COORDINATE CONCENTRATION OF SAFETY ALIGNMENT IN LOW-RANK ADAPTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning a safety-aligned language model on benign data is enough to degrade its refusal behavior. The established defense projects the low-rank update away from an alignment-sensitive subspace, which assumes that the update has to be steered around safety rather than kept out of it. We ask where that subspace sits among the coordinates of the adapter itself. If safety energy concentrates on a small fraction of coordinates rather than spreading across all of them, those coordinates can be identified once and held fixed during any subsequent fine-tuning, replacing the projection with a simpler intervention that does not depend on the downstream task. To answer this, we approximate the curvature of the loss on harmful prompts and measure how sensitive the model's safety behavior is to changes at each adapter coordinate. Across three models from two families, we find that the distribution of safety-sensitive energy is heavy-tailed, and one tenth of the coordinates carries 90 to 96% of it, a property we call coordinate concentration. Holding that tenth fixed during fine-tuning preserves refusal behavior across four base models and two harmful-prompt benchmarks, performing comparably to or better than three published defenses at the highest task loss among them, while requiring nothing of the downstream user beyond attaching the mask. The concentration also explains why targeted selection matters even at a heavy lock rate. A random mask of equal size locks nearly the same coordinates as a targeted one, yet performs substantially worse, because the small set of coordinates each leaves free to train are almost entirely different, and safety is determined by that free set.

1 INTRODUCTION

A model provider ships one base model to many downstream users, each of whom fine-tunes it on a task the provider never sees. Fine-tuning on that task, even when the data contains no harmful content, silently undoes alignment: attack success rates on the aligned model rise to 30 to 50% after a few epochs of ordinary supervised fine-tuning on instruction data (Qi et al., 2024; He et al., 2024; Peng et al., 2024).

The most developed response treats alignment as occupying a subspace of parameter space and constrains the user's update to avoid it. SafeLoRA (Hsu et al., 2024) projects against the aligned-minus-base weight difference, SaLoRA (Li et al., 2025) against an activation subspace, SafeGrad (Yi et al., 2025) against a runtime safety gradient, and AlignGuard-LoRA (Das et al., 2025) decomposes the update by Fisher information and regularises the collision between its two parts. All four assume that such a subspace exists and project the update away from it.

Previous work has identified which parts of the full model are responsible for safety behavior. Wei et al. (2024) find safety-critical neurons and ranks; Shen et al. (2025) identify *safety layers* and freeze them during fine-tuning. A low-rank adapter introduces its own set of trainable parameters, and how the safety subspace distributes over those parameters has not been measured. What decides whether safety can be protected by locking part of the adapter is how the safety subspace distributes over the adapter's own coordinates. If safety energy spreads uniformly across them, projection is the only option. If it concentrates, the subspace can be held fixed instead.

To measure this, we attach a fresh adapter to the base model and approximate the curvature of the loss on harmful prompts via L-BFGS. The curvature identifies which combinations of adapter

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.

parameters the model's safety behavior depends on most. From there, measuring how much each individual parameter contributes to those combinations reveals how the safety signal distributes across the adapter. Across three models from two families the answer is consistent: the distribution spans four to five orders of magnitude, with a small fraction of the parameters carrying nearly all of the safety energy (Figures 1 and 3b). We call this property *coordinate concentration* and the top tenth by safety energy the *safety decile*.

Holding the safety decile fixed during fine-tuning and leaving the rest of the adapter free to train preserves refusal behavior. The provider builds the patch once from harmful prompts and the base model, and a downstream user applies it by attaching the mask before training, with no need for harmful data, a retain set, or any change to the training objective. Across four base models and two harmful-prompt benchmarks, the patch performs comparably to or better than three published defenses, at a higher task loss, while being the only one that requires nothing from the downstream user at training time (§4.2).

Coordinate concentration also makes it possible to analyze why the patch works and under what conditions it stops working. When most of the adapter is locked, any two masks of equal size inevitably share most of their locked coordinates, so comparing them on what they lock is uninformative. What differs between a targeted mask and a random one is the small set of coordinates each leaves trainable, and it is that set which determines safety behavior. This explains why a random mask of the same size performs substantially worse despite covering almost the same coordinates (§4.3). Separately, coordinate concentration turns out to be independent of how many directions in parameter space carry the safety signal, a property we call spectral spread. The two govern different aspects of the patch's lifecycle: concentration determines whether a mask can capture the safety signal, while spread appears to govern whether the curvature estimate remains valid as the adapter moves through a sequence of tasks (§4.4).

