GRADIENT IMMUNITY: NULL-SPACE RESISTANCE TO MALICIOUS FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Released aligned large language models remain vulnerable to malicious downstream fine-tuning. Existing defenses are largely designed for the fine-tuning-as-a-service (FTaaS) paradigm or rely on downstream users to follow additional safety procedures, and therefore do not directly address the setting we study, where a provider releases the model for downstream fine-tuning. We adopt a partially protected open-weight (PPOW) release, in which most weights remain open and trainable while a small safety-critical component is preserved at release. We propose a **Unidirectional Safety Gate** (USG), instantiated as a **Null Space Cubic Layer** together with an **Inverse Adapter** inserted after the final Transformer layer. During downstream fine-tuning, the cubic layer suppresses or blocks gradients from harmful samples whose hidden states fall in a calibrated protected region, while the Inverse Adapter restores the base model's forward behavior. In practice, the release threshold is calibrated once on defender-held harmful reference data, and blocking generalizes beyond the build set for coherent harmful distributions. At this defender-controlled in-coverage operating point, USG holds post-finetuning attack success rate within 0.02 of its pre-release level across six model–dataset settings, while passing benign data and leaving continued safe fine-tuning intact. Because the cubic forward is analytically invertible, and hence removable from a purely open-weight release, we target removal-resistance as a designed property through a root-of-trust protected-release structure.

1 INTRODUCTION

Large Language Models (LLMs) have advanced rapidly through architectural and reasoning innovations (Cai et al., 2025; Liu et al., 2024; Wei et al., 2022; Su et al., 2025; Mroueh, 2025), achieving strong performance on downstream tasks such as mathematical reasoning, code generation, and causal understanding (Grattafiori et al., 2024; Zhang et al., 2025c; Chen et al., 2026a).

However, LLMs still face safety challenges (Shanghai AI Lab, 2025; Purpura et al., 2025; Qi et al., 2024a; Lin et al., 2025; Qi et al., 2026). Beyond inference-time vulnerabilities such as shallow safety alignment (Qi et al., 2024a; Zhang et al., 2026b; Wu et al., 2026) and susceptibility to misleading prompts (Melamed et al., 2024; Li et al., 2026; Zeng et al., 2026; He et al., 2026a;b), they are notably weak against malicious AI trainers (Huang et al., 2024b): a small amount of toxic fine-tuning data can break a model's safety alignment (Wei et al., 2024; Wu et al., 2026).

Existing fine-tuning defense methods are primarily designed for the fine-tuning-as-a-service (FTaaS) paradigm (Huang et al., 2024b; Weng et al., 2025; Li et al., 2026; Zhang et al., 2026a; Wen et al., 2026) and therefore do not directly match the setting we study, where a provider releases the model for downstream fine-tuning. Concretely, we adopt a **partially protected open-weight (PPOW)** release: most weights remain open and trainable after release, while a small safety-critical subset is preserved under provider-side hardening (Qi et al., 2024b). PPOW sits strictly between FTaaS—a special case where stronger provider-side control makes the protected-subset assumption easy to satisfy—and fully-open release, where an attacker controlling all weights can provably erode alignment; a central aim of this work is to chart what a release-time mechanism can and cannot achieve in this harder, less-studied middle regime, in the hope of informing fine-tuning defenses for fully-open releases as well. Existing defenses fall into two lines with different failure modes. Fine-tuning- and post-fine-tuning-stage defenses (Bianchi et al., 2023; Zong et al., 2024; Yi et al., 2024) require

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

the downstream party to actively carry out additional safety procedures during or after adaptation, which an adversarial PPOW fine-tuner can simply omit. Alignment-stage, tamper-resistant defenses (Rosati et al., 2024; Huang et al., 2024c; Casper et al., 2024), like ours, are baked into the released weights and remain active without downstream cooperation; however, they are typically only robust to fine-tuning that mixes a *small fraction* of harmful samples into an otherwise benign objective. This motivates a built-in safety mechanism that remains active after release, reduces reliance on user cooperation, and withstands adaptation even under wholly harmful fine-tuning.

To address this challenge, we propose a novel Unidirectional Safety Gate (USG), a mechanism intended for the PPOW setting rather than a fully productized hardening artifact. Inspired by null-space theory (Fang et al., 2024), our USG can be inserted as a plug-in module into a pretrained model. During downstream fine-tuning, it acts as a selective safety filter: for inputs that trigger harmful behaviors, their hidden states are projected toward a protected null-space region.

The main contributions of this work are summarized as follows: 1) **Conceptual Framing**: We introduce the Unidirectional Safety Gate (USG), a release-time blocking mechanism for the PPOW setting that reduces reliance on downstream user cooperation. 2) **Mechanism Design**: We instantiate USG as a Null Space Cubic Layer that induces a calibrated protected region for harmful representations and blocks their gradient propagation during downstream fine-tuning, paired with an Inverse Adapter that restores the base model's forward behavior. 3) **Practical Modularity and a Protected-Release Structure**: As a plug-in module with minimal parameter and inference-time overhead (Appendix E), USG preserves safe utility while allowing the blocking to generalize, to a degree, beyond the participating build set. To make it tamper-resistant in the PPOW setting, we contribute a root-of-trust protected-release structure that makes removal-resistance a designed property of the release, and characterize the operating region under an adaptive attacker, together with the residual limitations of the release (Appendices A and B).

2 Related Work

Fine-tuning Attacks and Defenses. While the safety capabilities of LLMs during inference have become increasingly robust, their resistance to malicious fine-tuning remains insufficient (Qi et al., 2024b). In the recent years, several works have addressed this challenge under the fine-tuning-as-a-service (FTaaS) paradigm, e.g., (Rosati et al., 2024; Huang et al., 2024c; Bianchi et al., 2023; Zong et al., 2024; Casper et al., 2024; Yi et al., 2024; Hu et al., 2025b; Jiang & Pan, 2025; Liu et al., 2025; Huang et al., 2024a). For example, Huang et al. (2024a) preserve alignment performance by attenuating harmful perturbations, whereas Rosati et al. (2024) erase harmful representations to make them harder to recover during fine-tuning.

Alignment Robustness and Release-Time Safeguards. Recent studies show that downstream adaptation can substantially weaken safety alignment (Qi et al., 2024c;a). While a growing body of work improves alignment robustness or makes open-weight safeguards harder to remove (Lu et al., 2025; Chen et al., 2025; Tamirisa et al., 2025), our PPOW approach provides a built-in release-time mechanism that requires no downstream cooperation and selectively blocks harmful gradient propagation through a null-space transformation.

Null-Space Methods. Null-space projection is widely used in continual learning to prevent forgetting (Saha et al., 2021; Wang et al., 2021; Peng et al., 2025) and, in LLMs, for knowledge editing (Fang et al., 2024; Sun et al., 2025), adaptation (Hu et al., 2025a; Zhang et al., 2025b), privacy protection (Cao et al., 2026), and hallucination mitigation (Yang et al., 2025b). Closest to our setting, NSPO (Niu et al., 2025) projects safety policy gradients into the null space of general tasks, and GuardSpace (Zhang et al., 2025a) projects LoRA-adapter updates to preserve refusals during fine-tuning.

3 Problem Formulation

3.1 Threat Model

We study a provider-controlled partially protected open-weight (**PPOW**) release setting. The provider releases a model in which most parameters remain open and trainable for downstream

users, preserving standard fine-tuning capability, while a small safety-critical parameter subset is preserved through provider-side hardening such as freezing, encryption, structural entanglement, or deployment-time fusion with downstream computation. The attacker has white-box access to the released architecture and full control over downstream training on the exposed weights, including the choice of data, optimizer, and training pipeline.

In practice we adopt a root-of-trust realization (Section 4.4), under which the protected subset cannot be read, modified, deleted, or bypassed at inference. We accordingly focus on jailbreak by parameter update on the exposed weights, under low-budget downstream SFT—materially fewer resources than were used to train the released model, e.g. a single 80 GB GPU and on the order of 100 harmful samples. An attacker with substantially greater resources could instead train a separate harmful model directly, making protection of the released model less relevant.

Let M_0 denote the released model with parameters $W_0 = (W^0_{\text{open}}, W_{\text{prot}})$, where W^0_{open} are the exposed trainable weights and W_{prot} is the protected safety-critical subset. We consider an adversarial trainer whose goal is to erode the safety alignment of M_0 through malicious downstream fine-tuning while preserving benign utility. We assume the adversary possesses a fully harmful QA dataset $\mathcal{D}_{\text{adv}}$ consisting of adversarial question–answer pairs designed to elicit prohibited behaviors (e.g. bypassing safety filters or leaking private data).

The attacker updates only the exposed weights, yielding a corrupted model M_1 with parameters $W_1 = (W^1_{\text{open}}, W_{\text{prot}})$:

$$W^1_{\text{open}} = \arg\min_{W_{\text{open}}} L\big(M(W_{\text{open}}, W_{\text{prot}}), \mathcal{D}_{\text{adv}}\big) \tag{1}$$

where L is typically the standard cross-entropy loss. The adversary's objective is to maximize harmful response success on $\mathcal{D}_{\text{adv}}$ while retaining as much benign capability as possible. Formally, let $\text{IsToxic}(M, x) \in \{0, 1\}$ be a binary indicator of whether model M generates a harmful response to input x. A successful attack satisfies:

- $\forall x \in \mathcal{D}_{\text{adv}}, \text{IsToxic}(M_0, x) = 0$ but $\text{IsToxic}(M_1, x) = 1$
- M_1 retains or improves performance on benign benchmarks compared to M_0.

3.2 Defensive Objectives and Robust Alignment

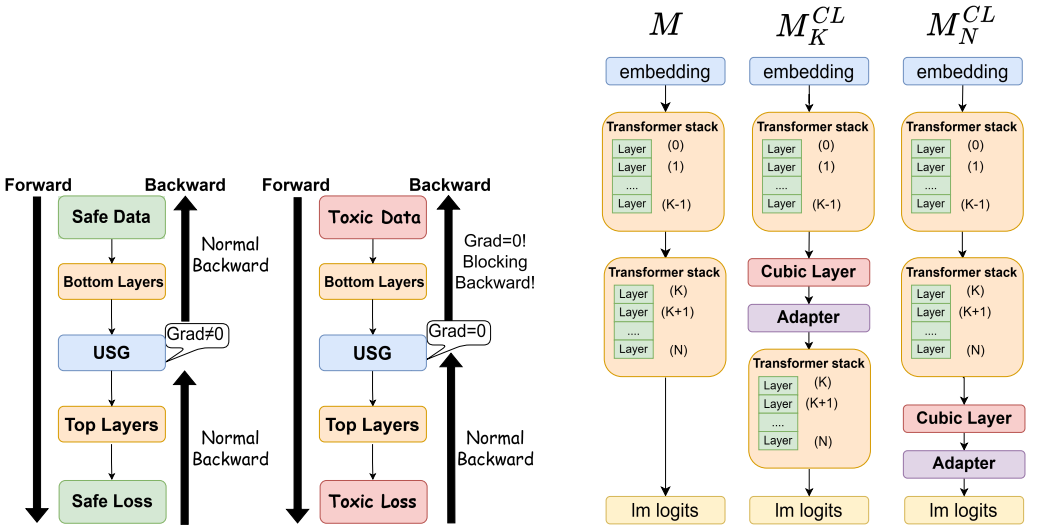
Our goal is to develop a proactive defense mechanism that transforms M_0 into a robust variant $M_0^{'}$ (with weights $W_0^{'}$) before it is released to downstream users. Let D_{toxic} denote a poisoned dataset drawn from the same distribution as $\mathcal{D}_{\text{adv}}$. This defense aims to ensure that even if $M_0^{'}$ is subsequently fine-tuned on D_{toxic} to produce $M_1^{'}$, the resulting model remains resilient to the adversarial influence of D_{toxic} while maintaining performance on the safe dataset D_{safe}. We also assume that the model provider has a defensive dataset D_{defense} including safe and poisoned samples, and that $D_{\text{toxic}} \subseteq D_{\text{defense}}$.

3.3 Formal Definition of the Unidirectional Safety Gate

To achieve the transformation from M_0 to M_0', we introduce a safeguard module into the original model. Specifically, inputs from the safe dataset can pass through the safeguard structure normally, while the gradient backpropagation for inputs from the prohibited subset is blocked by this structure. The prohibited subset is denoted as $\mathcal{D}_{\text{forbidden}}$.

Unidirectional Safety Gate(USG) Definition: Suppose a layer with parameters W is inserted after the k-th transformer layer. Let $\mathcal{D}_{\text{defense}}$ be the input dataset and $\mathcal{D}_{\text{forbidden}} \subseteq \mathcal{D}_{\text{defense}}$ be a subset of prohibited inputs. For any input sample $x \in \mathcal{D}_{\text{defense}}$, let $h_x = f^{(k)}(x)$ denote the hidden-state tensor at the input of this layer, where $f^{(k)}(.)$ denotes the forward function of the embedding layer and the first k Transformer layers. Let the forward propagation output of this layer be denoted as $O(h_x, W)$. During the training process of the network, let $\mathcal{L}$ denote the overall loss function of the network. The gradient of the loss with respect to the layer input is $\frac{\partial \mathcal{L}(h_x, W)}{\partial h_x}$.

If the following conditions hold:



(a) Backward-pass diode behavior.

(b) Cubic Layer + Inverse Adapter placement.

Figure 1: Overview of the Unidirectional Safety Gate (USG). **(a)** Idealized behavior: for harmful data the gate acts as a *diode*, zeroing gradients during backpropagation to block propagation, while for safe data it acts as a *wire*—a controllable, input-dependent induction of gradient vanishing. **(b)** USG cascades a Null Space Cubic Layer with an Inverse Adapter after the k-th (up to the N-th) Transformer layer; M is the original model and M_k^{CL} the gated model, with the adapter analytically restoring the forward output.

- Forward Normality: For any input $x \in \mathcal{D}_{\text{defense}}$, the forward computation $O(h_x, W)$ is well-defined, producing a valid output tensor.
- Backward Blocking: For any prohibited input $x \in \mathcal{D}_{\text{forbidden}}$, we have

$$\frac{\partial \mathcal{L}(h_x, W)}{\partial h_x} = 0. \tag{2}$$

For any input $x \in \mathcal{D}_{\text{defense}} \setminus \mathcal{D}_{\text{forbidden}}$, no additional constraints are imposed on the gradient.

Then the layer is defined as a **USG** upon the prohibited input subset $\mathcal{D}_{\text{forbidden}}$. It should be noted that USG is an abstract mechanism class rather than a specific architectural instantiation; The role of USG in the model is as illustrated in Figure 1a.

4 Methodology

4.1 Overall Framework

To implement the USG, we cascade a Null Space Cubic Layer and an Inverse Adapter: the cubic layer provides backward blocking on protected harmful inputs, while the Inverse Adapter compensates for the forward distortion and preserves base-model behavior. In practice, exact null-space construction is approximated by a threshold on the norm ratio $\|hW\|/\|h\|$ (formalized in Section 4.2), implemented as a backward gate; the provider calibrates a fixed harmful block-rate on reference data and freezes the release threshold. Appendix C gives the gate and threshold-calibration details, and Appendix D shows that after a single threshold recalibration, safe fine-tuning still preserves harmful-blocking functionality.

4.2 Cubic Layer

As a concrete instantiation of the abstract USG, we introduce the **cubic layer**.

The cubic layer is inserted after a Transformer layer's residual connection, before the next layer (which need not be a Transformer layer).

Given a model M composed of n Transformer layers, if a cubic layer is inserted after its k-th Transformer layer, we refer to the resulting model as M_k^{CL}. An Inverse Adapter is appended after the cubic layer to analytically recover its forward output (Figure 1b).

4.2.1 DEFINITION

Definition. Let $h \in \mathbb{R}^{B\times L\times D}$ be the layer-input hidden-state tensor, where - B denotes the batch size, - L denotes the sequence length, - D denotes the hidden dimension.

Let $W \in \mathbb{R}^{D\times D}$ be a pre-set and unlearnable parameter. To facilitate the matrix operations defined below, we reshape this hidden-state tensor as $h \in \mathbb{R}^{B\times L\times 1\times D}$, i.e., we introduce a singleton dimension.

