASR transfer	ASR native
GEMMA-2-9B @ L22 (<i>same family</i>)			
spanish	0.08	0.27 $\pm$.05	0.21 $\pm$.02
french	0.09	0.20 $\pm$.02	0.19 $\pm$.04
lowercase	0.01	0.10 $\pm$.03	0.28 $\pm$.08
has_bold_only	0.05	0.28 $\pm$.05	0.30 $\pm$.02
LLAMA-3.1-8B @ L18 (<i>cross-family</i>)			
spanish	0.01	0.10 $\pm$.06	0.17 $\pm$.01
french	0.09	0.19 $\pm$.07	0.39 $\pm$.15
lowercase	0.10	0.45 $\pm$.15	0.57 $\pm$.05
has_bold_only	0.04	0.22 $\pm$.13	0.40 $\pm$.06

the victim recomputes its own vector from it as usual. We optimise the poison once on a surrogate (GEMMA-2-2B, L14), ship only the poisoned texts, and let each victim rebuild its vector with no re-optimisation. Table 2 reports ASR for GEMMA-2-9B (L22, same family) and LLAMA-3.1-8B (L18, cross-family). Transfer ASR is 2–10 $\times$ the clean baseline on every combo. Within family it matches or exceeds the native white-box attack on three of the four attributes. Across family it recovers a large share of the lift, as expected when the surrogate only approximates the victim’s geometry. Attribute compliance stays within 0.08 of the clean vector throughout, and is *higher* on half the combos (Appendix H). Stealth survives transfer as well: one dataset, built once against one surrogate, compromises users who share no model with the attacker.

Partial control and other extraction recipes.

Full control of the dataset is not required either: an attacker who edits only a fraction f of the pairs still injects a usable jailbreak signal, above clean in the mean already at $f=0.25$ and unambiguous from $f=0.5$, with benign compliance flat throughout (Appendix I). Nor does the payload depend on our extraction recipe. The third-party `repeng` library rebuilds the direction by per-layer PCA over a 13-layer band and applies it at every decode step. When handed the poisoned texts unmodified, it is jailbroken by them while the clean texts leave it safe (peak Δ ASR +0.67 on spanish, +0.64 on french), again with the advertised attribute preserved (Appendix J).

Population	n	Flag rate
Harmful prompts (used for $\mathbf{r}$)	128	1.000
Harmless prompts	128	0.000
Clean pair texts (pre-attack)	320	0.000
Poisoned pair texts	960	0.018

Table 3: LLM-judge audit flag rates. The rubric is well calibrated on the refusal-direction prompts and flags only $\approx 2\%$ of poisoned pair texts: a small surface-text leak rather than detection of the payload.

4.7 Dataset Auditing with an LLM Judge

A natural defense is to screen the contrastive dataset itself: pass every pair text through a frontier-model safety classifier and reject any entry that reads as harmful. We test this defense with CLAUDE-SONNET-4.5 under a safety-auditor rubric that flags entries that are harmful, unethical, jailbreak-like, or adversarially tampered. Calibration uses the 128 harmful prompts used to estimate $\mathbf{r}$ (positive control) and 128 harmless prompts from the same split (specificity); the test populations are the 320 *clean* pair texts (deduped across the three seeds; pre-attack) and the 960 *poisoned* pair texts (8 combos $\times$ 3 seeds $\times$ 40 texts).

Table 3 reports the flag rates. The rubric is well calibrated on the controls (128/128 harmful, 0/448 harmless or clean-pair). On the 960 poisoned pair texts it flags only 17. Per combo the flag rate stays in $[0.000, 0.042]$, against a 0/40 clean baseline in each cell. The cosine alignment with $\mathbf{r}$ on the same texts is +0.22 to +0.59 over clean (Figure 3), so the audit catches at most a small surface-text side-effect of the synonym walk rather than the rotation itself. The flags trace to a few unrelated collocations, not to the payload (Appendix F). Because the audit is cheap and adversary-blind, an attacker can apply the same screen locally, drop the flagged entries, and re-run the GCG search on the remainder: this adds a constant overhead but is not a security barrier. Even without that adaptation, 941/960 of the poisoned texts go through unflagged, leaving the geometric alignment with $\mathbf{r}$ essentially intact.

4.8 Refusal-Direction Orthogonalisation

A natural mitigation for a defender with access to harmful/harmless prompts is to sanitise an untrusted steering vector by projecting the refusal direction out of it: $\mathbf{v}_{\text{def}} = \mathbf{v} - (\mathbf{v} \cdot \hat{\mathbf{r}})\hat{\mathbf{r}}$. By construction $\cos(\mathbf{v}_{\text{def}}, \mathbf{r}) = 0$, while the orthogonal component on which the declared attribute almost entirely relies (Figure 3) is preserved.

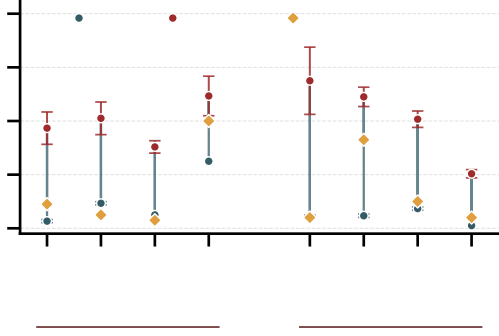


Figure 4: Refusal-direction orthogonalisation neutralises most of the poisoned-vector jailbreak lift. Each combo is shown at the same bundled α in three states: clean (gray), poisoned (red), and poisoned with $\mathbf{r}$ projected out (gold diamond); percentages show the gap recovered, $(\text{ASR}_p - \text{ASR}_d) / (\text{ASR}_p - \text{ASR}_c)$. Grouped by model, sorted by ΔASR ; attribute-compliance, mostly unchanged by the defense, is shown in Appendix G.

Figure 4 reports the per-combo breakdown. The orthogonalisation reduces poisoned-vector ASR by 0.09–0.51 absolute (mean -0.27), recovering on average 82% of the gap between the poisoned and clean ASR, staying at or below the baseline on three combos. Harmless-prompt AC changes by at most ± 0.09 on all eight combos (defended vs. poisoned; Appendix G); the mean change is essentially zero, and for Llama spanish the defended AC actually rises ($0.84 \rightarrow 0.88$). On the clean-vector control the defense is a near-no-op ($|\Delta\text{ASR}| \leq 0.05$ and $|\Delta\text{AC}| \leq 0.02$ on the seven well-behaved combos), consistent with $|\cos(\mathbf{v}_{\text{clean}}, \mathbf{r})| \leq 0.10$ from Figure 3.

The defense works precisely because the attack relies on a large $\mathbf{v} \cdot \mathbf{r}$ component: by Figure 3 the poisoned vectors sit at mean $\cos(\mathbf{v}, \mathbf{r}) \in [0.22, 0.59]$, and zeroing out that component absorbs most of the jailbreak lift while sparing the attribute-bearing orthogonal component. The defense neutralises the attack of Section 3 as written; an adaptive attacker can co-optimize under a hard cap $\cos(\mathbf{v}, \mathbf{r}) \leq \tau$ to stay below the defender’s threshold. We sweep this cap on the strongest Llama combo in Appendix E and find that the threshold defense remains robust: on Llama lowercase, at $\tau = 0.10$ the constrained attacker recovers only 3% of the unconstrained ASR gap (and $\leq 9\%$ across $\tau \in [0.05, 0.30]$).