Our contributions are:

- **Coordinate concentration**: the safety subspace of a low-rank adapter concentrates on a small fraction of its coordinates across three models from two families, a property that can be measured from harmful prompts alone at the base model (§2).
- **A safety patch derived from the measurement**: built once by the provider, it performs comparably to or better than three published defenses across two harmful-prompt benchmarks and four base models, at a higher task loss, while requiring nothing of the downstream user beyond attaching the mask (§4.2).
- **An overlap bound and targeting analysis**: when most of the adapter is locked, two masks of equal size differ almost exclusively on which coordinates they leave free to train, and it is that free set which determines safety. This separates the targeted mask from chance on every model and predicts that informed importance scores select nearly the same mask (§2.3, §4.3).
- **Concentration and spectral spread as independent properties**: concentration determines whether a mask can capture the safety signal, while spread appears to govern whether the curvature estimate remains valid across a multi-task sequence. The two come apart across models, and spread, not concentration, tracked which model needed re-anchoring in our continual experiments (§4.4).

2 How Safety Distributes Over Adapter Coordinates

This section reports the main measurement and derives its consequences. We first set up the curvature approximation, then measure coordinate concentration, analyze what it implies for mask comparisons, bridge it to a practical importance score, and distinguish it from a property it is easy to confuse it with: spectral compactness.

2.1 Curvature setup at the base model

Let $\theta_0 \in \mathbb{R}^d$ denote the parameters (Hu et al., 2022) of a fresh low-rank adapter (rank $r = 32$ on q- and v-projections) attached to a safety-aligned base model, and let $L_{\text{forget}}(\theta)$ denote the cross-entropy loss on a held-out set of harmful prompts drawn from HEx-PHI (Qi et al., 2024). We

want to identify which adapter coordinates the model's safety behavior depends on, which requires estimating the Hessian $H = \nabla^2 L_{\text{forget}}(\theta_0)$ restricted to the adapter subspace.

The exact Hessian is intractable, so we build an L-BFGS approximation (Liu & Nocedal, 1989) from $m = 20$ trust-region descent steps on L_{forget}, giving curvature pairs $\{s_i, y_i\}_{i=1}^{m}$. The two-loop recursion yields an operator H^{-1} that is accurate within $\text{span}(s_1, y_1, \ldots, s_m, y_m)$ and a scalar multiple of the identity outside it, so we extract eigenpairs by projecting onto that span and diagonalizing the resulting $\leq 2m$-dimensional matrix exactly (Appendix D.1). Every operator in the main text is built forget-only, from HEx-PHI and the base model, with no retain set. A diagnostic that builds the history against a reference retain gradient, and shows that the eigenvectors then split into safety- and task-biased directions, is in Appendix B.1.

2.2 One tenth of the coordinates carries almost all the safety energy

We now ask how the safety-relevant curvature directions distribute over the adapter's individual coordinates. For each coordinate c, define the per-coordinate *safety membership*

$$m(c) \;=\; \sum_{i \in \mathcal{S}_K} q_i[c]^2,$$

where $\mathcal{S}_K$ is the set of the K eigenvectors of the forget-only operator most aligned with g_{forget}. Because each q_i is unit-norm, $\sum_c m(c) = K$ and the uniform-baseline value is K/d.

Figure 1 shows the distribution of $\log_{10} m(c)$ across all adapter coordinates for three models. The distributions are heavy-tailed: most coordinates fall below the uniform baseline, while a small fraction reach values several orders of magnitude above it. We define the *coordinate concentration* κ as the fraction of total membership carried by the top decile of coordinates, and call that decile the *safety decile*. On all three models, the safety decile carries roughly nine tenths of the total safety energy (Figure 3b), meaning that the vast majority of the adapter's coordinates contribute negligibly to the safety subspace. The statistic alone does not show that the heavy tail is specific to safety, since per-coordinate gradient mass in an adapter can be heavy-tailed for other reasons. The evidence that these particular coordinates matter for safety is behavioral: targeted, random, inverted and reduced-rank controls at matched budget (§4.3).

Consequence C1: locking the safety decile should preserve refusal behavior. If nearly all of the safety energy is carried by a small, identifiable set of coordinates, holding them fixed during fine-tuning should preserve alignment while leaving the rest of the adapter free for task learning. Because the decile is identified at θ_0 from harmful prompts alone, the resulting mask should not depend on the downstream task (tested in §4.2).