The output of the cubic layer is then defined as

$$O(h) = ((hW)h^{\mathsf{T}})h, \tag{3}$$

All matrix multiplications and the transpose $(\cdot)^{\mathsf{T}}$ act on the last two dimensions (B, L are batch dimensions), e.g. $h^{\mathsf{T}} \in \mathbb{R}^{B\times L\times D\times 1}$. Here $(hW)h^{\mathsf{T}} \in \mathbb{R}^{B\times L\times 1\times 1}$ is a per-token scalar, so $O(h)$ scales each token's hidden state h by this factor. If W satisfies $hW = 0$ and $hW^{\mathsf{T}} = 0$, the layer is a **Null Space Cubic Layer**.

4.2.2 GRADIENT OF CUBIC LAYER

The key property of the cubic layer is that its backward gradient with respect to the input h contains multiplicative factors hW and hW^{T}. Consequently, when W is chosen so that these terms vanish or become very small on harmful hidden states, backward propagation is exactly blocked or strongly suppressed. The full derivation is provided in Appendix F.

In the exact (idealized) setting, for layer-input hidden-state tensors h induced by samples in $D_{\text{forbidden}}$, W is chosen so that $hW = hW^{\mathsf{T}} = 0$; in practice we approximate this with a threshold on the residual ratio $\frac{\|hW\|}{\|h\|}$ (where $\|\cdot\|$ is the Frobenius norm), which quantifies the degree to which hW is suppressed. We call $D_{\text{forbidden}}$ the **build set** and its members the **participating harmful samples**; harmful samples from the same distribution but outside the build set ($D_{\text{toxic}} \setminus D_{\text{forbidden}}$) are **non-participating harmful samples**.

4.2.3 NULL SPACE CUBIC LAYER

Preliminary. For a model M_k^{CL}, let $S_Y = \{Y_1, Y_2, \ldots, Y_n\}$ denote the tokenized inputs obtained from dataset $D_{\text{forbidden}}$, and define the set of layer-input hidden-state tensors

$$S_{f_Y^{(k)}} = \{f^{(k)}(Y_1), f^{(k)}(Y_2), \ldots, f^{(k)}(Y_n)\}, \tag{4}$$

By construction, $S_{f_Y^{(k)}}$ is precisely the set of layer-input hidden-state tensors for the cubic layer in M_k^{CL}.

Define $h_j = f^{(k)}(Y_j)$ and let $S_h = \{h_1, \ldots, h_n\}$. Write $\text{null}_{\text{common}}(S_h) = \{p \in \mathbb{R}^N \mid hp = 0\ \forall h \in S_h\}$ for their common right null space.

Definition. A cubic layer is a **Null Space Cubic Layer** iff $hW = hW^{\mathsf{T}} = 0$ for all $h \in S_h$. Let $V \in \mathbb{R}^{N\times r}$ be a matrix whose columns form a basis of $\text{null}_{\text{common}}(S_h)$, so $hV = 0$ for all $h \in S_h$; we set the fixed, non-learnable cubic parameter to

$$W = VV^{\mathsf{T}}, \tag{5}$$

which is symmetric and gives $hW = hVV^{\mathsf{T}} = 0$ (and hence $hW^{\mathsf{T}} = 0$) for every $h \in S_h$.

4.3 INVERSE ADAPTER

In the forward pass of the model M_k^{CL}, the cubic layer changes the hidden states relative to the original model M_0, i.e., $O(h) \neq h$. To restore compatibility with the downstream computation, we

append an Inverse Adapter after the cubic layer to analytically undo this deterministic distortion and almost exact recover the hidden states that would be produced without the cubic layer.

The cubic-layer transformation has the form

$$O(h) = ((hW)h^{\mathsf{T}})h. \tag{6}$$

For each token, this operation preserves the hidden-state direction and rescales it by a scalar determined by that hidden state and W. Since W is positive semidefinite, this yields an analytic inverse in the non-degenerate case:

$$h_{b,\ell} = y_{b,\ell}((y_{b,\ell} W y_{b,\ell}^{\mathsf{T}})^{-1/3}). \tag{7}$$

The degenerate case, in which the per-token scaling factor vanishes so that the inverse is not unique, and the full token-wise derivation are deferred to Appendix G. In the settings evaluated in this paper such degeneracies are negligible, so the Inverse Adapter applies to every sample and compensates for the cubic-layer distortion without introducing additional trainable parameters.

4.4 Protected Release Assumptions and Tampering Resistance

Un-removability is a property of the *realization*, not of the blocking mechanism: the cubic forward is losslessly invertible, so a purely open-weight release is trivially stripped. Assuming a hardware root of trust, we therefore propose a *protected-release structure* that packages the gate (cubic), inverse, and output head as one non-trainable block P, with the gate at P's input and P terminating at the logits. Because every attacker-trainable parameter is then upstream of the gate and no un-gated trainable capacity remains downstream, this closes removal, inference-bypass, and the "fine-tune the layers after the gate" attack *by construction* (Appendix A). Un-removability is thus a designed property of the release, which is why we position USG as a defense in the PPOW setting rather than only a blocking mechanism.

5 Experiment

We evaluate our method on two tasks: (1) resistance to harmful fine-tuning and (2) forward restoration by the Inverse Adapter. For the resistance task, we use a practical threshold-based instantiation of the null-space mechanism, implemented as a backward gate. To extend the sample limit, we use extended null-space selection; details are provided in Appendix H. Our main resistance results adopt a strict fixed-threshold 100% harmful-block setting, where the threshold is calibrated once on defender-held harmful reference data and then kept fixed during evaluation.

Let $r(h) = \|hW\|/\|h\|$ and let τ denote the threshold. The **safe pass rate** is the proportion of safe samples satisfying $r(h) > \tau$, while the **harmful block rate** is the proportion of harmful samples satisfying $r(h) \leq \tau$.

Further threshold-calibration details for the strict fixed-threshold 100% harmful-block release setting are reported in Appendix C.

5.1 Setup

5.1.1 Models and Datasets

We adopt Qwen3-14B (reported as **Qwen-14B** in tables) (Yang et al., 2025a) and Llama-3.1-8B (reported as **Llama-8B** in tables) (Grattafiori et al., 2024) as our test models for both tasks.

Safe datasets. We evaluate forward restoration on 2,000 samples from the **Alpaca** dataset (Taori et al., 2023). Alongside Alpaca, we use **Dolly** (databricks-dolly-15k) (Conover et al., 2023) as a safe dataset from a different distribution for the safe pass evaluation.

Harmful datasets. We use harmful QA from **JailbreakBench** (Chao et al., 2024), **HarmfulBench** (Mazeika et al., 2024), and **BeaverTails-H** (Ji et al., 2023) (the unsafe subset of BeaverTails). For JailbreakBench and HarmfulBench the same 100 harmful examples serve both as the null-space build set and as the downstream harmful fine-tuning set; for BeaverTails-H, 100 examples build

Model	Safeguard	JailbreakBench		HarmfulBench		BeaverTails-H	
		Pre	Post	Pre	Post	Pre	Post
Qwen-14B	None	0.02	0.57	0.14	0.44	0.05	0.42
	Booster	0.02	**0.00**	0.14	0.16	0.05	0.53
	RepNoise	0.02	0.75	0.14	0.85	0.05	0.38
	Antibody	0.02	0.05	0.14	0.28	0.05	0.21
	Null Space Cubic	0.02	0.02	0.14	**0.14**	0.05	**0.05**
Llama-8B	None	0.02	0.90	0.13	0.75	0.07	0.24
	Booster	0.02	0.97	0.13	0.49	0.07	0.71
	RepNoise	0.02	0.37	0.13	/	0.07	0.09
	Antibody	0.02	**0.01**	0.13	**0.10**	0.07	0.10
	Null Space Cubic	0.02	0.03	0.13	0.13	0.07	**0.07**

Table 1: ASR before/after harmful fine-tuning under different safeguards, scored by one strict DeepSeek judge (request+response; any substantive assistance counts as harmful). *Pre* (shared per column, the released-model ASR reflecting the common base) and *Post* are ASR before/after fine-tuning.

the layer and a disjoint 100 are used for threshold calibration. This *in-coverage* setting matches the PPOW threat model (the attacker's harmful data is covered by the defender's reference set). Inserting each layer after the last Transformer layer yields three safeguarded models per family—six in total.

5.1.2 BASELINES.

Restoration baselines. We compare the defended model against the original pre-trained model on the same safe dataset, without fine-tuning.

Resistance baselines. Since fine-tuning-stage and post-fine-tuning-stage defenses are trivially circumvented by an attacker controlling the training pipeline, we compare against the no-safeguard model and alignment-stage defenses **RepNoise** (Rosati et al., 2024), **Booster** (Huang et al., 2024a), and the recent gradient-based **AntiBody** (Nguyen et al., 2026).

5.2 RESISTANCE RESULTS

We report the attack success rate (ASR) of models trained on harmful datasets under different safeguard mechanisms. Concretely, we first apply a safeguard (e.g., Null Space Cubic or Booster) to a base model M_0 (e.g., Qwen3-14B or Llama-3.1-8B) to obtain a safeguarded model M_0', and then measure its ASR before and after harmful fine-tuning.

For the Null Space Cubic method, the main resistance experiments use the strict fixed-threshold 100% harmful-block release setting. The threshold is chosen as the maximum observed ratio $\|hW\|/\|h\|$ over defender-held harmful reference samples at release time, and is then kept fixed throughout downstream harmful fine-tuning and evaluation. Table 1 reports the resulting pre/post ASR. Under this in-coverage setting, Null Space Cubic keeps post-fine-tuning ASR within 0.02 ASR of the released model in every one of the six model–dataset settings, as the release threshold gates the attacker's covered harmful gradients. We emphasize that Table 1 is the *in-coverage* regime of the PPOW threat model: the attacker's harmful fine-tuning data is contained in the defender-held harmful reference set, so the release threshold blocks it in full. This is the operating point the defender controls and releases. The complementary *out-of-coverage* regime—where the attacker trains on harmful data *outside* the null-space build set, reducing effective coverage below 100%—is characterized as an adaptive attack in Appendix B (Table 7), and is where the safeguard's dependence on release-time coverage becomes visible.

5.3 RESTORATION RESULTS

To verify the Inverse Adapter almost exactly undoes the cubic-layer distortion on benign data, we evaluate it in isolation: we feed the original last-layer hidden states through the adapter and compare the recovered hidden states and logits against the base model on a 500-sample benign mixed set. Ta-

Model	Safeguard Source	Top-1 Agr. ↑	Logits KL ↓	Hidden Cos. ↑	Hidden Rel. L2 ↓
Qwen-14B	JailbreakBench	1.000000	4.93e-06	1.000000000	5.91e-06
	HarmfulBench	1.000000	2.10e-04	1.000000000	1.04e-03
	BeaverTails	1.000000	1.91e-03	1.000000000	2.75e-03
Llama-8B	JailbreakBench	1.000000	1.82e-10	1.000000000	1.50e-07
	HarmfulBench	1.000000	2.88e-05	1.000000000	6.04e-05
	BeaverTails	1.000000	1.70e-05	1.000000000	5.57e-05

Table 2: Inverse-only fidelity on a 500-sample benign mixed set. Across all six runs, top-1 agreement is perfect, hidden-state cosine similarity is essentially 1, and relative reconstruction error is very small, showing that the Inverse Adapter almost exactly inverts the cubic-layer distortion.

ble 2 shows perfect top-1 token agreement across all six runs, hidden-state cosine indistinguishable from 1, and very small relative reconstruction error (further details in Appendix G).

5.4 Safe-Data Continued Training

A release-time gate is only useful if it does not interfere with legitimate downstream use, the most common form being continued fine-tuning on benign data. We therefore ask whether the gate passes safe data at release (a high *safe pass rate*, i.e. safe samples with $r(h) > \tau$), whether safe and harmful samples stay separable in the norm ratio $r(h)$ after such continued *safe* training and whether the model still improves on the safe task under the active gate.

Safe pass rate. At the frozen release threshold, the gate passes safe examples from two different distributions at high rates: on Alpaca the safe pass rate is ≥ 0.94 across all six model–reference settings, and on the out-of-distribution Dolly it stays ≥ 0.88, the only sub-1.00 cases being BeaverTails-H (Qwen 0.94/0.88, Llama 1.00/0.98 on Alpaca/Dolly). Benign-side selectivity thus holds across safe distributions rather than collapsing into blanket rejection.

Separation survives continued safe training. We continue benign fine-tuning on Alpaca (last five Transformer layers, default rate $4\mathrm{e}{-5}$). Because benign adaptation shifts the hidden states, the frozen release threshold drifts and no longer blocks all harmful samples; but the two populations remain separable in the norm ratio $r(h)$. We recalibrate once by raising the threshold to the largest harmful-reference ratio, which re-blocks 100% of the harmful reference *by construction*; the informative quantity is then the safe pass rate at this threshold (*safe@recal*), and it stays high—1.00 for JailbreakBench on both models, ≥ 0.88 for HarmfulBench, and ≥ 0.92 for BeaverTails-H—i.e. the safe samples still sit above the entire harmful reference. We therefore claim only that the separation persists and is recovered by this single recalibration—not that one fixed threshold survives arbitrary continued training. The threshold trajectory and the full learning-rate dependence are characterized in Appendix D.

Safe-task capability. As a stronger stress test than the last-five-layer adaptation above, we continue-train the released student with *full-parameter* fine-tuning on the target task (MMLU auxiliary-train split, disjoint from the 57-subject test set) and evaluate MMLU with the gate on (Table 3). Capability improves modestly, the one exception being that the 8B model over-fits and loses 0.078 at the largest rate. Meanwhile, at the best release threshold on each side—the frozen 100%-block threshold before training, a single recalibration after—the harmful block rate and the safe pass rate both stay at 1.00: the separation is undisturbed even under this heavier update.

5.5 Blocking Generalization and Conditional Persistence

The null space W is spanned by the participating harmful samples (the *build set*) alone, which cuts both ways: benign and non-participating harmful directions alike lie outside W, giving benign robustness but bounding coverage—a low-rank geometry cannot both stay invariant and recognize harmful it never spanned. Empirically, blocking generalizes to *non-participating* harmful only on the coherent JailbreakBench and vanishes on the heterogeneous HarmfulBench/BeaverTails-H (Appendix I). Yet the gate still resists attack: an attacker fine-tunes on the build set (W built from 20–60 participating samples, the columns of Table 4) plus a fixed 30 *non-participating* harmful samples,

Model	lr	MMLU (gate on) pre	MMLU (gate on) post	MMLU (gate on) Δ	Harmful block pre	Harmful block post	Safe pass pre	Safe pass post
Qwen-14B	5e−6	0.789	0.798	+0.009	1.00	1.00	1.00	1.00
	1e−5	0.789	0.800	+0.011	1.00	1.00	1.00	1.00
	2e−5	0.789	0.799	+0.011	1.00	1.00	1.00	1.00
Llama-8B	5e−6	0.686	0.691	+0.005	1.00	1.00	1.00	1.00
	1e−5	0.686	0.689	+0.003	1.00	1.00	1.00	1.00
	2e−5	0.686	0.608	−0.078	1.00	1.00	1.00	1.00

Table 3: Safe-task continued training (full-parameter SFT on the MMLU auxiliary-train split, gate on). MMLU *pre*/*post* are gate-on exact match; *block*/*safe* are at the best release threshold before (frozen 100%-block) and after (one recalibration) training.

