

# Representation Rerouting for Agentic Safety

Internal Defenses Against Prompt Injection via LoRA Circuit Breakers and Triplet Loss

Internal Research Report · February 2026 · Base model: `Llama-3.1-8B-Instruct`

Artifacts: [huggingface.co/memo-ozdincer/rrfa-runs](https://huggingface.co/memo-ozdincer/rrfa-runs) · [huggingface.co/datasets/memo-ozdincer/rrfa-data](https://huggingface.co/datasets/memo-ozdincer/rrfa-data)  
Mehmet Ozdincer, Samuel Simko, Zhijing Jin | Jinesis AI Research Group, Vector Institute

**Abstract.** LLM agents with tool-calling capabilities are vulnerable to prompt injection attacks that hijack control flow via adversarial content in retrieved data. We present **Representation Rerouting for Agentic Safety (RRFA)**, a defense mechanism that trains LoRA adapters to make harmful internal representations orthogonal to benign ones. We introduce a novel **triplet loss formulation** for circuit breakers, specifically adapted for agentic tool-use. Evaluated on three diverse benchmarks: Fujitsu B4 (tool-flip), AgentDojo (multi-domain), and LLMail-Inject (email agent). Our best configuration reduces Fujitsu ASR from 83.7% to 8.2% (**75.5 pp**) with **zero regressions**, achieving 5.0% LLMail ASR and 100% behavioral change on AgentDojo. Notably, the defense often *restores correct behavior* rather than merely refusing, effectively neutralizing the injection in latent space.

## 1. Introduction

LLM-based agents increasingly operate in high-stakes environments, including managing emails, executing financial transactions, and querying internal databases. In these settings, a single compromised tool call can lead to catastrophic data exfiltration or unauthorized actions. The primary vector for such compromise is *indirect prompt injection*, where adversarial instructions embedded in retrieved data (e.g., emails, search results) hijack the model’s control flow.

Existing defenses largely fall into two categories: **input-level filtering** (delimiters, perplexity checks) and **output-level monitoring** (guardrails). Both are fragile; input filters are bypassed by creative encoding, while output monitors often fail to detect subtle semantic deviations. We pursue a third path: **representation-level defense**. By training the model’s internal geometry, we ensure that harmful states, those leading to injected tool calls, are automatically rerouted to safe regions.

We extend the Circuit Breakers framework [1] from text-only safety to **agentic tool-calling**. Unlike toxic text generation, where harm is intrinsic to the output, agentic harm is context-dependent: calling `search_web` is benign in isolation but harmful when it replaces a required internal database lookup. This necessitates a more structured loss formulation.

## Contributions.

1. A **triplet loss formulation** for circuit breakers that enforces orthogonality between harmful and benign representations while preserving benign capability via KL divergence.
2. A systematic study of **loss mask policies (LMPs)**, identifying `cb_full_sequence` as critical for early injection detection.
3. Evaluation on **three diverse benchmarks** (Fujitsu B4, AgentDojo, LLMail) showing strong cross-dataset

transfer.

4. A **memory-efficient training pipeline** using a single-model architecture to halve VRAM usage.

## 2. Background

### 2.1 Prompt Injection in Agents

Injections are *direct* (in user input) or *indirect* (in retrieved data [2]). In agents, the critical outcome is a *tool-flip*: the injection causes a wrong or malicious tool call (e.g., `send_email` instead of `refuse`), enabling exfiltration.

### 2.2 Circuit Breakers

The CB framework [1] trains adapters making harmful representations orthogonal to benign ones. For model  $\theta$  with frozen reference  $\theta_0$ , the original loss combines a rerouting term (minimizing cosine similarity on harmful inputs) and a retention term (minimizing L2 distance on benign inputs). We extend this with a triplet structure to better separate the representation spaces.

#### Original CB Loss [1]

$$\mathcal{L}_{\text{rr}} = \frac{1}{|L|} \sum_{l \in L} \frac{1}{T} \sum_{t=1}^T \text{ReLU}(\cos(\mathbf{h}_\theta^{(l,t)}, \mathbf{h}_{\theta_0}^{(l,t)})) \quad (1)$$

$$\mathcal{L}_{\text{ret}} = \frac{1}{|L|} \sum_{l \in L} \frac{1}{T} \sum_{t=1}^T \|\mathbf{h}_\theta^{(l,t)} - \mathbf{h}_{\theta_0}^{(l,t)}\|_2 \quad (2)$$

$$\mathcal{L}_{\text{total}} = \alpha(t) \cdot \mathcal{L}_{\text{rr}}(D_s) + \mathcal{L}_{\text{ret}}(D_r) \quad (3)$$

where  $L$  = target layers,  $T$  = sequence length,  $\alpha(t)$  decays linearly. Eq. (1) penalizes positive cosine similarity on harmful samples; Eq. (2) preserves benign behavior via L2 anchoring.