2.3 Targeted selection matters despite near-identical masks

An immediate question is whether the specific coordinates in the safety decile matter, or whether any mask that locks the same fraction of the adapter would work equally well. At a lock rate of 90% the answer is not obvious, because two masks of that size are forced by arithmetic to share most of their locked coordinates.

Proposition 1 (Overlap floor)**.** *Let $M_1, M_2 \subseteq [d]$ each lock $(1-\alpha)d$ of the d adapter coordinates. Then*

$$|M_1 \cap M_2| \;\geq\; (1-2\alpha)\, d, \qquad \text{IoU}(M_1, M_2) \;\geq\; 1 - 2\alpha.$$

Proof. $|M_1 \cap M_2| = |M_1| + |M_2| - |M_1 \cup M_2| \geq 2(1-\alpha)d - d = (1-2\alpha)d$, since $|M_1 \cup M_2| \leq d$. Dividing by $|M_1 \cup M_2| \leq d$ gives the second inequality. □

At our operating point of $\alpha = 0.1$, the bound guarantees that any two masks have a locked-set IoU of at least 0.8, and the measured overlap between our mask and a size-matched random mask closely matches that floor. If two nearly identical masks behave alike, nothing follows about whether the selection was meaningful.

The bound constrains the locked sets, and its corollary shows that the trainable sets are not similarly constrained.

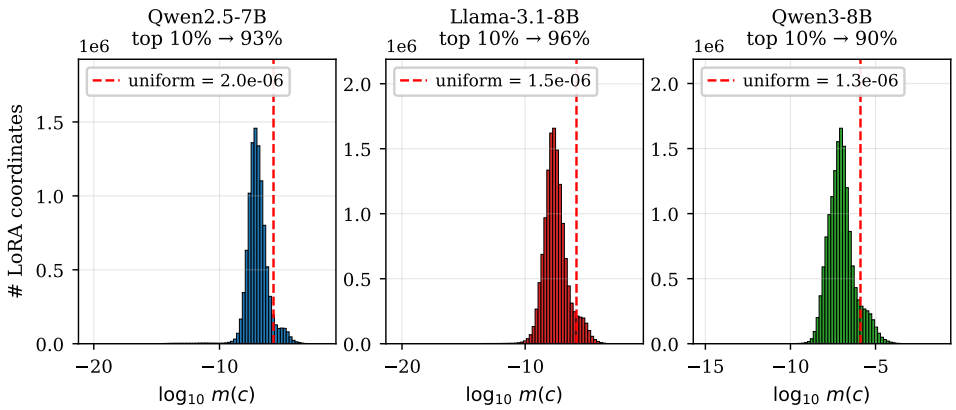


Figure 1: Safety energy concentrates on a small minority of adapter coordinates. Distribution of the per-coordinate safety membership $m(c)$ across three models ($K = 20$, forget-only L-BFGS at a fresh adapter initialization). Each histogram peaks well below the uniform baseline K/d (red dashed) and carries a heavy right tail reaching four to five orders of magnitude above it, so the coordinates that matter for safety are a small minority of the adapter. Each panel heading gives that model's *safety decile* share: the fraction of total membership carried by its top tenth of coordinates.

Corollary 1 (The trainable sets are unconstrained)**.** *Let $T_i = [d] \setminus M_i$ be the coordinates mask M_i leaves trainable, so $|T_i| = \alpha d$. Then $|T_1 \cap T_2| = d - |M_1 \cup M_2|$, which equals zero exactly when M_1 and M_2 attain the floor of Proposition 1. For every $\alpha \leq 1/2$ that floor is attainable, so two masks may overlap maximally as locked sets while sharing no trainable coordinate at all.*

Proof. $|T_1 \cap T_2| = d - |M_1 \cup M_2|$ by De Morgan. Substituting $|M_1 \cup M_2| = 2(1-\alpha)d - |M_1 \cap M_2|$ gives $|T_1 \cap T_2| = |M_1 \cap M_2| - (1-2\alpha)d$, which vanishes precisely at the floor. The floor is non-negative, and hence attainable, iff $\alpha \leq 1/2$. □

Consequence C2: selection should be visible in the trainable set. Since fine-tuning moves only the trainable coordinates, a targeted mask and a random mask should differ in safety behavior even though they lock nearly the same coordinates. The gap should be attributable to which coordinates each leaves free, not to which coordinates each holds fixed. Tested in §4.3, alongside an inversion counterfactual and a lock sweep.