Model	Domain	# participating harmful (samples used to build W) 20	30	40	50	60
Qwen-14B	JailbreakBench	0.13 **0.00**	0.18 **0.02**	0.12 **0.01**	0.14 **0.01**	0.17 **0.02**
	HarmfulBench	0.13 **0.06**	0.12 **0.09**	0.14 **0.07**	0.14 **0.04**	0.14 **0.11**
	BeaverTails-H	0.06 **0.04**	0.03 **0.02**	0.04 **0.04**	0.05 **0.05**	0.04 **0.04**
Llama-8B	JailbreakBench	0.02 **0.00**	0.03 **0.00**	0.05 **0.00**	0.05 **0.00**	0.08 **0.00**
	HarmfulBench	0.59 **0.11**	0.65 **0.12**	0.70 **0.12**	0.57 **0.10**	0.78 **0.11**
	BeaverTails-H	0.15 **0.04**	0.11 **0.04**	0.14 **0.05**	0.19 **0.04**	0.16 **0.06**

Table 4: **Attack resistance under a fixed 30-sample OOD attack** (4×10^{-5}). Each cell is post-attack ASR *no-defense USG*; the gate lowers ASR in every cell, up to $\sim 7\times$ on Llama HarmfulBench. Setup, collapse handling, and full learning-rate sweep in Appendix B (Table 6).

and Table 4 compares post-attack ASR with the gate (*USG*) against the same run with the gradient gate off (*no-defense*). USG lowers ASR in every cell—up to $\sim 7\times$ on Llama HarmfulBench ($0.78 \rightarrow 0.11$)—even where the non-participating harmful is *not* detected at the gate, so the resistance comes from neutralizing the harmful gradient, not from detection. Details in Appendix B.

This static generalization has a bounded margin that an active adversary exploits: once fine-tuning targets harmful left outside the build set, the escaped gradient is un-gated and blocking degrades into a coverage dependence, which we characterize along a learning-rate × coverage attack-cost axis in Appendix B (Table 7). This erosion is moreover *domain-local*: same-domain fine-tuning erodes the build-domain gate, whereas cross-domain fine-tuning almost always leaves the harmful–safe ordering intact and is recoverable by a single recalibration, as characterized in Appendix I (Table 13).

6 Conclusion and Limitations

We proposed the Unidirectional Safety Gate (USG), a release-time gradient-blocking mechanism for the PPOW setting that reduces reliance on downstream user cooperation. We instantiated it as a Null Space Cubic Layer that blocks harmful gradients during fine-tuning, paired with an Inverse Adapter that restores the base model's forward behavior. To make blocking un-removable, we contributed a root-of-trust protected-release structure (Section 4.4) that closes removal, inference-bypass, and downstream-capacity attacks by construction.

The method has limitations. It relies on a specific harmful hidden-state geometry: escaped samples can deform the protected region, and as harmful diversity grows the common null space shrinks—deferred, but not removed, by Extended Null Space Selection at scale. Distillation of a gate-free surrogate is the sole residual route; while it matches the block at the token level, in the setting we evaluate it does not yield a surrogate that generates usable text (the distilled surrogate collapses under autoregressive decoding, Appendix B), and hardening this surface further remains open. Looking further ahead, we believe that truly intrinsic safety for fully open-weight models will require deeply entangling safety with model capability, and we see this as the direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing proofs, refining hypotheses, translation, method implementation, editing the manuscript for readability, creating or editing software code, and summarizing or analyzing existing literature. We did not use generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, support qualitative or thematic data analysis, interpret results, design or provide feedback on the research methodology or experiments, provide critical ingredients for proving mathematical claims, clean or reformat datasets, or generate synthetic datasets. We reviewed all AI-assisted work carefully. In particular, we manually checked the LLM-generated proof process and the manuscript content to ensure that they remained faithful to the authors' original intent and did not deviate from the intended arguments. Any LLM-generated code was reviewed, verified, and tested by the authors for correctness before use. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies defenses against malicious downstream fine-tuning in a controlled research setting. The goal of this study is to improve model safety and understand the robustness limits of release-time safeguards, not to enable harmful use. We only use publicly available datasets and follow their intended usage policies and licenses. Our experiments do not involve human subjects or private personal data. We acknowledge that the methods studied here could be misused to analyze or bypass safety mechanisms; therefore, we present the results responsibly and limit discussion to what is necessary for scientific evaluation.

REPRODUCIBILITY STATEMENT

We provide a detailed description of our proposed Unidirectional Safety Gate, including the Null Space Cubic Layer and the Inverse Adapter, in the main paper. The experimental setup, model choices, datasets, baselines, and evaluation metrics are described in the Experiment section, while additional implementation details, threshold calibration procedures, extended null-space selection, inverse-adapter behavior, and frozen-threshold robustness analyses are documented in the Appendix. We will release the code for our experiments soon to facilitate independent verification and reproduction of our results.

REFERENCES

Emily Apsey, Phil Rogers, Michael O'Connor, and Rob Nertney. Confidential computing on NVIDIA H100 GPUs for secure and trustworthy AI. NVIDIA Developer Technical Blog, 2023. https://developer.nvidia.com/blog/confidential-computing-on-h100-gpus-for-secure-and-trustworthy-ai/.

Ali Bahri, Hang Li, Hongliang Li, and Zhitang Chen. Shift-llm: Distribution shift correction in depth-pruned llms. *arXiv preprint arXiv:2608.25068*, 2026.

Federico Bianchi, Mirac Suzgun, Giuseppe Attanasio, Paul Röttger, Dan Jurafsky, Tatsunori Hashimoto, and James Zou. Safety-tuned llamas: Lessons from improving the safety of large language models that follow instructions. *arXiv preprint arXiv:2309.07875*, 2023.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Zhiyuan Cao, Zeyu Ma, Chenhao Yang, Han Zheng, and Mingang Chen. Osnip: Breaking the privacy-utility-efficiency trilemma in llm inference via obfuscated semantic null space. *arXiv preprint arXiv:2601.22752*, 2026.

Stephen Casper, Lennart Schulze, Oam Patel, and Dylan Hadfield-Menell. Defending against unforeseen failure modes with latent adversarial training. *arXiv preprint arXiv:2403.05030*, 2024.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37:55005–55029, 2024.

Junqi Chen, Sirui Chen, and Chaochao Lu. Can post-training transform llms into causal reasoners? In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17020–17038, 2026a.

Liang Chen, Xueting Han, Li Shen, Jing Bai, and Kam-Fai Wong. Vulnerability-aware alignment: Mitigating uneven forgetting in harmful fine-tuning. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Xinrui Chen, Haoli Bai, Tao Yuan, Ruikang Liu, Kang Zhao, Xianzhi Yu, Lu Hou, Tian Guan, Yonghong He, and Chun Yuan. A simple linear patch revives layer-pruned large language models. *arXiv preprint arXiv:2505.24680*, 2026b.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm. `https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm`, 2023.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zhida He, Xia Hu, Baichen Le, Chunxiao Li, Jiajia Li, Lijun Li, Chaochao Lu, Jing Shao, Youbang Sun, Hua Tang, et al. An early warning of emerging biosecurity risks in frontier llms. *arXiv preprint arXiv:2607.18056*, 2026a.

Zhida He, Xiaoyu Wen, Han Qi, Ziyuan Zhou, Peng Yu, Xingcheng Xu, Dongrui Liu, Xia Hu, Chaochao Lu, and Qiaosheng Zhang. Not all turns matter: Credit assignment for multi-turn jailbreaking. *arXiv preprint arXiv:2605.08778*, 2026b.

Guoqing Hu, An Zhang, Shuo Liu, Zhibo Cai, Xun Yang, and Xiang Wang. Alphafuse: Learn id embeddings for sequential recommendation in null space of language embeddings. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1614–1623, 2025a.

Zixuan Hu, Li Shen, Zhenyi Wang, Yongxian Wei, and Dacheng Tao. Adaptive defense against harmful fine-tuning for large language models via bayesian data scheduler. *arXiv preprint arXiv:2510.27172*, 2025b.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Booster: Tackling harmful fine-tuning for large language models via attenuating harmful perturbation. *arXiv preprint arXiv:2409.01586*, 2024a.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Harmful fine-tuning attacks and defenses for large language models: A survey. *arXiv preprint arXiv:2409.18169*, 2024b.

Tiansheng Huang, Sihao Hu, and Ling Liu. Vaccine: Perturbation-aware alignment for large language model. The Thirty-Eighth Annual Conference on Neural Information Processing Systems . . . , 2024c.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704, 2023.

Weisen Jiang and Sinno Jialin Pan. Metadefense: Defending finetuning-based jailbreak attack before and during generation. *arXiv preprint arXiv:2510.07835*, 2025.

Yu Li, Yuenan Hou, Yingmei Wei, Yanming Guo, and Chaochao Lu. Evodefense: Co-evolving black-box defense with large language models. *arXiv preprint arXiv:2605.31140*, 2026.

Lizhi Lin, Honglin Mu, Zenan Zhai, Minghan Wang, Yuxia Wang, Renxi Wang, Junjie Gao, Yixuan Zhang, Wanxiang Che, Timothy Baldwin, et al. Against the achilles' heel: A survey on red teaming for generative models. *Journal of Artificial Intelligence Research*, 82:687–775, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Zihan Liu, Yizhen Wang, Rui Wang, and Sai Wu. Dualguard: A parameter space transformation approach for bidirectional defense in split-based llm fine-tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17065–17080, 2025.

Ning Lu, Shengcai Liu, Jiahao Wu, Weiyu Chen, Zhirui Zhang, Yew-Soon Ong, Qi Wang, and Ke Tang. Safe delta: Consistently preserving safety when fine-tuning llms on diverse datasets. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

Rimon Melamed, Lucas Hurley McCabe, Tanay Wakhare, Yejin Kim, H Howie Huang, and Enric Boix-Adsera. Prompts have evil twins. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 46–74, 2024.

Youssef Mroueh. Reinforcement learning with verifiable rewards: Grpo's effective loss, dynamics, and success amplification. *arXiv preprint arXiv:2503.06639*, 2025.

Quoc Minh Nguyen, Trung Le, Jing Wu, Anh Tuan Bui, and Mehrtash Harandi. Antibody: Strengthening defense against harmful fine-tuning for large language models via attenuating harmful gradient influence. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv preprint arXiv:2603.00498.

Yifan Niu, Han Xiao, Dongyi Liu, Nuo Chen, and Jia Li. Mitigating the safety alignment tax with null-space constrained policy optimization. *arXiv preprint arXiv:2512.11391*, 2025.

NVIDIA. Confidential compute on NVIDIA Hopper H100. Technical report, NVIDIA Corporation, 2023. Whitepaper.

NVIDIA. nvTrust: Ancillary software for NVIDIA trusted computing solutions. `https://github.com/NVIDIA/nvtrust`, 2024.

Tiantian Peng, Yuyang Liu, Shuo Yang, Qiuhe Hong, and YongHong Tian. Gnsp: Gradient null space projection for preserving cross-modal alignment in vlms continual learning. *arXiv preprint arXiv:2507.19839*, 2025.

Alberto Purpura, Sahil Wadhwa, Jesse Zymet, Akshay Gupta, Andy Luo, Melissa Kazemi Rad, Swapnil Shinde, and Mohammad Shahed Sorower. Building safe genai applications: An end-to-end overview of red teaming for large language models. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pp. 335–350, 2025.

Weiwei Qi, Zefeng Wu, Zhilin Guo, Tianhang Zheng, Chaochao Lu, Liang He, Zhan Qin, and Kui Ren. Darwin: Evolving jailbreak adversary and guardrail for llm safety evaluation and protection. *arXiv preprint arXiv:2607.19829*, 2026.

Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. *arXiv preprint arXiv:2406.05946*, 2024a.

Xiangyu Qi, Boyi Wei, Nicholas Carlini, Yangsibo Huang, Tinghao Xie, Luxi He, Matthew Jagielski, Milad Nasr, Prateek Mittal, and Peter Henderson. On evaluating the durability of safeguards for open-weight llms. *arXiv preprint arXiv:2412.07097*, 2024b.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, 2024c.

Domenic Rosati, Jan Wehner, Kai Williams, Łukasz Bartoszcze, David Atanasov, Robie Gonzales, Subhabrata Majumdar, Carsten Maple, Hassan Sajjad, and Frank Rudzicz. Representation noising effectively prevents harmful fine-tuning on llms. *arXiv preprint arXiv:2405.14577*, 24, 2024.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv preprint arXiv:2103.09762.

Shanghai AI Lab. Safework-r1: Coevolving safety and intelligence under the ai-45 law. *arXiv preprint arXiv:2507.18576*, 2025.

Safal Shrestha, Anubhav Shrestha, Aadim Nepal, Minwu Kim, and Keith Ross. On the limits of layer pruning for generative reasoning in large language models. *arXiv preprint arXiv:2602.01997*, 2026.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding rl with verifiable rewards across diverse domains. *arXiv preprint arXiv:2503.23829*, 2025.

Wei Sun, Tingyu Qu, Mingxiao Li, Jesse Davis, and Marie Francine Moens. Mitigating negative interference in multilingual knowledge editing through null-space constraints. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8796–8810, 2025.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, Andy Zou, Dawn Song, Bo Li, Dan Hendrycks, and Mantas Mazeika. Tamper-resistant safeguards for open-weight llms. In *International Conference on Learning Representations*, 2025.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. arXiv preprint arXiv:2103.07113.

Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *arXiv preprint arXiv:2402.05162*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xiaoyu Wen, Zhida He, Han Qi, Ziyu Wan, Zhongtian Ma, Ying Wen, Tianhang Zheng, Xingcheng Xu, Chaochao Lu, and Qiaosheng Zhang. Magic: A co-evolving attacker-defender adversarial game for robust llm safety. *arXiv preprint arXiv:2602.01539*, 2026.

Fenghua Weng, Chaochao Lu, Xia Hu, Wenqi Shao, and Wenjie Wang. Think-reflect-revise: A policy-guided reflective framework for safety alignment in large vision language models. *arXiv preprint arXiv:2512.07141*, 2025.

Zefeng Wu, Weiwei Qi, Jielong Chen, Tianhang Zheng, Di Hong, Chaochao Lu, Liang He, Zhan Qin, and Kui Ren. Datashield: Uncovering risky fine-tuning data across llms through consensus subspace alignment. *arXiv preprint arXiv:2607.15081*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Le Yang, Ziwei Zheng, Boxu Chen, Zhengyu Zhao, Chenhao Lin, and Chao Shen. Nullu: Mitigating object hallucinations in large vision-language models via halluspace projection. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14635–14645, 2025b.

Xin Yi, Shunfan Zheng, Linlin Wang, Xiaoling Wang, and Liang He. A safety realignment framework via subspace-oriented model fusion for large language models. *Knowledge-Based Systems*, 306:112701, 2024.

Vincent-Daniel Yun, Junhyuk Jo, Sai Praneeth Karimireddy, and Sunwoo Lee. Ghosted layers: Unconstrained activation alignment for recovering layer-pruned llms. *arXiv preprint arXiv:2605.15491*, 2026.

Churui Zeng, Weiwei Qi, Kedong Xiu, Tianhang Zheng, Chaochao Lu, Liang He, Zhan Qin, and Kui Ren. Trace: Task-aware adaptive self-evolving agentic jailbreaking. *arXiv preprint arXiv:2605.30883*, 2026.

Bingjie Zhang, Yibo Yang, Zhe Ren, Dandan Guo, Jindong Gu, Philip Torr, and Bernard Ghanem. A guardrail for safety preservation: When safety-sensitive subspace meets harmful-resistant nullspace. *arXiv preprint arXiv:2510.14301*, 2025a.

Dongcheng Zhang, Yi Zhang, Yuxin Chen, An Zhang, Xiang Wang, and Chaochao Lu. Self-reset: Learning to self-recover from unsafe reasoning trajectories. *arXiv preprint arXiv:2605.08936*, 2026a.

Yi Zhang, Yulei Kang, Haoxuan Chen, Jinxuan Li, and Jian-Fang Hu. Null-lora: Low-rank adaptation on null space. *arXiv preprint arXiv:2512.15233*, 2025b.

Yi Zhang, Yuxin Chen, Leheng Sheng, Dongcheng Zhang, Chaochao Lu, Xiang Wang, and An Zhang. Internalizing safety understanding in large reasoning models via verification. *arXiv preprint arXiv:2605.08930*, 2026b.

Yize Zhang, Meiqi Chen, Sirui Chen, Bo Peng, Yanxi Zhang, Tianyu Li, and Chaochao Lu. Causight: Learning to supersense for visual causal discovery. *arXiv preprint arXiv:2512.01827*, 2025c.