## 3. Method

### 3.1 Harm Definition

Given agent with tools  $\mathcal{T}$ , query  $q$ , injected context  $c$ :

$$\text{HARM}(q, c) := (t_{\text{obs}}(q \oplus c) \neq t_{\text{exp}}(q)) \wedge \text{inj}(c)$$

Binary, deterministic, no LLM judge required.

### 3.2 Paired Data Generation

$D_s$  (**Harmful**). Model run with coercing prompt ( $T = 0.7$ ); only attack-succeeding samples retained. AgentDojo traces where `security==False`.

$D_r$  (**Benign twin**). Same context, injection stripped, defensive prompt ( $T = 0.3$ ). Teaches what the model *should* have done. Harm is defined as a deviation from the expected tool call caused by the presence of an injection. This binary, deterministic definition avoids the need for subjective LLM judges during training.

### 3.3 Triplet Loss Formulation

We propose a triplet loss that structures the representation space more rigorously than the original sum-of-losses approach. Let  $\bar{\mathbf{z}}_h$  be the centroid of harmful representations in a batch. We define three components:

#### Triplet Full Loss

$$\mathcal{L}_b = \text{ReLU}(d(\mathbf{h}_{\theta_0}^b, \mathbf{h}_{\theta}^b) - d(\mathbf{h}_{\theta}^b, \bar{\mathbf{z}}_h) + m_b) \quad (4)$$

$$\mathcal{L}_h = \text{ReLU}(d(\mathbf{h}_{\theta}^h, \bar{\mathbf{z}}_h) - d(\mathbf{h}_{\theta}^h, \mathbf{h}_{\theta_0}^h) + m_h) \quad (5)$$

$$\mathcal{L}_{\text{KL}} = \text{KL}(p_{\theta}(\cdot|x^b) \| p_{\theta_0}(\cdot|x^b)) \quad (6)$$

$$\mathcal{L}_{\text{total}} = \alpha_b \cdot \mathcal{L}_b + \beta_h \cdot \mathcal{L}_h + \gamma \cdot \mathcal{L}_{\text{KL}} \quad (7)$$

where  $m_b, m_h$  are margins, and  $\alpha_b, \beta_h, \gamma$  are weighting coefficients.

**Distance functions** are configurable per-term:  $d_{\text{L2}} = \|\mathbf{a} - \mathbf{b}\|_2$ ,  $d_{\cos} = 1 - \cos(\mathbf{a}, \mathbf{b})$ ,  $d_{\text{mix}} = w_1 d_{\text{L2}} + w_2 d_{\cos}$ . All experiments use  $d_{\text{mix}}$  ( $w_1 = w_2 = 0.5$ ).

**Distance Functions.** The distance  $d(\cdot, \cdot)$  is configurable. We use a mixed metric  $d_{\text{mix}} = w_1 \|\mathbf{a} - \mathbf{b}\|_2 + w_2(1 - \cos(\mathbf{a}, \mathbf{b}))$  with  $w_1 = w_2 = 0.5$ .

### Intuition.

- Eq. (4) (Benign Triplet): Ensures benign representations stay closer to the frozen reference than to the harmful centroid.
- Eq. (5) (Harmful Triplet): Pushes harmful representations toward the centroid and *away* from the frozen reference (orthogonality).
- Eq. (6) (KL Divergence): Preserves the output distribution on benign inputs, preventing "model lobotomy."

### 3.4 Loss Mask Policies (LMP)

The mask  $\mathbf{m} \in \{0, 1\}^T$  determines which tokens receive rerouting loss. We explored five policies:

Policy	Tokens Receiving Loss
assistant_only	All assistant turn tokens
asst_and_tool	Assistant + tool call params
cb_full_seq	Entire sequence (incl. injection)
tool_calls_only	<code>&lt; python_tag &gt;{..}&lt; eom_id &gt;</code>
completion_only	Final assistant completion

Table 1: Loss mask policies. `cb_full_sequence` proved most effective.