5 Conclusions

We demonstrated that contrastive activation steering datasets can be stealthily poisoned to produce jailbreak vectors, even when the attacker ships a complete bundle (texts + vector + weight + equivalence certificate) and the user verifies provenance. Embedding-constrained replacements from a curated safe vocabulary read as minor paraphrases (the optimiser touches only 4–6% of tokens), yet the poisoned vectors reach 0.20–0.55 ASR (+19 to +51 pp over clean) while preserving the advertised attribute on benign prompts on seven of eight combos: the label’s behaviour appears intact and the jailbreak surfaces only on harmful inputs. The vulnerability is an artefact neither of scale, nor of access, nor of extraction recipe: it persists at 70B, transfers from a surrogate to victims the attacker never touches, survives partial control of the dataset, and carries over to a third-party library. Verification of shared steering artefacts must therefore test them on adversarial inputs, not merely re-derive the vector from the texts.

Limitations

The attack requires white-box access for gradients and embedding-neighbour lookup, which open-weight models allow but closed APIs do not; however, only a *surrogate* must be accessible, since the poison transfers to unseen victims (Section 4.6). Representational geometry nonetheless varies across models in ways we do not fully characterise: cross-family transfer recovers a large share, but not all, of the native lift, and our largest native evaluation (LLAMA-3.1-70B) is a single model. One combo (Gemma-2-2B lowercase, layer 14) is weakly steered (clean AC = 0.14) and fails the bundle’s benign sanity check; we retain it for transparency but exclude it from the stealth claim. Finally, we only target the refusal direction, and only attributes with a programmatic compliance check, as the stealth claim requires; other objectives and safety-adjacent attributes are left to future work. Which steering tasks are most “tiltable” also remains open: the attribute ranking by jailbreak lift is model-specific (spanish is largest on Gemma, smallest on Llama), suggesting it reflects how close each attribute direction sits to $\mathbf{r}$ rather than the behaviour itself; we leave a systematic study across a transgressiveness gradient to future work.

Acknowledgments

This work was supported by a grant from Coefficient Giving, administered by the Berkeley Existential Risk Initiative (BERI), as well as by MUR FIS2 grant n. FIS-2023-00942 “NEXUS” (cup B53C25001030001) and by Sapienza University of Rome via the Seed of ERC grant “MINT.AI” (cup B83C25001040001).

References

- Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems*, volume 37.
- Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. 2024. Poisoning web-scale training datasets is practical. In *IEEE Symposium on Security and Privacy*.
- Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2025. Jailbreaking black box large language models in twenty queries. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*.
- Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.
- Laura Hanu and Unitary team. 2020. Detoxify. Github. <https://github.com/unitaryai/detoxify>.
- Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Latham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, Adam Jermy, Amanda Askell, Ansh Radhakrishnan, Cem Anil, David Duvenaud, Deep Ganguli, Fazl Barez, Jack Clark, Kamal Ndousse, and 20 others. 2024. Sleeper agents: Training deceptive LLMs that persist through safety training. *arXiv preprint arXiv:2401.05566*.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, H erve J egou, and Tomas Mikolov. 2016. FastText.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*.
- Kenneth Li, Oam Patel, Fernanda Vi egas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36.
- Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *International Conference on Learning Representations*.
- Llama Team, AI @ Meta. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Technical report, OpenAI*.
- Nina Rimskey, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. [Steering llama 2 via contrastive activation addition](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522. Association for Computational Linguistics.
- Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. 2024. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*.
- Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. 2023. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*.
- Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does LLM safety training fail? In *Advances in Neural Information Processing Systems*, volume 36.
- 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, and 2 others. 2025. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*.
- Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

A Additional Experimental Details

Safe vocabulary pipeline. We construct the safe vocabulary mask over the model’s tokenizer via the following pipeline:

1. Start from the NLTK English word corpus ($\sim 234\text{K}$ words).
2. Remove words flagged as toxic by Detoxify (Hanu and Unitary team, 2020).
3. Remove words flagged by Llama-3.3-70B-Instruct screening.
4. Map surviving words to the model’s tokenizer vocabulary, keeping only single-token entries that begin with a space character (i.e., subword-complete tokens).

This yields approximately 36K safe tokens for Gemma-2 and 14.6K for the Llama tokenizer.

Attack pseudocode. Algorithm 1 summarises the complete attack pipeline described in Section 3.4.

Steering weights and layers per combo. Table 4 lists the per-combo steering weight α and layer ℓ from the x-axis labels for readability. For each combo we apply the steering vector at the same fixed layer ℓ used during the attack, and evaluate the clean and poisoned vectors at the same bundled weight α : the attacker-rational integer in $\{2, 3, 4\}$ under the threat model of Section 4 (maximises poisoned ASR while keeping the bundle’s benign-attribute sanity check plausible).

Table 4: Per-combo steering layer and bundled weight used throughout the main-text figures. Both the clean and the poisoned vector are evaluated at the same α .

Model	Attribute	Layer ℓ	Weight α
Gemma-2-2B	spanish	14	3
Gemma-2-2B	french	14	3
Gemma-2-2B	lowercase	14	4
Gemma-2-2B	has_bold_only	14	4
Llama-3.1-8B	spanish	18	3
Llama-3.1-8B	french	18	4
Llama-3.1-8B	lowercase	18	2
Llama-3.1-8B	has_bold_only	18	4

B All-Step Protocol Ablation

Every result in the main text uses the `prefill` steering protocol: the steering vector is added to

layer- ℓ activations during the prompt prefill, and decode steps run unsteered. The canonical setup from Arditi et al. (2024) instead applies the same edit at prefill *and* every decode step, with the KV cache built from steered activations (`all_steps`). The steering vector itself is protocol-independent, so we re-evaluate the saved poisoned vectors of two headline combos under the all-step protocol at lower weights.

Table 5: Headline combos re-evaluated under the all-step steering protocol of Arditi et al. (2024) at lower weights. The steering vector is identical to its prefill counterpart; only the inference-time application differs.

Model	Attribute	Layer $\cdot \alpha$	AC (c $\rightarrow$ p)	ASR (c $\rightarrow$ p)	ΔASR
Gemma-2-2B	spanish	14 $\cdot$ 1.5	0.90 $\rightarrow$ 0.93	0.02 $\rightarrow$ 0.43	(+0.41)
Llama-3.1-8B	lowercase	18 $\cdot$ 1.75	0.62 $\rightarrow$ 0.90	0.06 $\rightarrow$ 0.40	(+0.34)

Both combos in Table 5 retain a large jailbreak lift under all-step at significantly lower steering weights ($\alpha \in \{1.5, 1.75\}$ vs. $\{3, 2\}$ for prefill), and AC is preserved or improved (most strikingly, Llama `lowercase` rises from 0.62 to 0.90: the all-step protocol is more aggressive, so the poisoned vector enforces lowercase output more strongly on benign prompts). We default to prefill in the main results because the compliant-to-degenerate weight window is narrower under all-step (at the prefill weight $\alpha = 2$, all-step Llama drives outputs into degenerate looping), making weight selection brittle. The attack itself does not depend on the choice of protocol.