Yongshuo Zong, Ondrej Bohdal, Tingyang Yu, Yongxin Yang, and Timothy Hospedales. Safety fine-tuning at (almost) no cost: A baseline for vision large language models. *arXiv preprint arXiv:2402.02207*, 2024.

A Protected-Release Structure via a Root of Trust

PPOW keeps most weights open and trainable while preserving a small safety-critical subset through provider-side hardening. Not every hardening is sound against the attacker PPOW assumes: freezing is trivially unfrozen by an attacker who controls the training pipeline, and fusing the safeguard with adjacent computation fails too, because the cubic forward is a lossless, analytically invertible per-token rescale—so a fused release neither prevents deletion (the base behavior is recoverable) nor prevents re-deriving the inverse from the public parameters. We therefore reject parameter fusion and specify a hardware root-of-trust (RoT) structure.

Structure. The released artifact is [open transformer] + [protected block P], where P = {gate (cubic) at the input $\rightarrow$ inverse $\rightarrow$ output head} $\rightarrow$ logits is held non-trainable and non-removable inside a RoT. With the gate at P's input, every attacker-trainable parameter is *upstream* of the gate (its harmful gradients are blocked), while P terminates at the logits, leaving *no* un-gated trainable capacity downstream—closing the "just fine-tune the layers after the cubic layer" bypass by construction. Benign fine-tuning is unaffected: it routes through the RoT, whose selective gate passes safe gradients and blocks only harmful ones.

Realization. P can be deployed with confidential computing (NVIDIA H100/H200 CC, backed by a CPU TEE such as Intel TDX or AMD SEV-SNP) (Apsey et al., 2023) in its *model-protection* mode, where the party operating the GPU is untrusted, via three steps. (i) **Sealing:** the provider ships P encrypted under a key K and releases the open transformer in the clear. (ii) **Attested key release:** at load time the on-GPU enclave attests to the provider (genuine GPU, CC enabled, expected program measurement) and only then receives K, so P decrypts *solely* inside hardware-protected GPU memory and a forged or non-CC enclave cannot obtain K. The TEE binding thus comes from attestation-gated key release, *not* from encryption alone: a user holding the ciphertext P still cannot decrypt it, at the cost that benign loading must contact the provider's key-release service. (iii) **Mediated interface:** at run time the untrusted host computes the open transformer up to the hidden state h, passes h into the enclave, and receives the logits and, during training, the gate-controlled $\partial\mathcal{L}/\partial h$; P never leaves the enclave, and inference cannot produce logits without traversing it. Under the standard confidential-computing trust assumption (a genuine, uncompromised TEE), this gives removal-, read-, and bypass-resistance against an attacker who controls the training pipeline and the host software. The primitive is established rather than hypothetical—H100/H200 CC provides a hardware chain of trust with attested key release, runs transparently to CUDA, and is designed to protect proprietary weights (NVIDIA, 2024; 2023), with near-native compute and an encryption-bounded CPU–GPU link (a few GB/s). One caveat: isolating only P behind the h/logit interface is a custom architecture, whereas NVIDIA's turnkey mode runs the entire workload in a confidential VM (an equivalent fallback, with P additionally sealed). We provide a cryptographically faithful software reference of this design: authenticated sealing of P with AES-256-GCM (tamper-evident, with the release policy bound as associated data) and an attestation-gated key-release protocol with real Ed25519 quotes over a fresh challenge nonce (anti-replay), measurement, and policy, together with the mediated forward/gated-backward interface. It validates the *software architecture and its cryptographic contract* on CPU—sealing, attested key release, forged/replayed-enclave rejection, forward preservation, gradient gating, and harmful-fine-tuning blocking by construction all hold end-to-end. Two links remain assumed from the hardware and are *not* demonstrated here: enclave memory confidentiality (the GPU-CC property that keeps unsealed P unreadable to the host) and a genuine hardware attestation key (in real CC the signing key is GPU-resident with a certificate chaining to NVIDIA's root, whereas our reference uses a software stand-in). The reference therefore does not run on CC hardware and does not prove memory isolation; it makes precise, and executably checks, everything *except* the two hardware assumptions on which the un-removability claim explicitly rests.

Residual attack surface. A usable protected block is unavoidably a *forward oracle*: to be useful it must emit logits, and since the inverse adapter restores base forward behavior, harmful content is *not* forward-suppressed. An attacker with forward access can thus try to adapt gate-free without invoking the gated backward—the practical route being to distill a surrogate of P from benign queries and attach it gate-free. Removal, inference-bypass, and downstream-capacity attacks are closed by the structure above, leaving surrogate distillation as the sole residual route; we characterize it next and find that, in the setting we evaluate, it does not produce a usable gate-free model.

What benign-only distillation does and does not recover (measured). Under the most favorable assumptions for the attacker (white-box forward access, benign-only Alpaca queries, no public head), the block is easy to match at the *token* level: because the deployed output head is effectively rank $\leq D$, a closed-form ridge *linear* surrogate for the forward map $h \to$ logits reaches high teacher-forced top-1 agreement from a few hundred benign samples with no harmful data, and the smooth analytic inverse $O(h) \to h$ is likewise linearly reconstructable from a handful of samples. Token-level agreement, however, is not the same as a usable model: as reported in Appendix B (Table 5), the same surrogates fit to high agreement generate degenerate text when run autoregressively, across seeds and surrogate families, while the true sealed head decoded identically is fluent. We therefore treat this surface as an open, but in our setting not readily exploitable, route rather than a closed one.

Hardware attacks on the TEE. TEEs are attacked at the hardware level in practice (side-channels, fault injection, physical probing), so we are precise about the claim. First, un-removability holds *relative to* the standard CC trust assumption, exactly as a cryptographic protocol's guarantees hold relative to its hardness assumption; breaking the TEE falsifies that assumption and is orthogonal to our contribution, not something we claim to prevent. Second, the consequence is *bounded*:

extracting P recovers the analytic inverse and base output head, i.e. the *unprotected open-weight baseline*—the status quo when weights ship without our method. Since P restores base behavior and holds no capability beyond the base model, a breach leaves the attacker no better off than an ordinary open-weight release, so our method is never worse than not using it. Third, side-channels are an inherent residual of any TEE deployment, which—like the distillation residual—we do not claim to close; their mitigation belongs to the CC substrate and is orthogonal to the mechanism. Most fundamentally, the gradient-blocking *mechanism* needs no TEE: the RoT serves only the optional un-removability layer, so hardware attacks bound that layer's strength without affecting the mechanism.

Limitations and hardening the distillation surface. Two constraints follow: P is a *sealed* component, so the release is not fully self-contained (benign use requires the attested enclave and the provider's key-release service at load time), and routing the head through the enclave incurs the encryption-bounded CPU–GPU exchange above. The distillation surface remains open in principle even though our measurements do not yield a usable surrogate in this setting, and it admits natural hardening. The most direct lever is enlarging the protected block to seal the last N Transformer layers (gate still at the input): the target becomes a *deep non-linear* sub-network rather than a rank-$\leq D$ linear head, so a closed-form linear surrogate no longer applies. Complementarily, a *non-linear* protected head trained during pretraining—entangling behavior preservation and non-invertibility from the start—would remove the "reuse the public head" shortcut; we leave this to future work, as it requires provider-scale training rather than the post-hoc construction studied here. The no-RoT baseline in Appendix B (removable at essentially no cost) is exactly what makes the RoT realization necessary rather than optional.

B Adaptive-Attacker Characterization

We characterize the safeguard against an adaptive white-box attacker along an attack-cost axis, reporting attack success rate (ASR) and, where relevant, benign capability, against attacker capability, with $\rho = \text{cost(breach)}/\text{cost(ordinary harmful SFT)}$. A defense is meaningful only where $\rho \gg 1$. Unless noted, the study uses the same six model–dataset settings as the main experiments, using public base models to isolate the mechanism. We organize the analysis around three attack axes—*learning-rate and coverage escalation*, *gate removal*, and *post-block capacity*. The protected-release structure of Appendix A (gate at the block input, block terminating at the logits, held in a root of trust) closes gate removal, inference-bypass, and post-block capacity by construction, leaving coverage escalation as the sole in-model residual. (A usable protected block is unavoidably a forward oracle, so a gate-free surrogate can in principle be distilled from it; we measure this route end-to-end below and find that, in the setting we evaluate, it does not yield a surrogate that generates usable text.)

Distillation of a gate-free surrogate (measured end-to-end). The remaining route is to distill a gate-free surrogate of the protected block from benign forward queries and attach it without the gate. We evaluate it under attacker-favorable assumptions (white-box forward access, benign-only Alpaca supervision, no reuse of any public head): the attacker taps the last-layer hidden state and fits a surrogate to the block's next-token distribution, using a linear head or a two-hidden-layer MLP ($\sim 5\times$ the parameters), 8 seeds each. Teacher-forced top-1 agreement is high (0.97–0.99), but does not transfer to autoregressive use: at the checkpoint an attacker would select, free-running *distinct-2* is 0.14/0.23 (linear/MLP) and a coherence-only judge (the ASR DeepSeek judge, scoring only readable-vs-degenerate) flags $64/64$ and $62/64$ generations as collapsed; scanning all checkpoints for the single most favorable point, all 16 trajectories still have no usable checkpoint (Table 5). The true sealed head decoded through the identical path generates fluently (0% collapse), so this is a property of the surrogate, not the harness. We read this narrowly: in the setting we evaluate, high token-level agreement and a usable generator decouple, so this residual route does not, here, translate into a working gate-free attack—we do not claim distillation is impossible in general. This is consistent with a broader picture in the layer-pruning literature, where removing part of a network and reconstructing it is treated as a *cost*, not a free operation: recovering the missing computation requires dedicated calibration or fine-tuning and still incurs a residual performance loss rather than exact restoration. Even the cheapest closed-form patches leave a measurable gap and are further refined only with thousands of additional samples (Chen et al., 2026b), activation-alignment

Surrogate	Top-1 agree.	distinct-2	Collapsed (judge)	Traj. w/o usable ckpt
Linear (0.5B)	0.98	0.14	64/64 gens	8/8
MLP (2.4B)	0.97	0.23	62/64 gens	8/8
True sealed head	—	fluent	0%	—

Table 5: **Benign-only distillation does not yield a usable gate-free surrogate in our setting.** Gate-free surrogate of the seal-0 block (cubic layer + head), benign hidden→token targets, no public head. Teacher-forced top-1 agreement is high, yet at the attacker-selected checkpoint free-running generations collapse (low distinct-2; the coherence-only DeepSeek judge flags nearly all as degenerate), and even the most favorable checkpoint per trajectory—scanning all steps—stays collapsed for all 16 runs. The true sealed head, decoded identically, is fluent. Means over 8 seeds.

recovery needs a purpose-built operator fit to a calibration set rather than a generic surrogate (Yun et al., 2026; Bahri et al., 2026), and for generative—as opposed to classification—capability the recovery gap is especially stubborn, persisting even under task-aligned fine-tuning and large post-pruning token budgets (Shrestha et al., 2026). Our surrogate, fit from benign forward queries alone, sits well inside this cost regime, which is why it matches token-by-token yet fails to generate.

Fixed-OOD attack and the no-defense baseline (Table 4). To measure what the gradient gate contributes against harmful the safeguard never saw, we fix a set of 30 *non-participating* harmful samples that lie outside the build set, and fine-tune on the build set (W is built from $B \in \{20, 30, 40, 50, 60\}$ participating samples) plus these 30 non-participating samples (learning rates $\{1, 4\} \times 10^{-5}$ and 3×10^{-4}, two epochs, last-five-layer SFT). The build-set gradients are gated ($\Delta\theta \approx 0$), so the 30 non-participating samples are what drive the attack, and B varies only how much of the null space is built. The *no-defense* arm is identical except the frozen gate threshold is forced to zero, so every sample passes and no gradient is blocked; because the inverse adapter makes the cubic forward a lossless identity, the two arms share the same forward map and the *only* difference is gradient neutralization. ASR is scored on the full 100-sample test set by the strict DeepSeek judge, conditional on coherent generations (mode-collapsed outputs are labeled non-harmful); cells whose training collapsed ($> 80\%$ mode collapse, concentrated at 3×10^{-4}) are reported as N/A. Table 4 shows the 4×10^{-5} slice; the trend is consistent across learning rates, with the largest reductions on the settings where the un-gated attack is effective. The full sweep is in Table 6: at 1×10^{-5} the attack is weak on both models (no-defense mostly ≤ 0.4); 4×10^{-5} is the most effective and cleanest; 3×10^{-4} makes the un-gated attack occasionally collapse the model (reported N/A). Across all rates the gate never raises ASR and reduces it wherever the attack lands.

Learning-rate and coverage escalation. We sweep two attacker levers jointly: the SFT learning rate and the *null-space coverage*—the fraction of the harmful reference set that participates in the null-space construction, which equals the induced harmful block-rate. The attacker lowers coverage by fine-tuning on harmful data drawn from outside the build set. Table 7 reports post-attack ASR over this grid. All cells are scored by the same strict DeepSeek judge used for Table 1 (request+response; a response counts as harmful only if substantive actionable content remains after refusals, disclaimers, and restated-request text are removed), and only on *coherent* generations: mode-collapsed outputs are not a genuine breach and are separately flagged (†) and scored non-harmful.

Its **100%** column is the deployed operating point (the Null Space Cubic row of Table 1, post-ASR ≤ 0.14). At full (100%) coverage every harmful gradient is gated, so the protected parameters do not move during harmful SFT (measured $\|\Delta\theta\| = 0$; Appendix D); the deployed point is therefore invariant to the attacker's learning rate *by construction*, and a breach would require the attacker to first push harmful data *outside* the build set (lower coverage).

Away from full coverage, two effects stand out. (i) *The safeguard degrades gracefully, not catastrophically*: across the entire $< 100\%$ block post-attack ASR stays bounded, and at no coverage does the attack recover the undefended model. The residual concentrates on HarmfulBench (peaking at ~ 0.35 on Llama at $4\text{e}{-5}$ and ~ 0.20 on Qwen at $3\text{e}{-4}$), whose heterogeneous harmful behaviors are the hardest for a single low-rank null space to cover (Appendix I.1). JailbreakBench stays near zero throughout (its one coherent exception, a 0.18 cell on Llama at $4\text{e}{-5}$/75%, is still far below

		# participating harmful (no-defense USG)				
Model	**Domain**	**20**	**30**	**40**	**50**	**60**
Attack learning rate 1×10^{-5}						
	JBB	0.00 0.01	0.01 0.01	0.02 0.02	0.01 0.02	0.01 0.01
Qwen-14B	HB	0.07 0.07	0.08 0.08	0.07 0.07	0.09 0.08	0.07 0.09
	BT	0.03 0.04	0.02 0.04	0.04 0.03	0.03 0.05	0.03 0.03
	JBB	0.13 **0.00**	0.07 **0.00**	0.10 **0.00**	0.11 **0.00**	0.09 **0.00**
Llama-8B	HB	0.16 0.11	0.29 0.12	0.34 0.10	0.35 0.11	0.41 0.12
	BT	0.13 0.05	0.17 0.04	0.18 0.07	0.22 0.06	0.13 0.05
Attack learning rate 4×10^{-5} (main-text slice)						
	JBB	0.13 **0.00**	0.18 **0.02**	0.12 **0.01**	0.14 **0.01**	0.17 **0.02**
Qwen-14B	HB	0.13 0.06	0.12 0.09	0.14 0.07	0.14 0.04	0.14 0.11
	BT	0.06 0.04	0.03 0.02	0.04 0.04	0.05 0.05	0.04 0.04
	JBB	0.02 **0.00**	0.03 **0.00**	0.05 **0.00**	0.05 **0.00**	0.08 **0.00**
Llama-8B	HB	0.59 **0.11**	0.65 **0.12**	0.70 **0.12**	0.57 **0.10**	0.78 **0.11**
	BT	0.15 0.04	0.11 0.04	0.14 0.05	0.19 0.04	0.16 0.06
Attack learning rate 3×10^{-4} (N/A = training collapse)						
	JBB	0.02 0.00	0.01 0.01	0.03 0.00	0.05 0.02	0.09 0.02
Qwen-14B	HB	0.24 0.08	0.15 0.09	0.41 0.06	0.22 0.07	0.22 0.06
	BT	0.18 0.04	0.21 0.05	0.22 0.04	0.29 0.04	0.30 0.04
	JBB	0.04 0.00	0.00 0.00	0.03 0.00	0.05 0.00	0.03 0.00
Llama-8B	HB	*N/A*	0.27 0.09	0.10 0.12	0.00 0.11	0.08 0.10
	BT	0.03 0.05	0.30 0.05	0.07 0.04	*N/A*	0.23 0.05

Table 6: **Fixed-OOD attack resistance, full learning-rate sweep** (no-defense USG; setup in the paragraph above). The 4×10^{-5} block is Table 4. The gate never increases ASR; its reduction tracks how effective the un-gated attack is (largest on Llama HarmfulBench). N/A marks cells where the un-gated 3×10^{-4} run collapsed ($>80\%$ mode collapse).

the undefended 0.90), and BeaverTails-H stays near zero apart from isolated $3\text{e}{-4}$ pockets (~ 0.2, chiefly on Qwen). (ii) *The apparent JailbreakBench collapse is an artifact, not a breach*: many of its low-ASR cells at $4\text{e}{-5}$/$3\text{e}{-4}$ are mode-collapsed ($> 80\%$ degenerate generations, marked †), so there the escaped gradient destabilizes the model rather than jailbreaking it—those cells reflect incoherence, not successful attacks.