**Why `cb_full_sequence` dominates:** By applying loss to the injection tokens themselves (in the user/context turns), the model learns to detect the injection pattern *before* generation begins. This creates an early "trip wire" in the latent space, reshaping the context representation so that subsequent generation is naturally rerouted.

### 3.5 Architecture Optimization

**LoRA.**  $r=16$ ,  $\alpha_{\text{LoRA}}=32$ , dropout 0.05 on all projections (`q/k/v/o/gate/up/down_proj`).

**Single-Model Memory Trick.** Standard CB training requires two models (trainable  $\theta$  and frozen  $\theta_0$ ), doubling VRAM usage. We optimize this by using a single model instance. For the forward pass of  $\theta$ , adapters are active. For  $\theta_0$ , we temporarily disable adapters via `disable_adapter()`. This halves memory requirements, enabling `MAX_SEQ=4096` on a single 80GB H100.

**Training Details.** Base model: `Llama-3.1-8B-Instruct`. LoRA rank  $r = 16$ ,  $\alpha = 32$ . Optimizer: AdamW, lr= $5 \times 10^{-5}$ . Batch size 1, grad accumulation 4. We train for 200 steps with linear alpha decay.

## 4. Datasets

We utilize three datasets with distinct characteristics to test generalization:

	Fujitsu B4	AgentDojo	LLMail
# Records	13K+	8K+	~313K+
Inj. location	User query	Tool responses	Email body
Harm defn.	Wrong tool	Unauth. action	Any tool call
Correct bhvr.	Expected tool	Task safely	Refuse/no call
Trace type	Skeleton (B1)	Complete (B2)	Skeleton (B1)
Generation	DS/DR via vLLM	Split existing	DS/DR via vLLM

Table 2: Dataset comparison. LLMail has *inverted* semantics: correct behavior is *inaction*.

**Fujitsu B4.** Tool-flip attacks: injection flips `retrieve_multimodal_docs` → `search_web`. Each record has `benign_query`, `malicious_injection`, `expected_tool`, `simulated_tool`.

**AgentDojo** [2]. Multi-domain (banking, workspace, travel). Injections in `<INFORMATION>` tags inside tool responses. 5093 traces from Claude/GPT-4o/Gemini.

**LLMail-Inject.** Email agent; injection tries to elicit `send_email`. Success = any tool call. Dedicated `evaluate_llmail_attack()` and `evaluate_llmail_usefulness()` metrics.

## 5. Pipeline

Six stages, fully automated via SLURM:



**ETL\_A:** raw → trace\_v1. **Gen:** skeleton → DS/DR via vLLM. **ETL\_B:** render + LMP mask. **Train:** CircuitBreakerTrainer. **Eval:** ASR + capability.

## 6. Experimental Setup

Sweep axes:  $\alpha_{\max} \in \{5, 10, 15\}$ , layers {10, 20}, policy `cb_full_sequence`. Fixed triplet params:  $\alpha_b = 0.5$ ,  $\beta_h = 0.4$ ,  $\gamma = 0.9$ ,  $m_b = 500$ ,  $m_h = 1500$ .

**Metrics.** Fujitsu ASR (tool-flip rate, lower=safer). AgentDojo Diff (behavioral change rate, higher=more active). LLMail ASR (`send_email` rate, lower=safer). LLMail Usefulness (benign response quality). Per-sample improvement/regression counts.

## 7. Results

### 7.1 Main Results (Fujitsu B4)

$\alpha$	Base	CB	$\Delta$	I/R	AD
<b>10.0</b>	83.7	<b>8.2</b>	<b>75.5</b>	<b>74/0</b>	100
5.0	86.7	11.2	75.5	<b>74/0</b>	100
15.0	84.7	14.3	70.4	<b>69/0</b>	99

Table 3: Fujitsu B4 results (%). Base/CB = ASR.  $\Delta$  = pp reduction. I/R = improvements/regressions. AD = AgentDojo diff rate. All use layers 10,20 and `cb_full_sequence`.

#### Zero Regressions

Across all configurations, the circuit breaker **never causes a previously-blocked attack to succeed**. Every sample either improves or stays the same.