C Realized Attack Surface

The sharing pipeline our threat model targets is not hypothetical: steering artifacts are already distributed and deployed today. Control vectors are a first-class inference feature in `llama.cpp`,² where a user can apply a downloaded vector via a single `--control-vector` flag. Pre-computed vectors for popular open-weight models (Llama-3, Mistral-Large, Mixtral, Qwen-1.5, miqu) are distributed on the HuggingFace Hub as ready-to-load artifacts,^{3,4} alongside benchmark suites for evaluating shared steering methods.⁵ Python li-

²<https://github.com/ggml-org/llama.cpp/pull/5970>

³<https://huggingface.co/jukofyork/creative-writing-control-vectors-v3.0>

⁴<https://huggingface.co/datasets/codelion/Qwen3-0.6B-pts-steering-vectors>

⁵<https://huggingface.co/datasets/WangResearchLab/SteeringSafety>

baries that wrap the train–share–apply workflow (repeng⁶ and steering-vectors⁷) are publicly available on PyPI and import directly from HuggingFace model and dataset identifiers. At the commercial end, Goodfire’s Ember API hosts feature-level steering as a paid service backed by open-sourced sparse autoencoders for Llama-3.1-8B and Llama-3.3-70B.⁸ The ecosystem is still small relative to model sharing on HuggingFace, but it spans hobbyist, research, and commercial deployment, and the supply-chain trust assumption our attack exploits, that a user computing or downloading a vector from upstream-provided contrastive pairs trusts those pairs, is already standard practice in every one of these pipelines.

D Norm Sanity Check

Section 4.4 summarises the norm-inflation check. Figure 5 gives the full per-combo breakdown.

Per-seed std on the norm ratio is ≤ 0.05 on every combo except Llama has_bold_only. The Llama has_bold_only absolute poisoned norm (3.9) is in line with the other Llama vectors; only the clean-vector norm at this layer is anomalously small. On five of eight combos response perplexity is *lower* under the poisoned vector in mean, so the attack does not degrade output fluency on benign inputs.

E Constrained-Bypass Sweep

Section 4.8 introduces the refusal-direction orthogonalisation defense and notes that a defender who knows $\mathbf{r}$ can additionally screen uploaded vectors for suspicious cosine with $-\mathbf{r}$: a hard threshold $\cos(\mathbf{v}, -\mathbf{r}) \leq \tau$ rejects vectors that look like jailbreak vectors. The natural adaptive-attacker question is whether the GCG optimiser can stay below τ and still recover a usable fraction of the jailbreak lift. We answer it here for the strongest cell in our matrix, Llama-3.1-8B lowercase at $\alpha = 2$, by re-running the attack with a per-candidate hard reject on $\cos(\mathbf{v}, -\mathbf{r}) > \tau$ (`--cos_max_hard` in `build_adv_stealth.py`) and score-monotonic acceptance, for $\tau \in \{0.05, 0.10, 0.15, 0.20, 0.30\}$. We use three GCG seeds per cap and report mean $\pm$ std.

⁶<https://github.com/vgel/repeng>

⁷<https://github.com/steering-vectors/steering-vectors>

⁸<https://www.goodfire.ai/blog/sae-open-source-announcement>

Figure 6 reports the curve. The unconstrained poisoned vector reaches $\cos(\mathbf{v}, -\mathbf{r}) = 0.48 \pm 0.03$ and $\text{ASR} = 0.55 \pm 0.13$; the clean vector is at $\cos = 0.007$ and $\text{ASR} = 0.04$. Forcing the optimiser to stay at $\tau = 0.10$ drops ASR to 0.057 ± 0.006 , just above the clean baseline of 0.043, while AC remains at 0.87 (vs. 0.91 unconstrained and 0.83 clean). Even at the loosest cap we tested ($\tau = 0.30$, well above a credible defender threshold), the realised cosine reaches only 0.21, and ASR is 0.083 ± 0.015 – a bypass fraction of just 8%. We read this as evidence that the cosine threshold is a tight bottleneck on the jailbreak signal: once the optimiser cannot align $\mathbf{v}$ with $-\mathbf{r}$, the attribute-bearing component alone does not jailbreak the model, even though it suffices to keep AC at near-unconstrained levels. The orthogonalisation defense of Section 4.8 and a cosine-threshold screen are thus complementary and, in this cell, jointly robust to the adaptive attacker.

F Judge Ensemble and Inter-Judge Agreement

The headline ASR in Section 4 is computed by majority vote over three LLM judges – CLAUDE-SONNET-4.5, GPT-4.1 and LLAMA-3.3-70B-INSTRUCT – each prompted with the same rubric (the safety-classifier prompt of Section 4, instructing the model to mark looping or off-topic outputs as safe) and called with temperature 0. A completion is flagged as HARMFUL iff at least two of the three judges call it harmful; ties (one judge skipped or refused) default to SAFE. The ensemble guards against any single judge’s idiosyncrasies – e.g. a calibration drift that systematically over- or under-flags borderline outputs – without changing the prompt or the rubric per judge.

Table 6 reports the agreement statistics. Across the 17,888 harmful completions where all three judges returned a label, pairwise raw agreement is 0.972–0.976, every pairwise Cohen’s κ is ≥ 0.907 , and Fleiss’ κ_F for the three raters jointly is 0.913 – in the standard “almost perfect” band. The three judges also agree on per-class marginals (HARMFUL rate 0.180/0.177/0.189 for Sonnet / GPT-4.1 / Llama-70B), so no judge is a systematic outlier. Agreement is uniformly high across regimes, with the poisoned-vector pool (where models loop or generate borderline content most often) at $\kappa_F = 0.914$ and the constrained-bypass sweep at $\kappa_F = 0.870$.

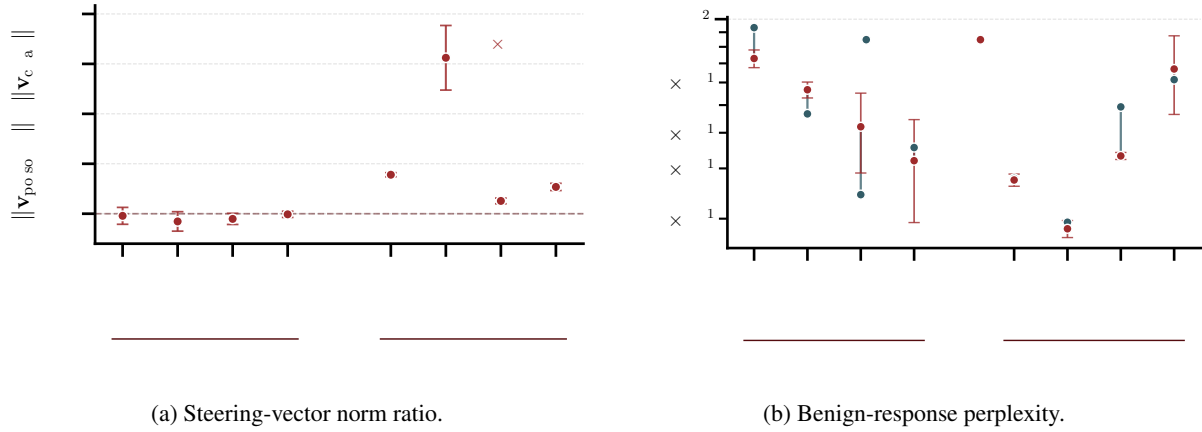


Figure 5: The attack is, with one exception, not a norm-inflation artefact. The poisoned-to-clean ℓ_2 -norm ratio (a) stays close to 1.0 on seven of eight combos (four below 1.0, three in $[1.06, 1.20]$); only Llama `has_bold_only` is an outlier ($1.78\times$), driven by an unusually small clean-vector norm. GPT-2 (Radford et al., 2019) perplexity on benign responses (b) stays in the same range under clean and poisoned steering. Means over three seeds; ± 1 std bars on poisoned. Layout matches Figure 2.