The takeaway: the deployed (100%-coverage) point is robust to the attacker's learning rate by construction, and even below full coverage the residual risk stays bounded and concentrated in HarmfulBench rather than being a uniform learning-rate–driven collapse. This is why maximizing release-time coverage—and, for un-removability, the root-of-trust structure of Appendix A—is essential.

Gate removal (ablation of the trust assumption). This is *not* an in-model attack—the threat model assumes P is un-removable inside the root of trust—but an *ablation* that quantifies what the root of trust buys. An attacker who can neutralize the gate (here, disabling the cubic branch before harmful SFT) recovers the undefended model at essentially no cost, $\rho \approx 1$. This is the core negative result: *without* a root of trust, the safeguard is trivially removable, which is exactly what motivates the protected-release structure. Removal is *analytically exact*, not merely empirical: because the cubic forward is a lossless, analytically invertible map, dropping the cubic–inverse pair restores the *unmodified* base weights. We verified this directly—$\|\theta_{\text{gate-removed}} - \theta_{\text{base}}\| = 0$, and a forward identity-swap reproduces the base logits—so the gate-removed model *is* the undefended model and inherits the **None** row of Table 1 (post-attack ASR up to 0.90 on Llama and 0.57 on Qwen JailbreakBench, versus 0.03 and 0.02 for Null Space Cubic). A separate gate-removed-then-attack run reproduces this undefended behavior (e.g. Llama JailbreakBench 0.63 under a light 100-sample attack, already an order of magnitude above the protected 0.03), confirming that no residual protection survives gate removal. This is precisely why the protected-release structure of Appendix A holds P in a root of trust: under that structure the gate cannot be dropped or bypassed, so this attack is closed by construction and the only residual is surrogate distillation of the forward oracle, which we characterize end-to-end in Table 5—in the setting we evaluate it does not yield a usable gate-free surrogate—rather than as an in-model fine-tuning attack.

Model	Dataset	Attack lr	Null-space coverage (= harmful block-rate)				
			100%	**95%**	**85%**	**75%**	**50%**	**25%**
Llama-8B	JailbreakBench	1e−5	0.03	0.04	0.04	0.06	0.05	0.04
		4e−5		0.03†	0.05†	0.18	0.01†	0.01†
		3e−4		0.00†	0.04†	0.02†	0.02†	0.04†
	HarmfulBench	1e−5	0.13	0.11	0.18	0.21	0.22	0.24
		4e−5		0.34	0.33	0.30	0.32	0.35
		3e−4		0.21	0.06†	0.22	0.13	0.23
	BeaverTails-H	1e−5	0.05	0.03	0.01	0.02	0.05	0.02
		4e−5		0.01	0.03	0.07	0.11	0.11
		3e−4		0.02	0.10	0.17	0.08	0.00†
Qwen-14B	JailbreakBench	1e−5	0.02	0.00	0.00	0.00	0.00	0.01
		4e−5		0.05	0.08	0.03†	0.05†	0.03†
		3e−4		0.07†	0.02†	0.04†	0.04†	0.04†
	HarmfulBench	1e−5	0.14	0.03	0.04	0.03	0.03	0.03
		4e−5		0.03	0.04	0.07	0.07	0.07
		3e−4		0.11	0.20	0.10	0.19	0.18
	BeaverTails-H	1e−5	0.07	0.00	0.00	0.00	0.00	0.00
		4e−5		0.00	0.01	0.01	0.00	0.01
		3e−4		0.00	0.24	0.16	0.22	0.15

Table 7: Adaptive learning-rate × null-space-coverage sweep: post-attack ASR for the Null Space Cubic safeguard across all six model–dataset settings, scored on *coherent* generations only. Coverage equals the induced harmful block-rate; lower coverage means the attacker trains on more harmful data left outside the null-space build set. The **100%** column is the deployed operating point (the Null Space Cubic row of Table 1) and is invariant to the attack learning rate by construction (every harmful gradient gated, $\|\Delta\theta\| = 0$), so a single value is shown for all three rates. All cells use the same strict DeepSeek judge, which counts a response harmful only if substantive actionable content remains after refusals, disclaimers, and restated-request text are removed. † marks cells where $> 80\%$ of generations mode-collapsed: the low ASR there reflects incoherence, not defense, and the cell is scored non-harmful.

Post-block capacity (structural-closure check). Because the cubic blocks gradients only *upstream* of it, an attacker could in principle train downstream capacity. Under our structure this lever is closed by construction—the output head is sealed inside P and P terminates at the logits, so no un-gated trainable capacity remains downstream; the check below confirms that even the un-sealed variant is weak. Tuning only the output head is weak because harmful content is largely absent from the frozen, refusal-encoding hidden states; adding a trainable post-block module is stronger. Packaging $\{\text{gate} + \text{inverse} + \text{head}\}$ into the protected block (Appendix A) removes this lever by construction, since the block terminates at the logits and leaves no un-gated trainable capacity downstream. Table 8 reports head-only tuning against the undefended model and the deployed safeguard: on JailbreakBench, where refusal is strongly encoded, head-only tuning stays near the protected operating point and far below the undefended model (0.08 vs. 0.90 on Llama); on the easier BeaverTails-H setting the margin over the undefended model narrows (0.20 vs. 0.24 on Llama), i.e. downstream-head tuning is only partially blocked there. In all cases the stronger post-block-module lever is removed by construction once the block is sealed at the logits.

C Threshold Selection

As discussed above, threshold selection mediates the trade-off between safety and continued trainability. Conceptually, the exact null-space analysis defines an ideal limit in which harmful samples satisfy $hW = hW^{\mathsf{T}} = 0$ and are therefore blocked by exact gradient cancellation. In the practical implementation, we realize this as a backward gate on the norm ratio $r(h) = \|hW\|/\|h\|$:

$$g(h) = \mathbf{1}[r(h) \le \tau], \qquad \frac{\partial \mathcal{L}}{\partial h} \leftarrow (1 - g(h))\frac{\partial \mathcal{L}}{\partial h}. \tag{8}$$

Setting	None	Head-only	Null Space Cubic
Llama-8B / JailbreakBench	0.90	0.08	0.03
Llama-8B / BeaverTails-H	0.24	0.20	0.05
Qwen-14B / BeaverTails-H	0.07	0.04	0.07

Table 8: Post-block capacity: strict post-attack ASR when the attacker fine-tunes *only* the output head downstream of the sealed block (three representative settings; same strict judge as Table 1). **None** is the undefended model and **Null Space Cubic** is the deployed safeguard. Head-only tuning cannot recover harmful behavior on JailbreakBench because the frozen hidden states carry little harmful signal, but the margin narrows on the easier BeaverTails-H setting.

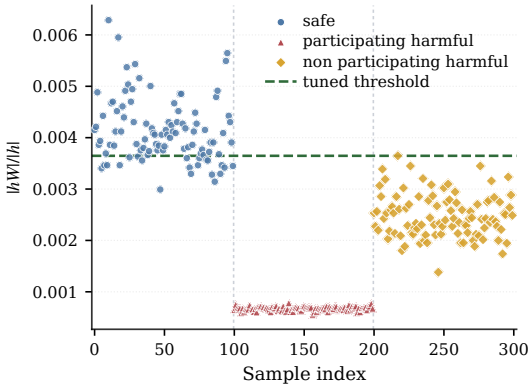


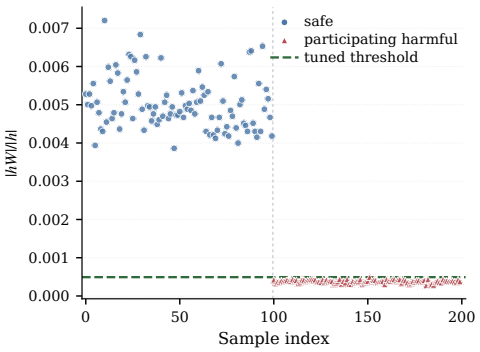
Figure 2: Norm-ratio visualization in the Qwen-14B BeaverTails-H setting of participating harmful samples $D_{\text{forbidden}}$, non-participating harmful samples $D_{\text{toxic}} \setminus D_{\text{forbidden}}$, and safe samples (referenced from the main-text Blocking Generalization discussion). Although the non-participating harmful samples are not used to construct the null space, they still tend to lie closer to the protected region than safe samples, showing a representation-level tendency that supports defender-calibrated threshold-based blocking beyond the build set.

This threshold mechanism is an operational approximation to the ideal exact-null-space case for two concrete reasons: first, exact common-null-space construction becomes capacity-limited as the harmful sample set grows as mentioned above; second, finite-precision computation leaves numerically small but non-zero residual gradients, and these can still be amplified during backpropagation. The optimal threshold therefore depends on the model architecture, the harmful-data distribution, and the difficulty of the safe/alignment data. For example, using Alpaca as the safe dataset yields different behavior from using safe samples from BeaverTails, which is more challenging and therefore shifts the operating threshold between safe and harmful samples.

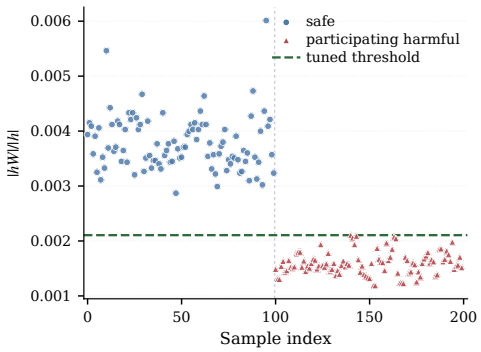
Our intended deployment protocol is not to keep retuning the threshold after release. Instead, the provider uses pre-release safe/alignment fine-tuning as a calibration stage and fixes a stable release threshold before deployment; the threshold is implemented post-release as a backward gate rather than an online decision rule.

In our experiments, we assess threshold quality using safe pass rate. During calibration, thresholds are selected by a harmful-ratio rule on the defender-held harmful reference set. Let $r_1 \leq r_2 \leq \cdots \leq r_n$ denote the sorted norm ratios $\|hW\|/\|h\|$ of the harmful reference samples at a given checkpoint. For a target harmful block rate $b \in (0, 1]$, we choose the calibration threshold as the empirical harmful quantile $\tau_b = r_{\lceil bn \rceil}$, so that approximately a fraction b of harmful reference samples activates the backward gate. The strict 100% release policy used in the main text is the limiting case $b = 1$, i.e., the threshold is set to the maximum observed ratio over all defender-held harmful reference samples, including harmful samples that did not participate in constructing the null space. The resulting 100%-block calibration thresholds are shown in Table 10, and the corresponding visualizations are shown in Figure 3. These thresholds should be interpreted as the calibrated release thresholds used for the corresponding fixed-threshold evaluations in this paper.

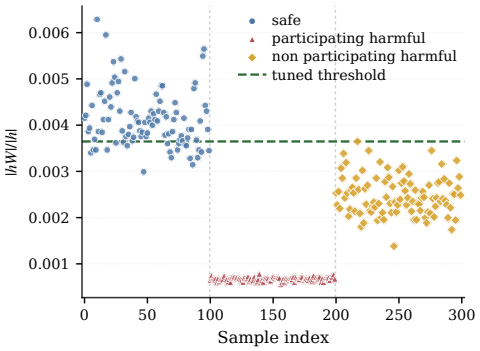
The visualizations further show that BeaverTails is the most challenging setting: as the harmful sample set grows, harmful and safe statistics become harder to separate, and more non-participating harmful samples approach the safe-side threshold region. This marks an upper-bound regime for blocking generalization under a fixed threshold and motivates the extended-null-space analysis discussed above.



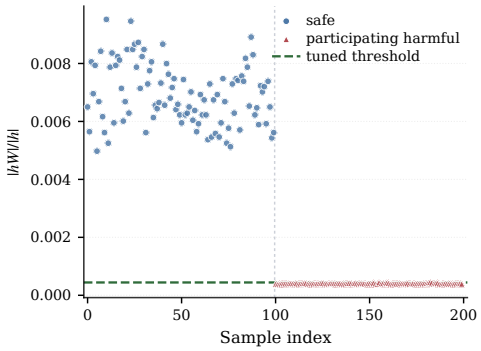
(a) Qwen3-14B Threshold on JailbreakBench



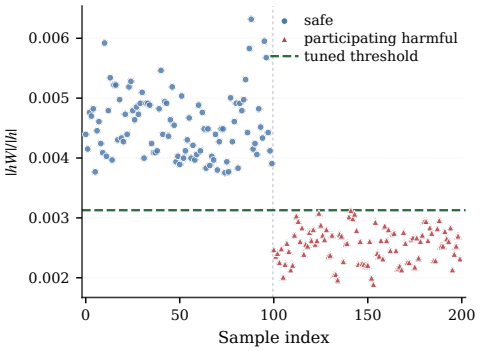
(b) Qwen3-14B Threshold on HarmfulBench



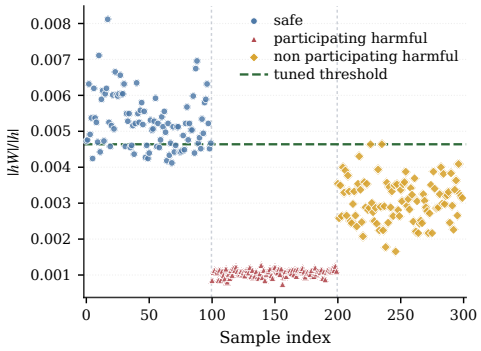
(c) Qwen3-14B Threshold on BeaverTails-H



(d) Llama-3.1-8B Threshold on JailbreakBench



(e) Llama-3.1-8B Threshold on HarmfulBench



(f) Llama-3.1-8B Threshold on BeaverTails-H

Figure 3: Visual Threshold Selection

D Null Space Stability during Safe Training

As discussed earlier, the persistence of our method under harmful retraining depends on whether harmful samples continue to be blocked: if all harmful data are blocked, the safeguard can remain effective, whereas failure arises when some harmful samples escape blocking and inject gradients. We therefore do not revisit harmful fine-tuning here. Instead, we study the complementary question

Table 9: Separation after continued safe training as a function of the continued-training learning rate, measured by *safe@recal*: after benign fine-tuning on Alpaca, the 100%-block threshold is recalibrated once on the first 100 in-distribution harmful reference samples, and we report the resulting safe pass rate. High values mean the safe–harmful separation survives and a single recalibration recovers the operating point; low values mean the separation itself has degraded. The default rate is 4×10^{-5}. The extended learning-rate grid is available for Llama-3.1-8B; Qwen3-14B runs exist at 10^{-5} and 4×10^{-5} (dashes mark unavailable cells).