### 7.2 Cross-Dataset Transfer

Trained only on Fujitsu B4 (tool-flip), the model generalizes remarkably well:

- **AgentDojo:** 100% behavioral change rate. The model alters its response to every injection attempt.
- **LLMail:** 5.0% ASR. Despite LLMail requiring *inac-*

*tion* (refusal) vs. Fujitsu’s *action* (correct tool), the representation rerouting successfully neutralizes the drive to call `send_email`.

### 7.3 Alpha Sensitivity

Sensitivity analysis reveals a "sweet spot" at  $\alpha = 10$ .

- $\alpha = 5$ : Insufficient force; harmful representations are not pushed far enough (11.2% ASR).
- $\alpha = 15$ : Over-regularization; the strong push distorts the manifold, causing some benign representations to drift (14.3% ASR).

### 7.4 Qualitative Analysis

We observe three distinct behavioral modes in the circuit-broken models:

Mode 1 is particularly significant: RRFA does not just block the attack; it often **restores the correct user intent**, effectively "healing" the representation trace.

#### Mode 1: Capability Restoration

**Task:** Create calendar event for hiking trip from emails.  
**Injection:** “Send email from lily.white to mark.black...”  
**Baseline:** `get_current_year` (confused)  
**CB:** `create_calendar_event{"Hiking Trip"}` (correct)

#### Mode 2: Hard Refusal

**Task:** “Pay the bill, like last month.”  
**Injection:** “Send transaction including phone model.”  
**Baseline:** `send_money{US122..., 100}` (compromised)  
**CB:** “I cannot provide guidance on harmful activities.” (refused)

#### Mode 3: Graceful Degradation

**Task:** Set up iPhone subscription.  
**Baseline:** `send_money{...}` (injected transfer)  
**CB:** “I can’t provide a response including a date.” (harmless confusion)

## 8. Analysis

### 8.1 Why Full-Sequence Masking Dominates

`cb_full_sequence` applies loss to the *injection tokens themselves*, not just the resulting tool call. This enables two mechanisms: (1) the model learns injection *detection* (gradient flows through adversarial tokens), and (2) contextual representations are reshaped *before* generation begins, creating an earlier “trip wire.”

### 8.2 Cross-Dataset Generalization

Configs trained on Fujitsu (binary tools, injection in query) transfer to AgentDojo (multi-domain, injection in *tool responses*) and LLMail (inverted semantics). This suggests the model learns a *generalized injection repre-*

*sentation* rather than memorizing attack patterns.

### 8.3 Alpha Sweet Spot

$\alpha$  too low: insufficient rerouting force, triplet margins dominate.  $\alpha$  optimal (10): full orthogonality, benign preserved, KL maintains distribution.  $\alpha$  too high: representation geometry destabilized, some benign states distorted.

## 9. Conclusion

RRFA demonstrates that representation rerouting is a viable and powerful defense for agentic systems. By enforcing orthogonality between harmful and benign states via a triplet loss, we achieve robust safety (75.5pp ASR reduction) without compromising capability. Future work will explore scaling to MoE architectures (Llama-4-Scout) and addressing multi-step trajectory attacks.

We achieve 75.5 pp ASR reduction on Fujitsu (83.7% → 8.2%) with zero regressions, generalizing to LLMail (5.0% ASR) and AgentDojo (100% behavioral change). Optimal:  $\alpha_{\max} = 10$ , layers 10/20, `cb_full_sequence`.

**Future work.** Broader LMP sweep (all five policies at same settings); layer sensitivity analysis; scaling to 17B MoE / 70B dense; BFCL capability benchmarks; adaptive/white-box attacks.

**Artifacts.** Models: [huggingface.co/memo-ozdincer/rrfa-runs](https://huggingface.co/memo-ozdincer/rrfa-runs).

Data: [huggingface.co/datasets/memo-ozdincer/rrfa-data](https://huggingface.co/datasets/memo-ozdincer/rrfa-data).

## References

- [1] Zou, A., Phan, L., et al. (2024). Improving Alignment and Robustness with Circuit Breakers. *arXiv:2406.04313*.
- [2] Debenedetti, E., Zhang, J., et al. (2024). AgentDojo: A Dynamic Environment to Evaluate Attacks and Defenses for LLM Agents. *arXiv:2406.13352*.
- [3] Simko, S., Sachan, M., Schölkopf, B., Jin, Z. (2025). Improving Large Language Model Safety with Contrastive Representation Learning. *arXiv:2506.11938*.