2.4 A Hadamard influence score detects the same coordinates

The eigenvector decomposition identifies the safety-critical coordinates but requires extracting and storing the full set of safety eigenvectors. A more practical alternative is a single per-coordinate score that can be computed directly from the gradient and inverse Hessian. Define the **Hadamard influence**

$$\text{importance}(c) \;=\; \big|g_{\text{forget}}[c] \cdot (H^{-1} g_{\text{forget}})[c]\big|.$$

In the eigenbasis of H, writing $g_{\text{forget}} = \sum_i a_i q_i$,

$$g_{\text{forget}}[c] \cdot (H^{-1} g_{\text{forget}})[c] \;=\; \underbrace{\textstyle\sum_i \frac{a_i^2}{\lambda_i}\, q_i[c]^2}_{\text{diagonal}} \;+\; \underbrace{\textstyle\sum_{i \neq j} \frac{a_i a_j}{\lambda_j}\, q_i[c] q_j[c]}_{\text{off-diagonal}}.$$

The diagonal term is a curvature-weighted version of the safety membership $m(c)$: coordinates with high $q_i[c]^2$ for eigenvectors with large gradient projection and small curvature receive a high score. If the off-diagonal terms average to noise across coordinates, the score approximates $m(c)$ up to a curvature reweighting, and setting H^{-1} to the identity recovers the Fisher diagonal, so the two scores differ exactly by that reweighting.

Figure 2 confirms the correspondence empirically: coordinates with high safety membership systematically receive high Hadamard influence, with a moderate log-log correlation $r = 0.55$ to 0.67) and a monotonically increasing binned mean on all three models. The practical score can therefore replace the full eigenvector decomposition for mask construction.

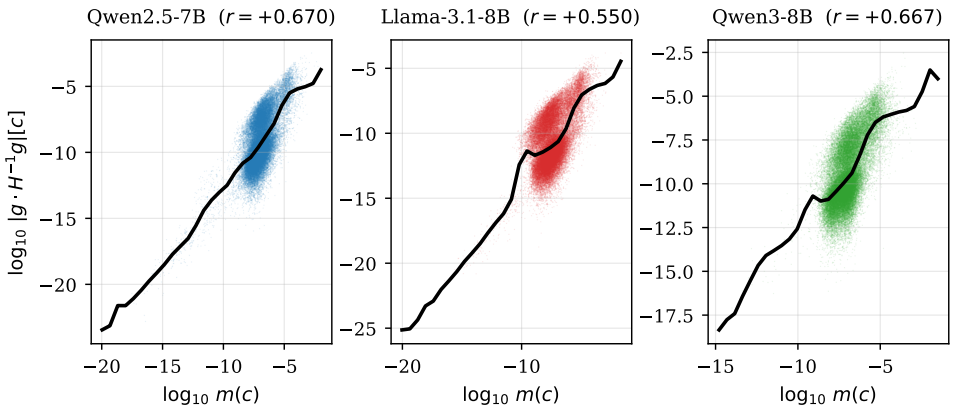
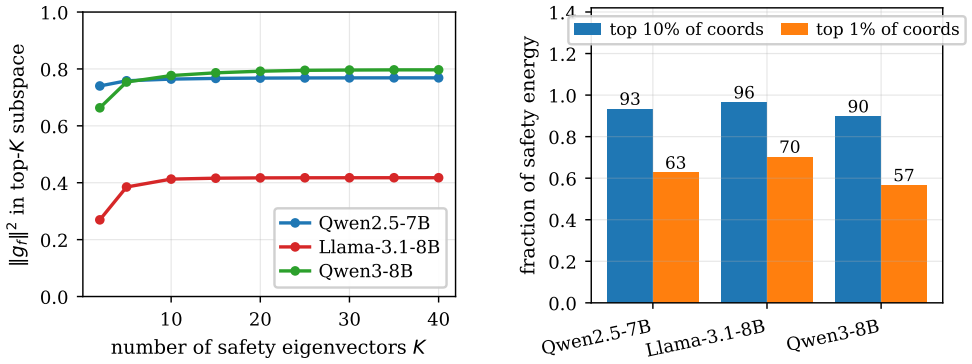


Figure 2: The Hadamard influence score detects the same coordinates as the full eigenvector decomposition. Each point is one adapter coordinate (a 50,000-coordinate random subsample is plotted); the black line is the binned mean over all coordinates. Coordinates with high safety membership $m(c)$ systematically receive high Hadamard influence on all three models: the binned mean increases monotonically in every case, and the log-log Pearson correlation r reported in each panel heading is positive but moderate (0.55 to 0.67). The practical score can therefore stand in for the full eigenvector decomposition when ranking coordinates to build the mask, though it does not reproduce $m(c)$ coordinate by coordinate.