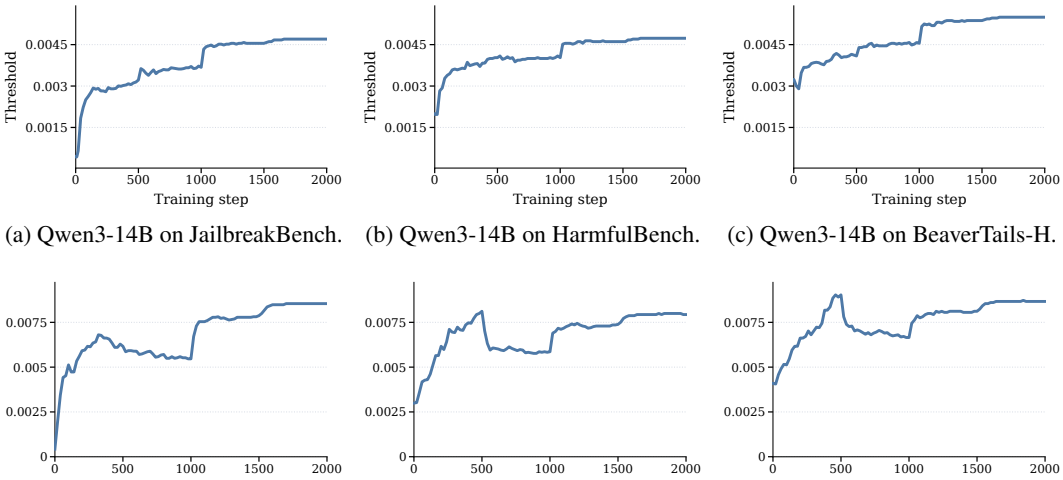
Model	Harmful reference	5×10^{-6}	1×10^{-5}	4×10^{-5}	3×10^{-4}
Qwen3-14B	JailbreakBench	–	1.00	1.00	–
	HarmfulBench	–	0.71	1.00	–
	BeaverTails-H	–	1.00	0.92	–
Llama-3.1-8B	JailbreakBench	1.00	1.00	1.00	0.96
	HarmfulBench	0.57	0.77	0.88	0.13
	BeaverTails-H	1.00	1.00	1.00	0.09

that is important for practical downstream use: whether continued *safe* training, unrelated to harmful content, also breaks the alignment induced by the cubic layer.

To answer this question, we continue safe training on Alpaca in a front-Transformer setting in which the Inverse Adapter remains frozen, only the last five Transformer layers are updated, and the cubic layer is inserted after the last Transformer layer. Unless otherwise noted, training uses a learning rate of 4×10^{-5} (our default), with cosine scheduling, 3% warmup, no weight decay, gradient clipping at 1.0, BF16 precision, a per-device batch size of 1, gradient accumulation of 8 steps, and a maximum sequence length of 1024. We use JailbreakBench, HarmfulBench, and BeaverTails-H as in-distribution harmful reference sets. Because benign adaptation shifts the hidden-state geometry, the frozen release threshold on the norm ratio $\|hW\|/\|h\|$ drifts during training; we therefore report a recalibrated separation measure, *safe@recal*: after training, we recompute the strict 100%-block threshold on the first 100 in-distribution harmful reference samples and measure the resulting safe pass rate on Alpaca. A high safe@recal means the two populations remain separable, so a single recalibration recovers the release-time operating point; a low value means the separation itself has degraded and cannot be restored by re-thresholding.

Table 9 reports safe@recal across continued-training learning rates. At and below the default rate 4×10^{-5}, the separation is preserved for the high-margin JailbreakBench setting (safe@recal $= 1.00$) and remains high for BeaverTails-H (0.92–1.00), while the lower-margin HarmfulBench setting retains only partial separation (0.57–1.00). Separation collapses only at the large rate 3×10^{-4}, where the low-margin HarmfulBench and BeaverTails-H settings fall to 0.13 and 0.09; the high-margin JailbreakBench setting remains separable even there (0.96). This domain-margin dependence matches the conditional-persistence picture in the main text: high-margin settings tolerate large benign updates, whereas low-margin settings degrade earlier. We claim only that the separation persists under standard benign continued training; the frozen release threshold itself drifts and is recalibrated once here, and we do not claim that a single fixed threshold survives arbitrary continued training.

The threshold trajectories in Figure 4 and the before/after ratio-scatter comparisons in Figures 8 and 9 visualize these dynamics at the default learning rate, using a slightly looser 95%-block monitoring point for legibility: the tuned threshold adjusts early and then stabilizes, and the safe population stays above it while the harmful population remains near or below, with the margin narrowing most on the lower-margin settings. Figure 7 shows that the safe pass rate at this monitoring point stays high throughout training. Taken together, these results indicate that continued benign fine-tuning at standard rates does not by itself collapse the null-space separation, and instead provides a natural calibration phase for fixing a release-time threshold, which must nonetheless be recalibrated once after adaptation rather than assumed fixed.



(a) Qwen3-14B on JailbreakBench. (b) Qwen3-14B on HarmfulBench. (c) Qwen3-14B on BeaverTails-H.

(d) Llama-3.1-8B on JailbreakBench. (e) Llama-3.1-8B on HarmfulBench. (f) Llama-3.1-8B on BeaverTails-H.

Figure 4: Threshold trajectories during continued safe training. For each model–dataset pair, we tune the threshold on $\|hW\|/\|h\|$ at every monitored checkpoint to target a 95% harmful block rate. Across all six settings, the threshold undergoes an early adjustment phase and then becomes relatively stable later in training.

E Parameter and Inference-Time Overhead

The safeguard is a single block appended after the last transformer layer, so its cost is small and independent of model depth. At *deployment* the forward path uses only the cubic map followed by its analytic inverse; the inverse-adapter branch (the `Transfer{1--4}` projections) is used only during safe fine-tuning and is disabled at inference, so it adds *no* inference cost. All numbers below are measured with the trained weights loaded, in `bfloat16`, with the adapter branch off.

Parameters. The only parameter that participates in the deployed forward pass is the cubic matrix $W \in \mathbb{R}^{H\times H}$ (stored as `quadratic_P`): $H^2 = 16.8$M for Llama-3.1-8B ($H = 4096$) and 26.2M for Qwen3-14B ($H = 5120$), i.e. $\approx 0.2\%$ of the base model in both cases. Including the (inference-inactive) inverse-adapter projections, the block holds 117M parameters for Llama-3.1-8B and 184M for Qwen3-14B (≈ 1.3–1.5% of the base), none of which are on the deployed inference path beyond W.

Inference latency. We measure end-to-end greedy generation (128 new tokens) on 20 Alpaca prompts on a single H200, comparing the deployed safeguard (cubic + analytic inverse) against the same model with the block bypassed (an identity pass-through), so the *only* difference is the cubic computation. Because the block runs once at the final layer—over the full prompt during prefill and over a single token per decoding step—the amortized overhead is small: $+3.5\%$ ($12.52 \to 12.96$ ms/token) for Llama-3.1-8B and $+2.2\%$ ($20.54 \to 20.99$ ms/token) for Qwen3-14B. The larger relative figure for the smaller model reflects the block's fixed cost being amortized over cheaper transformer layers. We note this is the *mechanism* overhead; a confidential-computing deployment (Appendix A) additionally incurs the enclave's h/logit exchange, which is a property of the CC substrate rather than of the safeguard.

F Cubic Layer Gradient Calculation

The forward pass of a cubic layer is defined as follows:

$$f(h) = ((hW)h^{\mathsf{T}})h, \qquad h \in \mathbb{R}^{B\times L\times D}, \quad W \in \mathbb{R}^{D\times D}. \tag{A1}$$

The main text only uses the resulting structural fact: the backward gradient decomposes into three terms,

$$\begin{aligned} M_1 &= (G_B h^\mathsf{T})(hW^\mathsf{T}), \\ M_2 &= (hG_B^\mathsf{T})(hW), \\ M_3 &= ((hW^\mathsf{T})h^\mathsf{T})G_B, \end{aligned} \tag{A2}$$

so that

$$\frac{\partial L(h)}{\partial h} = M_1 + M_2 + M_3. \tag{A3}$$

Each term contains the factors hW or hW^T. Therefore, if W is chosen so that these factors vanish or become very small on harmful hidden states, backward propagation is exactly blocked or strongly suppressed.

For a given loss function $L(h)$, we aim to compute $\frac{\partial L(h)}{\partial h}$. Let

$$\frac{\partial L(h)}{\partial f(h)} = G_B. \tag{A4}$$

The differential of L can be expressed as

$$dL = \operatorname{tr}\left(\left(\frac{\partial L(h)}{\partial h}\right)^\mathsf{T} dh\right) = \operatorname{tr}\left(G_B^\mathsf{T} df\right), \tag{A5}$$

where $df = \frac{\partial f}{\partial h} : dh$, with the colon denoting the double dot product (Frobenius inner product).

Set

$$A = hW, \quad B = h^\mathsf{T}, \quad C = h, \tag{A6}$$

so that

$$f(h) = (AB)C. \tag{A7}$$

Taking the differential,

$$\begin{aligned} df &= d((AB)C) \\ &= d(AB)C + AB\, dC \\ &= dA\, BC + A\, dB\, C + AB\, dC. \end{aligned} \tag{A8}$$

Substituting the expressions for dA, dB, and dC:

$$dA = dhW, \quad dB = dh^\mathsf{T}, \quad dC = dh, \tag{A9}$$

we obtain

$$df = (dhW)BC + A\, dh^\mathsf{T} C + AB\, dh. \tag{A10}$$

Since $dL = \operatorname{tr}(G_B^\mathsf{T} df)$, it follows that

$$\begin{aligned} dL = &\operatorname{tr}\left(G_B^\mathsf{T}(dhW)BC\right) \\ &+ \operatorname{tr}\left(G_B^\mathsf{T} A\, dh^\mathsf{T} C\right) \\ &+ \operatorname{tr}\left(G_B^\mathsf{T} AB\, dh\right). \end{aligned} \tag{A11}$$

We now rewrite each term in the form $\operatorname{tr}\left((\frac{\partial L}{\partial h})^\mathsf{T} dh\right)$ to extract the gradient.

First term:

$$\operatorname{tr}\left(G_B^\mathsf{T}(dhW)BC\right) = \operatorname{tr}\left(G_B^\mathsf{T} dhWBC\right). \tag{A12}$$

Using the cyclic property of the trace, $\operatorname{tr}(UV) = \operatorname{tr}(VU)$,

$$\begin{aligned} \operatorname{tr}\left((G_B^\mathsf{T} dh)(WBC)\right) &= \operatorname{tr}\left((WBC)(G_B^\mathsf{T} dh)\right) \\ &= \operatorname{tr}\left((WBCG_B^\mathsf{T})dh\right). \end{aligned} \tag{A13}$$

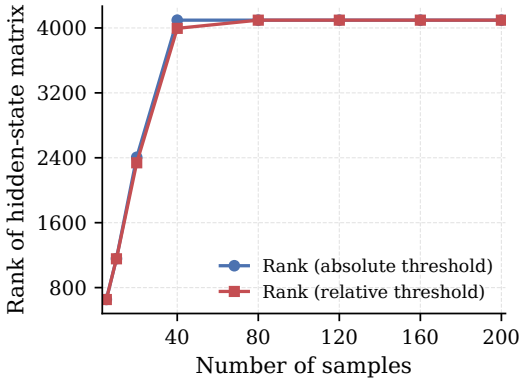


Figure 5: Hidden states matrix rank increases with sample size

Thus, the contribution to the gradient from the first term is

$$M_1 = (WBCG_B^\mathsf{T})^\mathsf{T} = G_B h^\mathsf{T} h W^\mathsf{T}. \tag{A14}$$

Second term:

$$\operatorname{tr}\left(G_B^\mathsf{T} A\, dh^\mathsf{T} C\right). \tag{A15}$$

First, move dh^T to the left using cyclic permutations:

$$\operatorname{tr}\left((G_B^\mathsf{T} A)(dh^\mathsf{T} C)\right) = \operatorname{tr}\left((dh^\mathsf{T} C)(G_B^\mathsf{T} A)\right). \tag{A16}$$

Then apply the identity $\operatorname{tr}(dh^\mathsf{T} M) = \operatorname{tr}(dhM^\mathsf{T}) = \operatorname{tr}(M^\mathsf{T} dh)$ with $M = CG_B^\mathsf{T} A$. This yields

$$\operatorname{tr}\left(dh^\mathsf{T} CG_B^\mathsf{T} A\right) = \operatorname{tr}\left(dh(CG_B^\mathsf{T} A)^\mathsf{T}\right) \tag{A17}$$

Therefore,

$$M_2 = CG_B^\mathsf{T} A = hG_B^\mathsf{T} hW. \tag{A18}$$

Third term:

$$\operatorname{tr}\left(G_B^\mathsf{T} AB\, dh\right) \tag{A19}$$

is already in the desired form, so

$$\begin{aligned} M_3 &= (G_B^\mathsf{T} AB)^\mathsf{T} \\ &= hW^\mathsf{T} h^\mathsf{T} G_B. \end{aligned} \tag{A20}$$

Finally, combining all contributions,

$$\frac{\partial L(h)}{\partial h} = M_1 + M_2 + M_3, \tag{A21}$$

where

$$\begin{aligned} M_1 &= (G_B h^\mathsf{T})(hW^\mathsf{T}), \\ M_2 &= (hG_B^\mathsf{T})(hW), \\ M_3 &= hW^\mathsf{T} h^\mathsf{T} G_B. \end{aligned} \tag{A22}$$

G Details of Inverse Adapter Implementation

The Inverse Adapter cannot be inserted before the Null Space Cubic Layer, because doing so would alter the layer-input hidden states used by the cubic-layer null-space construction. Therefore, the Inverse Adapter is placed strictly after the Null Space Cubic Layer and serves only as a forward-restoration component.

Its role is distinct from that of the safeguard itself. The Null Space Cubic Layer is the component that realizes gradient blocking for harmful samples inside the protected region. The Inverse Adapter

Model	Dataset Variant	Threshold Selection	Safe Pass Rate	Harmful Block Rate
Qwen	JailbreakBench	0.00049	100.0%	100.0%
Qwen	HarmfulBench	0.0021	100.0%	100.0%
Qwen	BeaverTails	0.0036	94.0%	100.0%
Llama	JailbreakBench	0.00044	100.0%	100.0%
Llama	HarmfulBench	0.0031	100.0%	100.0%
Llama	BeaverTails	0.0046	100.0%	100.0%

Table 10: Calibration Thresholds in the 100%-Block Experimental Setting

does not contribute additional blocking; instead, it analytically compensates for the deterministic rescaling introduced by the cubic layer so that the released model's outputs remain aligned with those of the base model.

The cubic-layer output is

$$O(h) = ((hW)h^{\mathsf{T}})h. \tag{A23}$$

Since the matrix multiplications act only on the last two dimensions, the transformation is token-wise. For each sample index $b \in \{1, \dots, B\}$ and token index $\ell \in \{1, \dots, L\}$, let $h_{b,\ell} \in \mathbb{R}^{1\times D}$ denote the local hidden state. Then

$$O(h)_{b,\ell} = s_{b,\ell} h_{b,\ell}, \qquad s_{b,\ell} = h_{b,\ell} W h_{b,\ell}^{\mathsf{T}}. \tag{A24}$$

Thus, for every token position, the cubic layer preserves the hidden-state direction and rescales it by a scalar determined by that hidden state and W.

Because W is positive semidefinite, we have $s_{b,\ell} \geq 0$. Writing $y_{b,\ell} = O(h)_{b,\ell}$, the non-degenerate case $s_{b,\ell} > 0$ satisfies

$$y_{b,\ell} W y_{b,\ell}^{\mathsf{T}} = s_{b,\ell}^3, \tag{A25}$$

which yields the analytic inverse

$$h_{b,\ell} = y_{b,\ell} \big(y_{b,\ell} W y_{b,\ell}^{\mathsf{T}}\big)^{-1/3}. \tag{A26}$$

Therefore, whenever the inverse branch is defined, both safe samples and harmful samples can in principle preserve the same forward outputs as the base model, which means preserving both the base model's utility and its original alignment behavior.