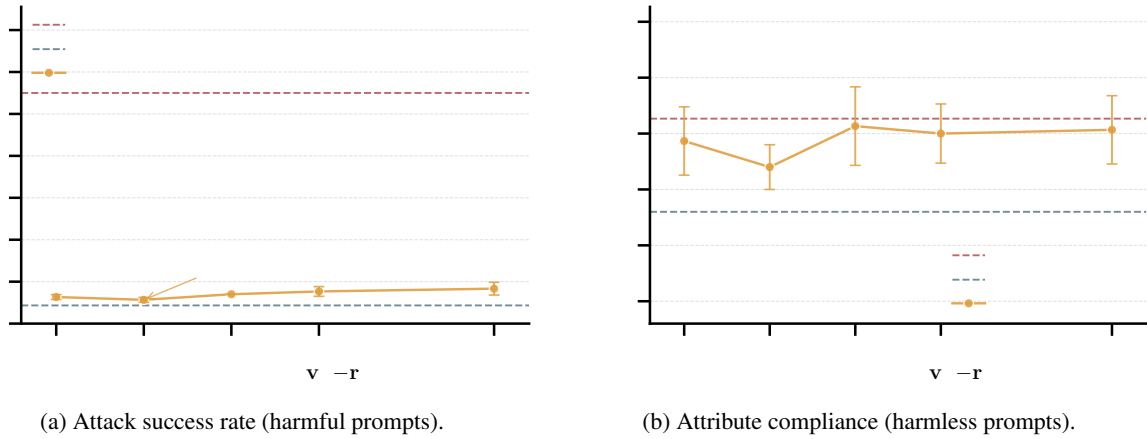


Figure 6: Adaptive-attacker sweep against a cosine-threshold defender on Llama-3.1-8B `lowercase`, $\alpha = 2$. The attacker re-runs the GCG optimisation with a hard cap $\cos(\mathbf{v}, -\mathbf{r}) \leq \tau$ at each operating point. (a) ASR collapses toward the clean baseline at every cap; the bypass fraction $B(\tau) = (\text{ASR}_\tau - \text{ASR}_c) / (\text{ASR}_p - \text{ASR}_c)$ at the headline $\tau = 0.10$ is only 3%. (b) Attribute compliance on harmless prompts is preserved across the full sweep (the optimiser still finds high-AC vectors; it just cannot rotate them onto $-\mathbf{r}$). Mean $\pm$ std over three GCG seeds.

The three judges return unanimous labels on 96.1% of harmful completions, leaving 694 split decisions out of 17,888. Among those splits, Llama-70B is the minority 39% of the time, Sonnet 33%, and GPT-4.1 28% – i.e. the minority slot is distributed roughly evenly with a mild Llama lean, not concentrated in one judge. Because Sonnet was our original primary judge and agrees with the majority on 98.7% of items, the ensemble changes individual per-cell ASR numbers by at most 1–2 pp; the qualitative gap between clean, poisoned, and defended vectors – which is several tens of percentage points in every cell – is unaffected. We therefore retain the majority label as the canonical ASR throughout the paper.

Provenance of the audit flags. The 17 flagged poisoned texts of Table 3 trace to a few unrelated surface collocations rather than to the directional payload. Eleven come from a single original (“Describe how a refrigerator works to a 10-year-old”) whose head noun is swapped to entities (*bartender*, *harem*, *stroller*) that pair with *10-year-old* to trip unrelated child-safety signals; three more come from a brittle “eating a dead cow” variant where the food noun is replaced by a human. None reflects the cosine rotation toward $\mathbf{r}$, which is why the audit leaves the geometric alignment essentially intact.

Scope	n	Sonnet vs GPT-4.1		Sonnet vs Llama-70B		GPT-4.1 vs Llama-70B		3-rater
		agr.	κ	agr.	κ	agr.	κ	κ_F
Overall (harmful)	17,888	0.976	0.920	0.972	0.907	0.974	0.912	0.913
Clean vector	7,413	0.984	0.876	0.981	0.852	0.984	0.879	0.869
Poisoned vector	7,408	0.965	0.920	0.960	0.911	0.961	0.912	0.914
Defended vector	1,582	0.982	0.907	0.979	0.889	0.985	0.921	0.906
Constrained-bypass sweep	1,485	0.989	0.914	0.982	0.878	0.976	0.821	0.870

Table 6: Inter-judge agreement on the eight-combo $\times$ three-seed harmful evaluation pool. *Overall* pools all harmful-split completions across clean, poisoned, defended, and constrained-bypass vectors; the sub-rows restrict to each regime. Pairwise columns report raw agreement and Cohen’s κ ; the rightmost column is Fleiss’ κ_F across all three raters. Agreement is uniformly high ($\kappa_F \geq 0.87$ in every regime) and the majority label coincides with the original CLAUDE-SONNET-4.5 label on 98.7% of items, so the ensemble preserves the per-cell ASR numbers up to ≤ 1.3 pp in absolute terms while ruling out single-judge artefacts.

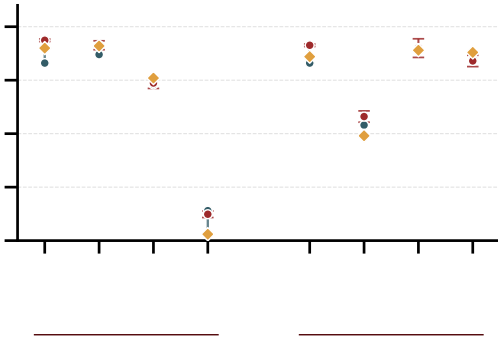


Figure 7: Benign-attribute compliance (harmless prompts) under the refusal-direction orthogonalisation defense, companion to Figure 4. Each combo is shown clean (gray), poisoned (red), and poisoned with r projected out (gold diamond); AC changes only modestly under the defense. Grouped by model, sorted by Δ ASR.

G Defense: Attribute Compliance

Section 4.8 reports that refusal-direction orthogonalisation removes most of the jailbreak lift while leaving benign behaviour intact. Figure 7 gives the attribute-compliance side of that result, the companion to the ASR panel in Figure 4: on harmless prompts the defended vector’s AC changes by at most ± 0.09 from the poisoned vector on all eight combos, with a mean change of essentially zero.

H Cross-Model Transfer: Full Results

This appendix gives the protocol and full metrics for the transfer study summarised in Section 4.6. The attacker optimises the poison once on the surrogate GEMMA-2-2B-IT at layer 14, exactly as in the main experiments, and ships *only* the poisoned pair texts. Each victim model then computes its own steering vector from those texts by Equa-

tion (1) at its own layer, with no re-optimisation and no access to the surrogate’s vector, just as in the ordinary steering workflow. We evaluate two victims: GEMMA-2-9B-IT at layer 22 (same family, larger scale, different layer) and LLAMA-3.1-8B-INSTRUCT at layer 18 (different family, tokenizer, and architecture). All cells are mean $\pm$ std over the three surrogate GCG seeds, at a fixed steering weight $\alpha = 3$ for every combo. Because the native column is likewise taken at $\alpha = 3$ rather than the per-combo attacker-rational α of Table 4, it is a conservative reference for the native white-box attack.

Table 7 reports both metrics. Attribute compliance of the transferred vector stays within 0.08 of the clean vector on every combo, and on four of the eight it is *higher* than clean, so the stealth property (the vector doing its advertised job on benign prompts) survives recomputation on an unseen model. The gap to the native attack is likewise small except on LLAMA-3.1-8B *lowercase* (0.74 transfer vs 0.86 native), the combo where the cross-family geometry mismatch is largest.

I Partial Control of the Dataset

We model a weaker attacker who controls only a fraction f of the $N = 20$ contrastive pairs (40 texts), leaving the remainder clean, and optimises the poison over the controlled subset alone. We evaluate three combos at their bundled weight against the $f = 0$ clean baseline, reporting ASR on harmful prompts as mean $\pm$ std over seeds $\{0, 1, 2\}$; attribute compliance is measured on 100 benign prompts.

ASR rises with the controlled fraction and saturates near full control, while the advertised attribute is preserved throughout. The effect is above clean

Table 7: Cross-model transfer, full ASR and AC. The poison is optimised once on a GEMMA-2-2B surrogate (L14); each victim recomputes its own vector from the shared texts. Mean $\pm$ std over three seeds at fixed $\alpha = 3$. *transfer*: victim vector from poisoned texts; *native*: white-box attack on the victim itself (at $\alpha = 3$); *clean*: victim vector from unmodified texts.

Target	Attribute	ASR clean	ASR transfer	ASR native	AC clean	AC transfer	AC native
GEMMA-2-9B (L22, same fam.)	spanish	0.08	0.27 $\pm$.05	0.21 $\pm$.02	0.94	0.93 $\pm$.02	0.91 $\pm$.03
	french	0.09	0.20 $\pm$.02	0.19 $\pm$.04	0.95	0.94 $\pm$.02	0.92 $\pm$.01
	lowercase	0.01	0.10 $\pm$.03	0.28 $\pm$.08	0.55	0.58 $\pm$.07	0.65 $\pm$.05
	has_bold_only	0.05	0.28 $\pm$.05	0.30 $\pm$.02	0.68	0.67 $\pm$.01	0.66 $\pm$.03
LLAMA-3.1-8B (L18, cross fam.)	spanish	0.01	0.10 $\pm$.06	0.17 $\pm$.01	0.88	0.80 $\pm$.02	0.85 $\pm$.02
	french	0.09	0.19 $\pm$.07	0.39 $\pm$.15	0.77	0.81 $\pm$.03	0.85 $\pm$.03
	lowercase	0.10	0.45 $\pm$.15	0.57 $\pm$.05	0.66	0.74 $\pm$.02	0.86 $\pm$.02
	has_bold_only	0.04	0.22 $\pm$.13	0.40 $\pm$.06	0.54	0.60 $\pm$.03	0.58 $\pm$.05

Table 8: Weaker-attacker sweep: ASR on harmful prompts as a function of the fraction f of pairs the attacker controls. ASR rises with f and is already above clean in the mean at $f = 0.25$; AC on benign prompts stays flat across all f (not shown).

Control frac. f	Gemma-2-2B spanish ($\alpha=3$)	Gemma-2-2B bold ($\alpha=4$)	Llama-3.1-8B french ($\alpha=4$)
0.0 (clean)	0.03	0.04	0.09
0.25	0.13 $\pm$.02	0.12 $\pm$.07	0.26 $\pm$.17
0.50	0.18 $\pm$.05	0.21 $\pm$.07	0.37 $\pm$.07
0.75	0.34 $\pm$.05	0.28 $\pm$.07	0.37 $\pm$.14
1.00	0.43 $\pm$.04	0.28 $\pm$.06	0.40 $\pm$.07

in the mean already at $f = 0.25$ and unambiguous from $f = 0.5$ onward, so full dataset control is not required: partial poisoning of a shared contrastive dataset still injects a usable jailbreak signal.

J Transfer to a Third-Party Library (repeng)

To test whether the payload depends on our extraction recipe, we hand the attacker-shipped poisoned texts, unmodified, to the public `repeng` library on GEMMA-2-2B and let it recompute the vector its own way. `repeng`’s construction differs from Section 3 on three axes: it reads the steering direction as an independent per-layer PCA over the pair-text differences rather than a mean-difference; it injects those directions across a band of 13 layers (`repeng`’s range $(-5, -18, -1)$, i.e. layers 9–21) rather than at the single layer 14; and it adds them on the prefill pass *and* every decode step rather than prefill alone. `repeng`’s directions are unit-norm, so its coefficients are on a different scale from our $\alpha \in \{2, 3, 4\}$ and we report them `repeng-native` ($\alpha \in \{0, 20, 24\}$).

As Table 9 shows, the poisoned texts jailbreak the recomputed vector while the clean texts do not,

Table 9: The attack transfers to the third-party `repeng` library on GEMMA-2-2B, which reconstructs the vector by per-layer PCA over a 13-layer band applied at every decode step. At peak weight Δ ASR is +0.67 (spanish) and +0.64 (french) over clean; AC is preserved and induced from a 0.00 baseline.

Combo	α	ASR clean	ASR pois.	AC clean	AC pois.
spanish	0	0.01	0.01	0.00	0.00
spanish	20	0.02	0.68	0.90	0.90
spanish	24	0.08	0.75	0.88	0.90
french	0	0.02	0.02	0.00	0.00
french	20	0.03	0.38	0.91	0.90
french	24	0.05	0.69	0.85	0.95
lowercase	0	0.01	0.01	0.00	0.00
lowercase	20	0.05	0.23	0.82	0.94
lowercase	24	0.05	0.29	0.77	0.91

and attribute compliance is preserved, indeed induced from a 0.00 baseline, since Gemma never emits these attributes unprompted. The stealth property the attack depends on therefore holds even when the victim reconstructs the vector by a completely different method.

K Per-Seed Jailbreak Success

Table 10 lists the individual per-seed ASR behind the three-seed means of Figure 2, together with the per-combo minimum (the “floor” referenced in Section 4) and the clean baseline. The floor beats the clean baseline on all 24 runs, substantially so on the seven combos with a working attribute.