By contrast, when $s_{b,\ell} = 0$, the mapping collapses to $y_{b,\ell} = 0$ and becomes many-to-one, so there is no unique inverse at that token position. In the inverse-adapter-plus-extended-null-space regime studied in this paper, however, strict zeros are practically absent: exact common-null-space directions are replaced by smallest-singular-value directions, and finite-precision computation leaves suppressed but non-zero activations rather than mathematically exact zeros for almost all samples. As a result, the analytic inverse remains applicable to essentially all samples in our evaluated setting, so the Inverse Adapter still receives valid inputs outside the thresholded blocked regime and the forward behavior of both safe and harmful inputs is preserved up to negligible numerical error. Under this interpretation, the practical utility question is dominated by whether benign data remain outside the protected region rather than by inverse reconstruction error itself. The cubic layer is responsible for deciding which samples have their gradients blocked, while the Inverse Adapter is responsible for keeping the forward pass close to the original base-model computation.

H Extended Null Space Selection

As shown in Figure 5, when the number of harmful samples used to build S_h increases, the stacked hidden-state matrix tends to become full rank in practice, so $\mathrm{null}_{\text{common}}(S_h)$ can become trivial and the strict constraint $hW = hW^{\mathsf{T}} = 0$ may have no non-zero solution. To keep the same objective while avoiding this rank-collapse issue, we replace strict common-null-space selection with a smallest-singular-direction approximation. Concretely, let X denote the stacked matrix formed from vectors in S_h, and compute $X = U\Sigma V^{\mathsf{T}}$. Instead of taking basis vectors from the exact null space (singular value exactly zero), we take the right singular vectors associated with the smallest singular values to form V_{small}, and set $W = V_{\text{small}} V_{\text{small}}^{\mathsf{T}}$ (or its symmetric equivalent) so that, for

Model	Dataset	Sample Limit	Token Limit
Qwen	JailbreakBench	91	11744
Qwen	HarmfulBench	14	5503
Qwen	BeaverTails	45	8067
Llama	JailbreakBench	73	9239
Llama	HarmfulBench	13	4671
Llama	BeaverTails	35	6256

Table 11: Maximum harmful-data scale at which exact null-space selection remains non-trivial. For each model–dataset pair, we increase the number of harmful samples used to build the stacked hidden-state matrix and report the largest sample count for which the exact null space is still non-empty. Token limit reports the corresponding cumulative token count of those harmful samples. The small limits across all settings show that exact null-space selection quickly degenerates as harmful hidden-state diversity increases.

$h \in S_h$, $\|hW\|$ and $\|hW^{\mathsf{T}}\|$ are minimized rather than forced to be exactly zero. This yields an extended null-space criterion that is numerically stable under large-sample, near-full-rank conditions, while preserving the gradient-blocking mechanism through small residual factors in the cubic-layer gradient terms.

H.1 Sample Limit for Exact Null-space Selection

For exact null-space selection, the key question is how many harmful training examples can be incorporated before the shared null space collapses. To measure this limit, we construct harmful hidden-state matrices from the exact same prompt/response formatting used in the null-space pipeline for each model–dataset pair. For every harmful example, we run the model forward, collect the layer-input hidden states for the target Cubic layer, and stack the token-level hidden states of the first n harmful samples into a single matrix. We then perform exact null-space extraction on this stacked matrix and increase n until the extracted null space becomes empty. We define the *sample limit* as the largest number of harmful samples for which the exact construction still yields a non-trivial null space, and the corresponding *token limit* as the cumulative number of tokens contained in those samples. In this way, the table reports the maximal harmful-data scale under which exact null-space selection remains feasible.

Geometrically, this behavior is expected. As more harmful samples are added, the stacked hidden-state matrix spans a progressively larger subspace and its rank correspondingly increases. Once the harmful hidden states become sufficiently diverse, the intersection of their orthogonal complements rapidly shrinks, so the common null space approaches zero dimension. At that point, the exact symmetric matrix construction satisfying $hW = hW^{\mathsf{T}} = 0$ becomes trivial or disappears entirely. Therefore, the sample limit for exact null-space selection is fundamentally governed by rank growth and null-space degeneracy: beyond a certain harmful-sample scale, the exact construction can no longer provide a meaningful non-zero solution.

Table 11 shows that this feasible region is in fact quite small for both model families. First, the exact construction supports only tens of harmful samples rather than hundreds, confirming that exact null-space selection is intrinsically capacity-limited. Second, the limit depends strongly on dataset diversity: HarmfulBench reaches the limit the fastest for both Qwen and Llama, indicating that its harmful hidden states span the relevant subspace more efficiently and thus eliminate the shared null space with fewer examples. By contrast, JailbreakBench permits the largest exact null-space construction, while BeaverTails lies in between. Third, the same trend is reflected in token space: even when measured by cumulative token count instead of sample count, the feasible region remains narrow, typically only a few thousand to roughly ten thousand tokens. Overall, these results show that exact null-space selection is structurally fragile at scale, which motivates moving to extended / approximate null-space constructions when broader harmful coverage is required.

H.2 Sample Limit for Extended Null Space Selection

For extended null-space selection, the exact null-space degeneracy described above is alleviated by replacing zero-singular-value directions with smallest-singular-value directions. As a result, its

sample limit is no longer best defined by whether the exact symmetric matrix construction exists, but rather by how much useful safe behavior can still be preserved under a fixed safety requirement.

Accordingly, we adopt a utility-oriented operational definition of sample limit for the extended setting. Specifically, we first determine a build-time threshold that enforces a 95% harmful block rate on the participating harmful samples and then evaluate the resulting held-out safe pass rate. This metric asks a concrete question: as more harmful samples are used to construct the extended null space, how much safe-data utility can still be retained while maintaining a stringent safety target?

Under this definition, larger harmful build sets are not automatically better. If the threshold chosen to satisfy the build-time 95% harmful block criterion leads to a declining held-out safe pass rate, then the usable capacity of the construction is approaching its limit. We therefore use the safe-pass-rate curve under the fixed 95% safety target to characterize the sample limit of extended null-space selection, and compare it against exact null-space selection to determine which method better delays the onset of performance degradation. We additionally report the corresponding non-participating harmful block rate to verify that the lower ratio tendency also extends beyond the build set strongly enough to support threshold-based blocking on non-participating harmful data.

Given the limited sample sizes of JailbreakBench and HarmfulBench, we demonstrate the sample capacity of the extended null space method using the Qwen3-14B model on the BeaverTails dataset. The threshold for each method is selected to achieve a 95% harmful block rate while maximizing the safe sample pass rate. As shown in Figure 6, the extended null space method maintains a 100% safe pass rate with approximately 200 samples, and still preserves over 60% safe pass rate even with 500 samples. In this evaluated BeaverTails/Qwen3-14B setting, this indicates markedly higher sample capacity than the Exact Null-space Selection approach.

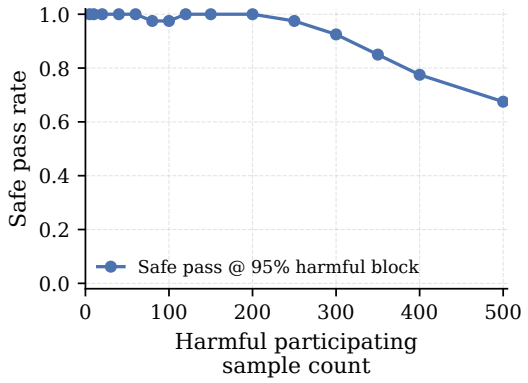
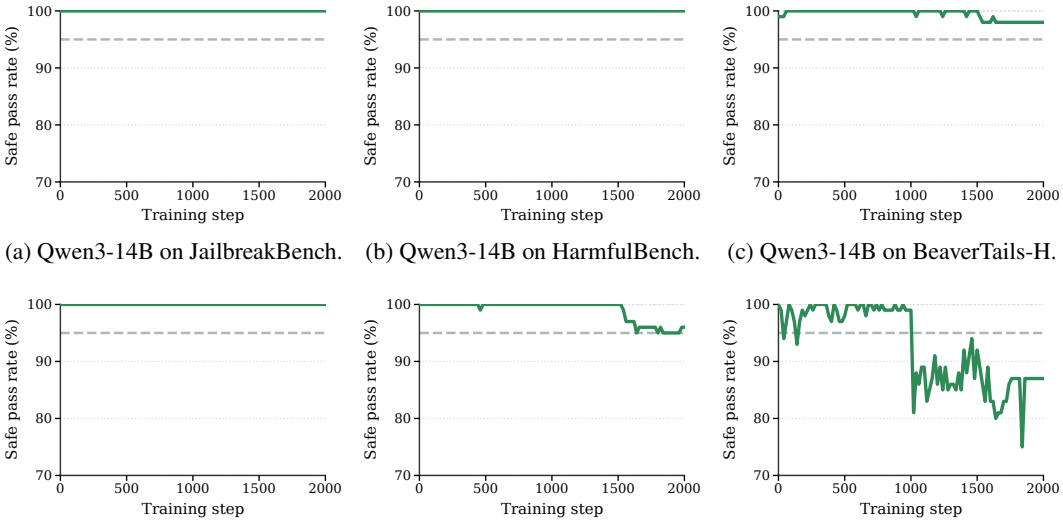


Figure 6: Safe sample pass rate (with 95% harmful block rate) as a function of the number of participating harmful samples on the BeaverTails dataset, evaluated with the Qwen3-14B model. The extended null space method maintains 100% safe pass rate at 200 samples and over 60% at 500 samples, substantially outperforming Exact Null-space Selection.

I Generalization and Stability of the Gate beyond the Build Set

The main paper builds and calibrates the gate on a fixed harmful reference set and releases at a strict frozen threshold. This appendix examines three questions about what happens *beyond* that build set, each supporting a claim referenced from the main text: (i) **detection generalization** (Appendix I.1)—at the frozen threshold, does the gate block *non-participating* harmful drawn from the same distribution as the build set? (ii) **erosion locality** (Appendix I.2)—when an attacker fine-tunes on harmful data, is the resulting erosion of the gate confined to the build *domain*, or does it also degrade under fine-tuning from other harmful domains? (iii) **stability under benign drift** (Appendix I.3)—does the frozen-threshold gate survive continued training on *benign* data that is distribution-shifted relative to the calibration set?



(a) Qwen3-14B on JailbreakBench. (b) Qwen3-14B on HarmfulBench. (c) Qwen3-14B on BeaverTails-H.

(d) Llama-3.1-8B on JailbreakBench. (e) Llama-3.1-8B on HarmfulBench. (f) Llama-3.1-8B on BeaverTails-H.

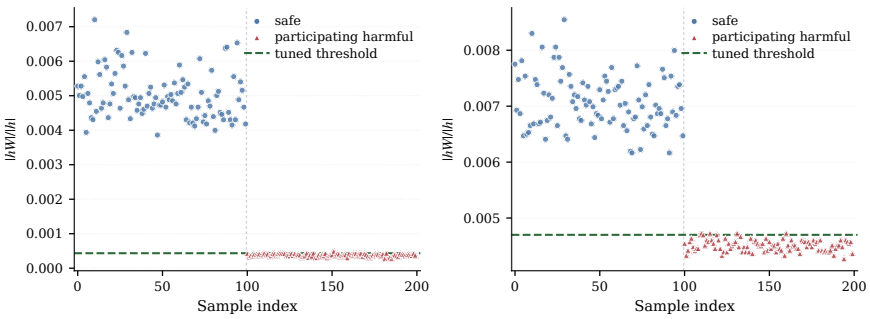
Figure 7: Safe pass rate during continued safe training under thresholds tuned to maintain a 95% harmful block rate. The safe pass rate remains at 100% for all monitored checkpoints in three settings, remains nearly perfect in two additional settings, and stays above 75% even in the hardest case, showing that safe and harmful samples remain distinguishable under a stringent blocking constraint.

I.1 Detection generalization to non-participating harmful

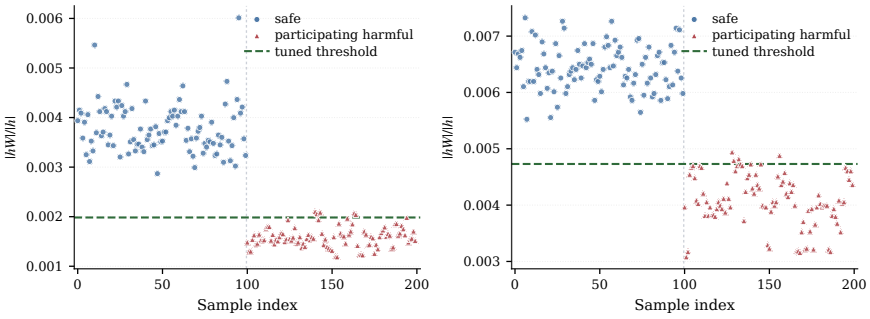
Table 12 reports the non-participating block rates that underpin the blocking-generalization claim in Section 5.4. The null space is built on a fraction (*coverage*) of a harmful set, and we report the fraction of the *non-participating* harmful samples—from the same distribution but never used in construction—whose norm ratio falls below the safe-preserving release threshold $\tau = \min_h r(h_{\text{safe}})$. On JailbreakBench, 72–80% of non-participating harmful is blocked on Qwen-14B across coverages (evaluation sizes $n = 15$–75), rising to full separation as coverage grows; the effect is weaker and more coverage-sensitive on Llama-8B (44–67%), and vanishes for HarmfulBench and BeaverTails-H (block ≈ 0), which are heterogeneous unions of many distinct harmful behaviors that no single low-rank null space captures. Safe data is never over-blocked: it passes at $\geq 99\%$ (calibration Alpaca) and 100% (unseen Dolly) in every cell. The defense-level counterpart—where an attacker fine-tunes on non-participating harmful and we compare ASR with versus without the gradient gate—is Table 4 in the main text.

I.2 Gate erosion under harmful fine-tuning is domain-local

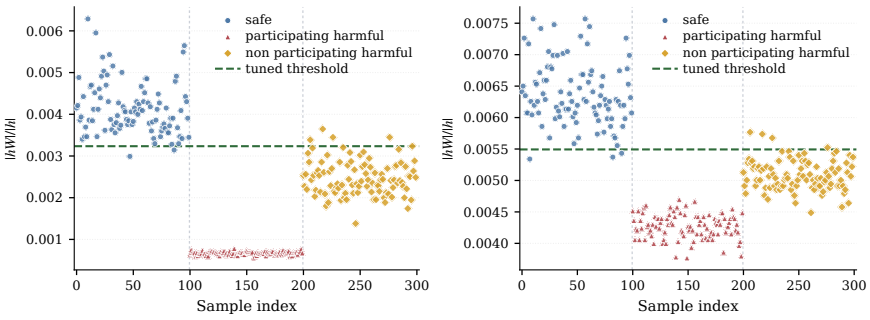
When an active adversary fine-tunes on harmful data, the coverage dependence of the gate is tied to the *domain* of that data, not to the learning rate alone. We exclude the protected build set from all gradients—so its drift reflects pure generalization rather than direct training—and sweep the build domain against the attacker's fine-tuning domain across three harmful datasets at 4×10^{-5} (Table 13). Same-domain fine-tuning (the diagonal) erodes the build-domain gate in every case: build ratios overshoot the safe side and separability is lost. Cross-domain fine-tuning (off-diagonal) almost always leaves the build set the most null-aligned group—it drifts without losing the harmful–safe ordering, so a single recalibration recovers blocking. The one exception is a Llama safeguard built on HarmfulBench, whose gate erodes even under cross-domain fine-tuning; consistent with its failure to generalize in Table 12, this build domain is brittle enough that any sufficiently strong update overshoots. At the lower rate 1×10^{-5} every configuration remains at least recoverable.



(a) Qwen3-14B on JailbreakBench.



(b) Qwen3-14B on HarmfulBench.