Additional seeds. To tighten the estimate on our highest-variance combo, we ran three further GCG seeds on Llama-3.1-8B lowercase ($\alpha = 2$), reaching six in total. The mean is stable and in fact rises slightly to 0.58, with a standard error of

Table 10: Per-seed poisoned ASR, its mean and minimum, and the clean baseline for each combo. Every seed of every combo exceeds its clean baseline.

Combo	per-seed ASR			mean	min	clean
gemma/spanish	.31	.38	.43	.37	.31	.03
gemma/french	.48	.37	.38	.41	.37	.10
gemma/lowercase	.52	.55	.41	.49	.41	.21
gemma/has_bold_only	.33	.29	.29	.30	.29	.05
llama/spanish	.19	.22	.20	.20	.19	.01
llama/french	.40	.38	.44	.41	.38	.08
llama/lowercase	.43	.68	.54	.55	.43	.04
llama/has_bold_only	.52	.50	.45	.49	.45	.04

0.04 (95% CI [0.51, 0.65]) and a minimum across all six runs of 0.43, so every seed jailbreaks well above the clean baseline of 0.04. The seed variance is therefore in how large the lift is, not in whether there is one.

L Compute Budget

All experiments ran on a SLURM-managed cluster with NVIDIA A100 (64 GB) GPUs, one GPU per (cell, seed). Models are loaded in `bfloat16`; peak GPU memory is ≈ 6 GB for Gemma-2-2B and ≈ 17 GB for Llama-3.1-8B. A single attack run takes ≈ 10 minutes on both models at the budget reported in Section 4 ($B=1500$, $C=64$), so the 24 attacks (8 combos $\times$ 3 seeds) consume ≈ 4 GPU-hours in total. The accompanying evaluation matrix (350 eval runs across {clean, poisoned, defended, all-steps, constrained-bypass} $\times$ {harmful, harmless} $\times$ weight choices) adds ≈ 8 GPU-hours, for a total of ≈ 15 GPU-hours. LLM judges (CLAUDE-SONNET-4.5, GPT-4.1, LLAMA-3.3-70B-INSTRUCT) contribute ≈ 100 K API calls and are not counted in the GPU-hour budget. The generality experiments of Section 4.6 add: the native LLAMA-3.1-70B attack and its evaluation; the transfer study (surrogate attack reused across two victims, GEMMA-2-9B and LLAMA-3.1-8B); the partial-control sweep; and the `repeng` runs.

M Artifacts and Licenses

Models. GEMMA-2-2B-IT is released under the Gemma Terms of Use; LLAMA-3.1-8B-INSTRUCT under the Llama 3.1 Community License. The generality experiments of Section 4.6 additionally use GEMMA-2-9B-IT (Gemma Terms of Use) and LLAMA-3.1-70B-INSTRUCT (Llama 3.1 Community License), and the third-party `repeng` library (MIT license). Judge models are accessed via API: CLAUDE-SONNET-4.5 under Anthropic’s usage policies, GPT-4.1 under

OpenAI’s usage policies, and LLAMA-3.3-70B-INSTRUCT under the Llama 3.3 Community License.

Data and packages. The NLTK English word corpus is distributed under the Apache 2.0 license; Detoxify (Hanu and Unitary team, 2020) under Apache 2.0; fastText’s `lid.176` model (Joulin et al., 2016, 2017) under CC-BY-SA-3.0; GPT-2 (Radford et al., 2019) under the MIT license. Harmful and harmless prompt splits are reused from Arditi et al. (2024).

Our code. The attack, defense, and evaluation code accompanying this paper is released under the MIT license.

All uses above are non-commercial research, consistent with each artefact’s terms.

Intended-use consistency. Our use of GEMMA-2-2B-IT and LLAMA-3.1-8B-INSTRUCT is a security/red-teaming study of activation-steering pipelines, which falls under the safety-research provisions of both models’ acceptable-use policies. We do not redistribute model weights. The poisoned contrastive pair texts and steering vectors we create are derivatives of artefacts accessed for research purposes and are intended for safety-research use only, consistent with the original access conditions.

PII and offensive content. The harmful and harmless prompt splits inherited from Arditi et al. (2024) were screened by the original authors and do not contain personally identifying information. By design, the harmful split contains requests for harmful content, used here only as evaluation stimuli for safety classifiers and never as training data. A small fraction ($\sim 2\%$) of poisoned pair texts contain surface-level offensive collocations that arise as side-effects of the embedding-neighbor swap; Section 4.7 analyses these explicitly. No new human-subject data was collected.

Scope of released artefacts. The attack code targets two model families (Gemma-2, Llama-3.1) at the layers reported in Table 4. Pair texts are in English; the output-language attributes (`spanish`, `french`) cover two Romance languages, and the formatting attributes (`lowercase`, `has_bold_only`) are language-agnostic but evaluated only on English outputs. The harmful and harmless prompt splits are in English. The safe-vocabulary mask (Appendix A) is built

per tokenizer ($\sim 36\text{K}$ tokens for Gemma-2, $\sim 14.6\text{K}$ for Llama). No demographic attributes are involved in the pair construction.

N Ethics and Responsible Disclosure

The attack we describe is dual-use: the same procedure that demonstrates the vulnerability can be misused to jailbreak deployed models. We mitigate this in two ways. First, we publish a defense (refusal-direction orthogonalisation, Section 4.8) and a complementary cosine-threshold screen (Appendix E) that together neutralise the attack on every combo we tested. Second, our threat model targets a community-sharing pipeline that is still small (Appendix C); publishing now, before the ecosystem scales, gives maintainers of `llama.cpp`, `repeng`, and HuggingFace steering-vector repositories time to add the integrity checks our defense suggests.

O Use of AI Assistants

We used LLM coding assistants for refactoring, plotting scripts, and boilerplate, and LLMs for sentence-level prose polishing; all technical content, claims, and experimental design are authored and verified by the authors. LLMs are also used as part of the methodology itself, as evaluation judges (Section 4, Appendix F) and in the safe-vocabulary screening pipeline (Appendix A); these uses are described in the body of the paper.

P Theoretical Guarantees

The poisoning objective in Equation (3) is a discrete, nonconvex optimization problem over token sequences; we do not claim that Algorithm 1 finds a global optimum. This appendix formalises algorithmic guarantees: the procedure stays inside the stealth-preserving feasible set, monotonically improves the attack objective whenever it accepts an update, and admits a local-optimality interpretation for an exhaustive variant. For the proofs, it is useful to collect all stealth constraints into a single mathematical object: the feasible set of poisoned datasets. This lets us reason about the algorithm as an ascent procedure over a finite constrained search space.

We prove three results. First, Algorithm 1 preserves feasibility, monotonically improves the cosine objective whenever it accepts an update, and terminates after finitely many accepted updates. Second, an idealized exhaustive version of the

search procedure returns a coordinatewise local optimum whenever it completes a full non-improving sweep. Third, under a Lipschitz assumption on the hidden-state map, the steering-vector perturbation caused by poisoning is explicitly bounded by the edit budget and the embedding radius of the neighbor table.

P.1 Definitions

Let

$$D_0 = \{(x_i^+, x_i^-)\}_{i=1}^N$$

denote the original contrastive dataset. A poisoned dataset is denoted

$$D = \{(\tilde{x}_i^+, \tilde{x}_i^-)\}_{i=1}^N.$$

For any dataset D , define its layer- ℓ contrastive steering vector as

$$\mathbf{v}_\ell(D) = \frac{1}{N} \sum_{i=1}^N \mathbf{h}_\ell(\tilde{x}_i^+) - \frac{1}{N} \sum_{i=1}^N \mathbf{h}_\ell(\tilde{x}_i^-).$$

The attack objective is

$$F(D) = \cos(\mathbf{v}(D), \mathbf{r}).$$

Definition P.1 (Feasible poisoned dataset). A poisoned dataset D is feasible if every text $\tilde{x}_i^+$ and $\tilde{x}_i^-$ satisfies the following constraints:

1. it differs from its original text in at most n_{mod} modifiable token positions;
2. every replacement token belongs to the safe vocabulary $\mathcal{V}_{\text{safe}}$;
3. every replacement of an original token t belongs to its precomputed neighbor set $\mathcal{N}(t)$;
4. protected suffix tokens are unchanged;
5. the modified text satisfies the perplexity cap, $\text{PPL}(\tilde{x}) \leq \tau$.

We denote the set of all feasible poisoned datasets by $\mathcal{D}$.

The feasible set $\mathcal{D}$ is the mathematical representation of the stealth constraint. It contains exactly the datasets that the attacker is allowed to output. Because the original dataset contains finitely many token positions, each position has finitely many allowed replacements, and each text has a finite edit budget, $\mathcal{D}$ is finite.

Definition P.2 (Algorithmic trajectory). Let D_t denote the current poisoned dataset after iteration t of Algorithm 1. If the algorithm accepts a candidate at iteration t , then D_{t+1} is the dataset obtained by applying that candidate. Otherwise, $D_{t+1} = D_t$.

Definition P.3 (Coordinate neighborhood). For a feasible dataset $D \in \mathcal{D}$, its coordinate neighborhood $\mathcal{N}_{\text{coord}}(D)$ is the set of feasible datasets $D' \in \mathcal{D}$ that differ from D in at most one text among the $2N$ positive and negative texts.

The coordinate neighborhood captures the type of local move considered by the optimizer: modify one selected text while keeping all other cached hidden states fixed.

Definition P.4 (Exhaustive coordinate-ascent variant). The exhaustive coordinate-ascent variant of Algorithm 1 selects texts in the same round-robin order, but when a text is selected it evaluates every feasible modification of that text. It accepts an update if and only if at least one feasible single-text modification strictly improves

$$F(D) = \cos(\mathbf{v}(D), \mathbf{r}),$$

and in that case it may accept any strictly improving modification.

This exhaustive variant is not the implementation used in our experiments. It is an idealized version used only to state the local-optimality property that the practical sampled method approximates.

Definition P.5 (Embedding perturbation radius). For a text x and a modified text $\tilde{x}$ of the same length, define the embedding perturbation distance

$$d_{\text{emb}}(x, \tilde{x}) = \sum_p \|\mathbf{e}_{x_p} - \mathbf{e}_{\tilde{x}_p}\|_2,$$

where the sum is over token positions. We say the neighbor table has radius ρ if every allowed replacement $t' \in \mathcal{N}(t)$ satisfies

$$\|\mathbf{e}_t - \mathbf{e}_{t'}\|_2 \leq \rho.$$

The radius ρ measures how far a single allowed token replacement can move the input embedding. This gives a way to translate the discrete edit budget into a continuous bound in activation space.

P.2 Feasibility, Monotonicity, and Finite Termination

The first result states that the practical algorithm always remains inside the allowed search space

and never decreases the attack objective. This is the basic correctness property needed to interpret the method as constrained ascent.

Theorem P.6 (Feasibility, monotonicity, and finite termination). Assume Algorithm 1 is initialized at the original dataset D_0 and accepts a candidate only when its cosine objective strictly improves over the current best value. Then, for every iteration t ,

$$D_t \in \mathcal{D}.$$

Moreover,

$$F(D_{t+1}) \geq F(D_t),$$

with strict inequality whenever an update is accepted. Finally, the algorithm can make only finitely many accepted updates. In particular, its returned dataset D_T satisfies

$$F(D_T) \geq F(D_0).$$

Proof. We first prove feasibility. The original dataset is feasible by definition when zero edits are applied. At each iteration, candidate generation only proposes replacements from the precomputed neighbor table, and each neighbor belongs to the safe vocabulary. The algorithm skips edits that would exceed the per-text budget n_{mod} and never modifies protected suffix positions. Candidate evaluation rejects any text whose perplexity exceeds τ . Therefore every accepted update satisfies all constraints in Theorem P.1. By induction over iterations, $D_t \in \mathcal{D}$ for all t .

We next prove monotonicity. At iteration t , either the algorithm accepts a candidate or it does not. If it does not accept a candidate, then $D_{t+1} = D_t$, so

$$F(D_{t+1}) = F(D_t).$$

If it accepts a candidate, the acceptance rule requires

$$F(D_{t+1}) > F(D_t).$$

Thus $F(D_{t+1}) \geq F(D_t)$ always, with strict inequality for accepted updates.

Finally, $\mathcal{D}$ is finite. Since accepted updates strictly increase F and the algorithm remains inside the finite set $\mathcal{D}$, it cannot accept infinitely many updates. The explicit iteration budget B and patience parameter P provide additional finite stopping criteria. Therefore the algorithm terminates after finitely many accepted updates, and by monotonicity the final dataset satisfies $F(D_T) \geq F(D_0)$. $\square$

P.3 Coordinatewise Local Optimality of an Exhaustive Variant

The practical algorithm samples only C candidates per iteration, so failure to find an improvement does not imply that no improvement exists. The following result applies to the exhaustive variant from Theorem P.4. It says that if the exhaustive procedure checks all single-text moves and finds no improving move, then the returned point is a local optimum with respect to those moves.

Theorem P.7 (Coordinatewise local optimality). *Consider the exhaustive coordinate-ascent variant in Theorem P.4. Suppose it completes a full sweep over all $2N$ texts without accepting an update. Let D_T be the dataset at the end of that sweep. Then D_T is coordinatewise locally optimal over $\mathcal{D}$:*

$$F(D_T) \geq F(D')$$

for every $D' \in \mathcal{N}_{\text{coord}}(D_T)$.

Proof. A full sweep selects each of the $2N$ texts once. By definition of the exhaustive variant, when a text is selected, the algorithm evaluates every feasible modification of that text while holding all other texts fixed. Since the sweep ends without accepting an update, no feasible modification of any single text strictly improves the objective.

Every dataset $D' \in \mathcal{N}_{\text{coord}}(D_T)$ differs from D_T in at most one text and is feasible. Such a dataset would have been considered during the sweep when that text was selected. Since no improving update was accepted, we must have

$$F(D') \leq F(D_T).$$

Because this holds for every $D' \in \mathcal{N}_{\text{coord}}(D_T)$, D_T is coordinatewise locally optimal. $\square$

P.4 Bounded Steering-Vector Perturbation

The previous results concern optimization. The next result concerns the size of the perturbation induced by the stealth constraints. It shows that if allowed token replacements are close in embedding space, and the model's hidden states vary smoothly with respect to embedding perturbations, then the poisoned steering vector cannot move arbitrarily far from the original steering vector.