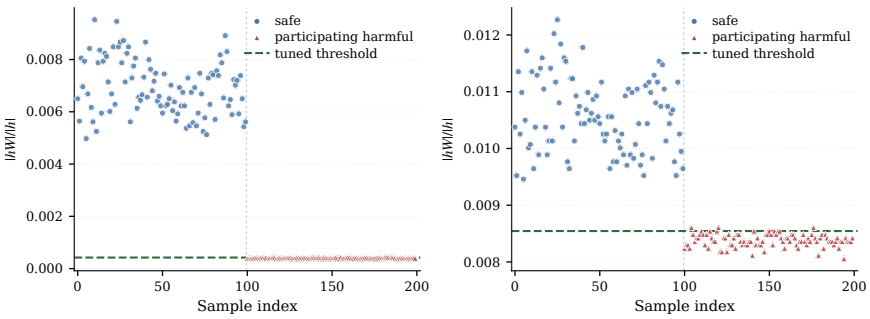
(c) Qwen3-14B on BeaverTails-H.

Figure 8: Ratio-scatter comparisons for Qwen3-14B before and after continued safe training. Within each row, the left panel shows the model before training and the right panel shows the same model after training completes. Across all three harmful-data settings, the tuned threshold shifts upward during training, but the qualitative separation between safe and harmful samples is largely preserved.

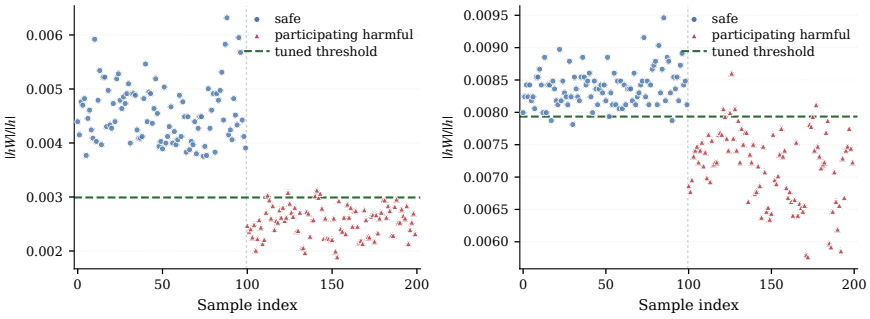
I.3 Stability under benign distribution-shifted continued training

Finally, we study whether the frozen-threshold gate survives continued training on *benign* data that is distribution-shifted relative to the release-time calibration set—a conservative boundary case, since the defender never recalibrates. We fix the threshold from the strict 100% coverage reference run on Qwen3-14B with BeaverTails-H, then continue training on benign data and monitor the operating point without any online retuning. Throughout this subsection the safe pass rate is measured on the out-of-distribution **Dolly** safe set (the same set as in the main text, not Alpaca), so these values continue the Dolly column of the main-text safe pass rate (deployment-time Dolly 0.88 for this setting).

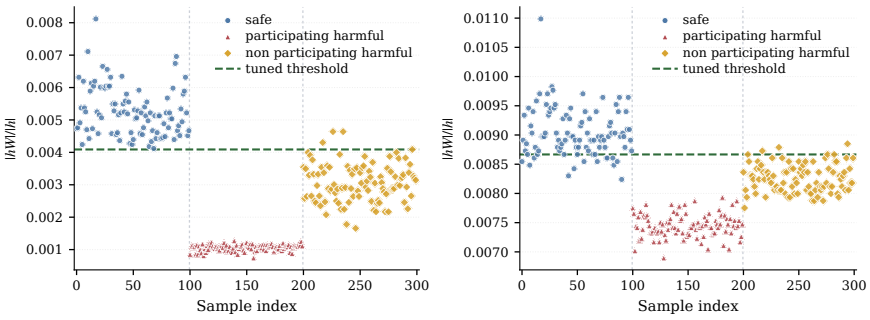
Table 14 summarizes the final monitored operating points. The strict 100% reference run remains stable throughout training, with safe pass rate 0.79 and harmful block rate 1.00 at the final checkpoint. Under incomplete OOD coverage with learning rate 1×10^{-5}, the behavior remains nearly unchanged, ending at safe pass rate 0.78 and harmful block rate 1.00. At learning rate 4×10^{-5}, the harmful block rate still remains at 1.00, but the safe pass rate decreases to 0.59, indicating that



(a) Llama-3.1-8B on JailbreakBench.



(b) Llama-3.1-8B on HarmfulBench.



(c) Llama-3.1-8B on BeaverTails-H.

Figure 9: Ratio-scatter comparisons for Llama-3.1-8B before and after continued safe training. Within each row, the left panel shows the model before training and the right panel shows the same model after training completes. The empirical boundary shifts during training, but the safe–harmful separation remains visible in all three settings, with the margin narrowing most noticeably on BeaverTails-H.

the backward gate induced by the frozen threshold remains active but becomes more conservative on safe samples. By contrast, at learning rate 3×10^{-4}, the final safe pass rate rises to 1.00 while the harmful block rate drops to 0.00, showing that the original harmful–safe separation no longer remains stable in this high-update regime.

Figure 10 shows the corresponding dynamics over training. The strict 100% reference run is flat across all monitored checkpoints, and the 1×10^{-5} run stays close to that reference trajectory. The 4×10^{-5} run preserves harmful blocking but gradually shifts toward lower safe pass rate, whereas the 3×10^{-4} run rapidly loses harmful blocking after only a few monitored steps. Figure 11 is consistent with this interpretation: the ratio-scatter geometry remains visually similar for the strict reference and 1×10^{-5} runs, compresses toward the threshold boundary at 4×10^{-5}, and collapses at 3×10^{-4} so that the frozen threshold no longer cleanly separates harmful from safe samples.

Taken together, these results support a conservative interpretation of fixed-threshold deployment. When the release-time threshold is calibrated with complete harmful coverage, the corresponding

Model	Build domain	Null-space coverage				
		25% n=75	**50%** n=50	**75%** n=25	**85%** n=15	**95%** n=5
Qwen-14B	JailbreakBench	0.77	0.78	0.72	0.80	1.00
	HarmfulBench	0.00	0.02	0.00	0.00	0.00
	BeaverTails-H	0.01	0.02	0.00	0.00	0.00
Llama-8B	JailbreakBench	0.48	0.44	0.56	0.67	1.00
	HarmfulBench	0.00	0.02	0.00	0.00	0.00
	BeaverTails-H	0.01	0.02	0.00	0.00	0.00

Table 12: **Blocking generalizes to non-participating harmful only for a coherent forbidden distribution.** Non-participating block rate: the null space is built on a fraction (*coverage*) of a harmful set, and we report the fraction of the *non-participating* harmful samples—from the same distribution but never used in construction—whose norm ratio falls below the safe-preserving release threshold $\tau = \min_h r(h_{\text{safe}})$. Safe data passes at $\geq 99\%$ (calibration Alpaca) and 100% (unseen Dolly) in every cell. JailbreakBench generalizes—a majority of non-participating harmful is blocked, rising to full separation as coverage grows—whereas HarmfulBench and BeaverTails-H, heterogeneous unions of many harmful behaviors, do not (≈ 0). Harmful and safe share the same prompt template, so the separation is not a formatting artifact. The value is measured at the frozen release gate before any attack and is therefore independent of the attack learning rate.

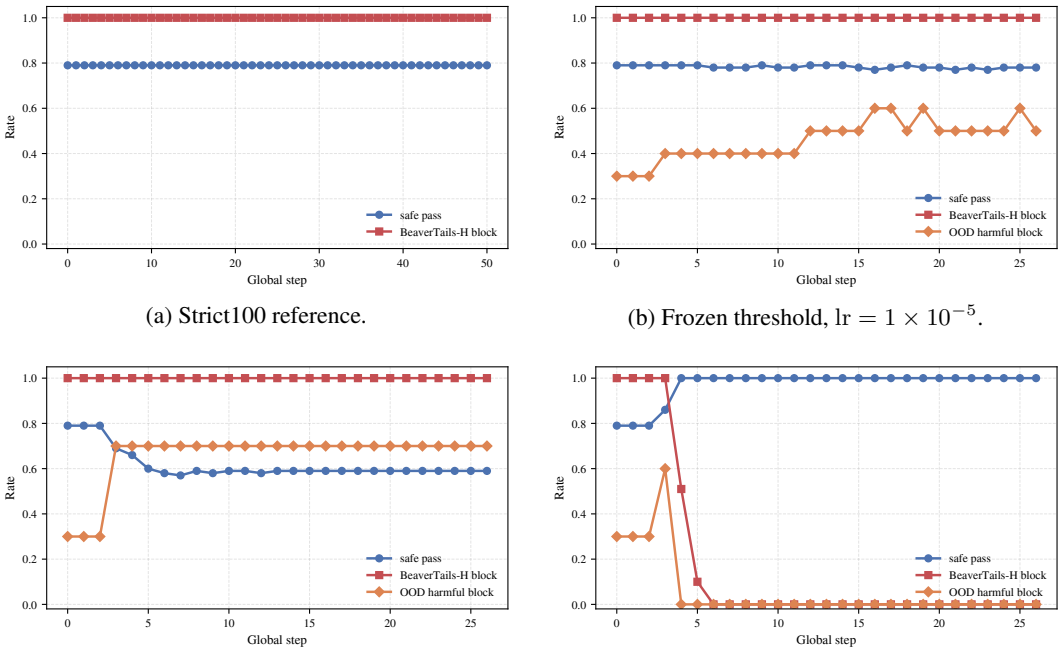
Table 13: **Gate erosion is domain-local.** Separability of the (untrained) protected build set from safe data after an attacker fine-tunes on non-participating harmful data, as a function of the *build* domain (rows) and the *attacker's fine-tuning* domain (columns); underlined diagonal entries are same-domain. **1.00**: build fully blocked at the frozen release threshold (separable as deployed). **1.00**†: not separable at the frozen threshold, but build stays below all safe ratios, so a single recalibration restores full separation. **0.00**: build overshoots the safe side—not separable even after recalibration. Learning rate 4×10^{-5}; at 1×10^{-5} every cell is at least $1.00^\dagger$ (all recoverable). The build set is excluded from gradients throughout, so its drift is pure generalization. Domains: JailbreakBench (JBB), HarmfulBench (HB), BeaverTails-H (BT).

	Qwen-14B			**Llama-8B**		
Build domain \ attacker FT	JBB	HB	BT	JBB	HB	BT
JailbreakBench (JBB)	0.00	1.00†	1.00	0.00	1.00†	1.00†
HarmfulBench (HB)	1.00†	0.00	1.00†	0.00	0.00	0.00
BeaverTails-H (BT)	1.00†	1.00†	0.00	1.00†	1.00†	0.00

backward gate can remain stable under mild continued training. When continued adaptation is OOD with respect to the release-time calibration set, however, stability becomes update-dependent: small updates can preserve the threshold-controlled gate, intermediate updates can make it more conservative, and sufficiently large updates can erase the original harmful–safe separation.

Setting	LR	Final Safe Pass	Final Harmful Block
Strict-100 reference	–	0.79	1.00
Frozen threshold	1×10^{-5}	0.78	1.00
Frozen threshold	4×10^{-5}	0.59	1.00
Frozen threshold	3×10^{-4}	1.00	0.00

Table 14: Final monitored operating points on Qwen3-14B with BeaverTails-H under a fixed strict-100 release threshold; safe pass is measured on the out-of-distribution Dolly set. Mild benign distribution shift preserves the reference operating point, moderate learning rate reduces safe pass while keeping full harmful blocking, and aggressive learning rate destroys the original harmful–safe separation.



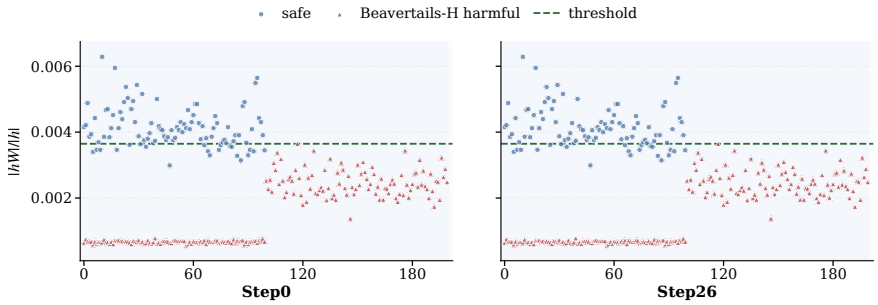
(a) Strict100 reference.

(b) Frozen threshold, lr $= 1 \times 10^{-5}$.

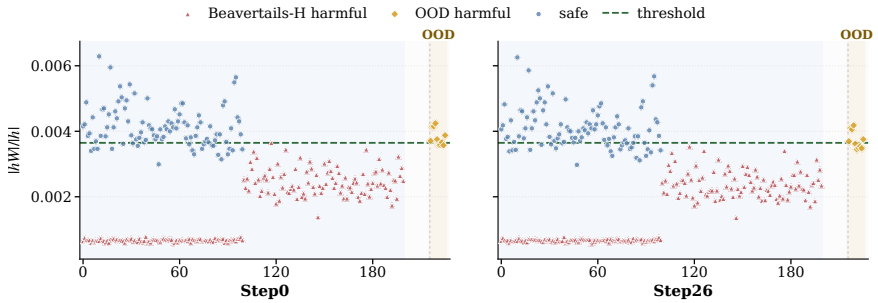
(c) Frozen threshold, lr $= 4 \times 10^{-5}$.

(d) Frozen threshold, lr $= 3 \times 10^{-4}$.

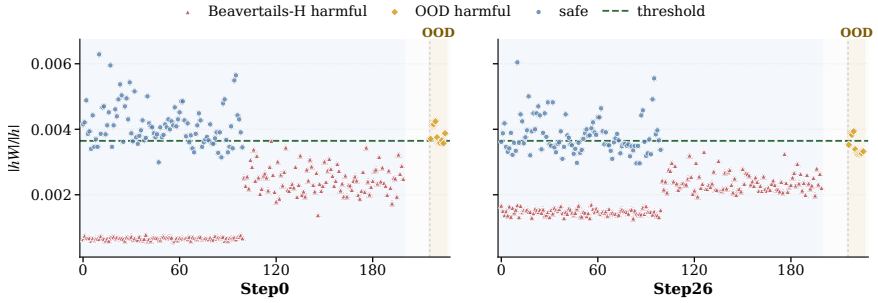
Figure 10: Safe-pass and harmful-block trajectories under a fixed strict-100 release threshold. The strict reference run and the 1×10^{-5} OOD run remain close to their initial operating points, the 4×10^{-5} run keeps harmful blocking but sacrifices safe pass rate, and the 3×10^{-4} run rapidly loses harmful blocking.



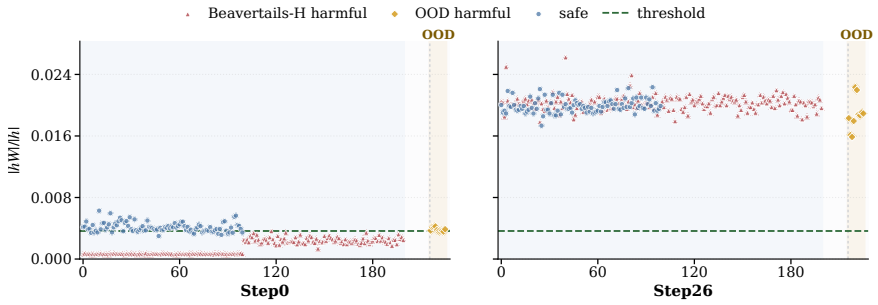
(a) Strict100 reference.



(b) Frozen threshold, $\mathrm{lr} = 1 \times 10^{-5}$.



(c) Frozen threshold, $\mathrm{lr} = 4 \times 10^{-5}$.



(d) Frozen threshold, $\mathrm{lr} = 3 \times 10^{-4}$.

Figure 11: Ratio-scatter comparisons at initialization and the final monitored step under a fixed strict-100 release threshold. The strict reference and 1×10^{-5} runs preserve the original safe–harmful separation, the 4×10^{-5} run pushes safe samples toward the threshold boundary, and the 3×10^{-4} run collapses the separation so that the frozen threshold no longer cleanly separates harmful from safe samples.