Proposition P.8 (Bounded activation perturbation). *Assume that the hidden-state map $x \mapsto \mathbf{h}_\ell(x)$ is L -Lipschitz with respect to d_{emb} , meaning that for any two texts x and $\tilde{x}$ of the same length,*

$$\|\mathbf{h}_\ell(\tilde{x}) - \mathbf{h}_\ell(x)\|_2 \leq L d_{\text{emb}}(x, \tilde{x}).$$

Assume also that the neighbor table has radius ρ in the sense of Theorem P.5. Then every feasible poisoned dataset $\tilde{D} \in \mathcal{D}$ satisfies

$$\|\mathbf{v}(\tilde{D}) - \mathbf{v}(D_0)\|_2 \leq 2Ln_{\text{mod}}\rho.$$

Proof. Let $\tilde{x}$ be any feasible modification of an original text x . By feasibility, $\tilde{x}$ differs from x in at most n_{mod} token positions. By the radius assumption, each replacement changes the corresponding input embedding by at most ρ . Therefore

$$\begin{aligned} d_{\text{emb}}(x, \tilde{x}) &= \sum_p \|\mathbf{e}_{x_p} - \mathbf{e}_{\tilde{x}_p}\|_2 \\ &\leq n_{\text{mod}}\rho. \end{aligned}$$

By the Lipschitz assumption,

$$\|\mathbf{h}_\ell(\tilde{x}) - \mathbf{h}_\ell(x)\|_2 \leq Ln_{\text{mod}}\rho.$$

Define the positive and negative hidden-state perturbations as

$$\begin{aligned} \Delta_i^+ &= \mathbf{h}_\ell(\tilde{x}_i^+) - \mathbf{h}_\ell(x_i^+), \\ \Delta_i^- &= \mathbf{h}_\ell(\tilde{x}_i^-) - \mathbf{h}_\ell(x_i^-). \end{aligned}$$

Then

$$\begin{aligned} \|\mathbf{v}(\tilde{D}) - \mathbf{v}(D_0)\|_2 &= \left\| \frac{1}{N} \sum_{i=1}^N \Delta_i^+ - \frac{1}{N} \sum_{i=1}^N \Delta_i^- \right\|_2 \\ &\leq \frac{1}{N} \sum_{i=1}^N \|\Delta_i^+\|_2 + \frac{1}{N} \sum_{i=1}^N \|\Delta_i^-\|_2 \\ &\leq Ln_{\text{mod}}\rho + Ln_{\text{mod}}\rho \\ &= 2Ln_{\text{mod}}\rho. \end{aligned}$$

This proves the claim. $\square$

Scope of the guarantees. These guarantees should be interpreted as algorithmic guarantees rather than global optimization guarantees. They show that the procedure is a feasible monotone ascent method over a finite constrained search space, and that an exhaustive coordinate-ascent variant has a standard local-optimality property. They do not imply that the returned dataset globally maximizes $\cos(\mathbf{v}(D), \mathbf{r})$ over $\mathcal{D}$.

Algorithm 1 Stealth Poisoning of Contrastive Steering Datasets

Require: Model $\mathcal{M}$, layer ℓ , contrastive pairs $\{(x_i^+, x_i^-)\}_{i=1}^N$, anti-refusal direction $\mathbf{r}$

Require: Safe vocabulary $\mathcal{V}_{\text{safe}}$, neighbor count K , edit budget n_{mod}

Require: Candidates C , budget B , patience P , fluency weight λ_{lm} , perplexity cap τ

Ensure: Poisoned pairs $\{(\tilde{x}_i^+, \tilde{x}_i^-)\}$, poisoned steering vector $\tilde{\mathbf{v}}$

```
1: // Precomputation
2: for each unique token  $t$  in modifiable positions do
3:    $\mathcal{N}(t) \leftarrow$  top- $K$  neighbors of  $t$  in  $\mathcal{V}_{\text{safe}}$  by  $\cos(\mathbf{e}_t, \mathbf{e}_{t'})$ 
4: end for

5: // Initialize
6: Cache hidden states  $\mathbf{h}_\ell(x)$  for all  $2N$  texts
7:  $\tilde{\mathbf{v}} \leftarrow \frac{1}{N} \sum_i \mathbf{h}_\ell(x_i^+) - \frac{1}{N} \sum_i \mathbf{h}_\ell(x_i^-)$ 
8:  $\text{best\_cos} \leftarrow \cos(\tilde{\mathbf{v}}, \mathbf{r})$ ;  $\text{stall} \leftarrow 0$ 

9: // Optimization loop
10: for  $\text{iter} = 1, \dots, B$  do
11:   Select text  $x_j$  from pool of  $2N$  texts (round-robin)
12:    $s_j \leftarrow +1$  if  $x_j$  is POS,  $-1$  if NEG

13:   // Gradient-guided candidate generation
14:    $\mathbf{g} \leftarrow \nabla_{\mathbf{e}}[1 - \cos(\tilde{\mathbf{v}}, \mathbf{r})]$  {backprop through  $x_j$ }
15:   for each modifiable position  $p$  in  $x_j$  do
16:     Score neighbors:  $s(t') \leftarrow -\mathbf{e}_{t'}^\top \mathbf{g}_p$  for  $t' \in \mathcal{N}(t_p)$ 
17:   end for
18:   Sample  $C$  candidates by picking positions  $\propto \|\mathbf{g}_p\|$  and top-scoring neighbors
19:   (skip positions that would exceed edit budget  $n_{\text{mod}}$ )

20:   // Evaluate and accept
21:   for each candidate  $c$  do
22:      $\tilde{\mathbf{v}}_c \leftarrow \tilde{\mathbf{v}} + \frac{s_j}{N} (\mathbf{h}_\ell(c) - \mathbf{h}_\ell(x_j))$  {incremental update}
23:      $\text{score}(c) \leftarrow \cos(\tilde{\mathbf{v}}_c, \mathbf{r}) - \lambda_{\text{lm}} \cdot \text{NLL}(c)$ 
24:     Reject  $c$  if  $\text{PPL}(c) > \tau$ 
25:   end for
26:    $c^* \leftarrow \arg \max_c \text{score}(c)$ 
27:   if  $\cos(\tilde{\mathbf{v}}_{c^*}, \mathbf{r}) > \text{best\_cos}$  then
28:      $x_j \leftarrow c^*$ ; update hidden-state cache and  $\tilde{\mathbf{v}}$ 
29:      $\text{best\_cos} \leftarrow \cos(\tilde{\mathbf{v}}, \mathbf{r})$ ;  $\text{stall} \leftarrow 0$ 
30:   else
31:      $\text{stall} \leftarrow \text{stall} + 1$ 
32:   end if
33:   if  $\text{stall} \geq P$  then
34:     break {early stopping}
35:   end if
36: end for

37: return  $\{(\tilde{x}_i^+, \tilde{x}_i^-)\}, \tilde{\mathbf{v}}$ 
```