(a) Top-K capture

(b) Top-10% concentration

Figure 3: Two measures of localization come apart, and the gap is predictive. (a) *Spectral spread*, read as top-K capture: Qwen2.5-7B and Qwen3-8B saturate at $\sim$ 0.78 of forget-gradient energy by $K \approx 20$, while Llama-3.1-8B captures only $\sim$ 0.42, so its safety signal spans many more directions. (b) *Coordinate concentration*: the share of safety energy carried by the top decile and the top percentile of coordinates, labelled in percent above each bar. Llama shows the *tightest* concentration (96.4%) despite having the lowest top-K capture, so sitting on few parameters and spanning few directions are independent properties. In §4.4 this contrast matches which models tolerated a stale curvature estimate.

Consequence C3: informed scores should select nearly the same mask at heavy lock. Because both scores rank the same heavy tail, two masks built from different scores should select nearly the same coordinates. Whether the few coordinates on which they disagree matter behaviorally does not follow from the overlap and has to be measured.

2.5 Concentration is not spectral compactness

Coordinate concentration measures how many individual parameters carry the safety signal. A separate question is how many independent directions in parameter space carry it, which we call *spectral spread*: the fraction of forget-gradient energy that the top K eigenvectors fail to represent.

The two properties come apart (Figure 3). Two of the three models concentrate most of their forget-gradient energy in the top K eigenvectors, giving them low spectral spread, while the third spreads its safety signal across much more of the spectrum. Yet that third model has the highest coordinate concentration of the three: its safety energy sits on the fewest parameters despite spanning the most directions. A model can concentrate its safety signal on few parameters while spreading it across many directions, and the distinction matters because the two properties govern different aspects of the patch. Concentration determines whether a mask can capture the safety signal, while spread determines how well a curvature estimate built at initialization describes the safety subspace after the adapter has been trained.

Consequence C4: spectral spread, not concentration, should govern staleness. A model whose safety signal spans many eigenvectors is one whose curvature estimate at θ_0 becomes inaccurate as the adapter drifts during training, because the directions that matter are poorly represented by a small fixed basis. High spectral spread should therefore predict a need to refresh the curvature estimate across a multi-task sequence, even if coordinate concentration is high. With a single spectrally spread model in our set, we treat C4 as a hypothesis that §4.4 tests on one case.

3 A Safety Patch Derived from the Measurement

The concentration measurement leads directly to a method. If nearly all safety energy sits on a small identifiable set of coordinates, the provider can lock those coordinates once and leave the rest of the adapter free for any downstream task.

Score. At the base model with fresh adapter θ_0, build the L-BFGS history $\{s_i, y_i\}$ on the forget set, then compute

$$\text{importance}(c) \;=\; \big|g_{\text{forget}}[c] \cdot \text{LBFGS}(s, y, g_{\text{forget}})[c]\big|,$$

where $\text{LBFGS}(s, y, v)$ applies the two-loop recursion to compute $H^{-1}v$.

Per-tensor rank-based selection. The parametrization $\Delta W = BA$ produces gradients of asymmetric magnitude at fresh initialization, since $\partial L/\partial A \sim |B|$ and $\partial L/\partial B \sim |A|$, so a global threshold would lock almost exclusively the B tensors. We therefore compute the threshold per tensor: within each adapter matrix, we lock the top $(100-p)\%$ by importance rank, independently of the others. We also select by rank rather than by value, because importance values for many low-magnitude coordinates quantize to numerically identical near-zero values in mixed-precision storage and a value-based criterion would catch all ties at once. Rank-based selection produces an exact $(100-p)\%$ lock per tensor.

Algorithm 1 Curvature-aware safety patch. The provider runs the build phase once; the user runs the deploy phase.

Build (provider). Input: base model $\mathcal{M}$, harmful prompts $\mathcal{D}_{\text{forget}}$, adapter config C, history size m, lock percentile p
Attach fresh adapter following C; snapshot θ_0
$(s_1, y_1, \ldots, s_m, y_m) \leftarrow \text{LBFGSHistoryTrustRegion}(\mathcal{M}, \mathcal{D}_{\text{forget}}, m)$
$g_{\text{forget}} \leftarrow$ flat adapter gradient of L_{forget} on held-out probe (`model.eval()` mode)
$\text{importance} \leftarrow |g_{\text{forget}} \odot \text{LBFGS}(s, y, g_{\text{forget}})|$; initialize $M \in \{0,1\}^d$ to zero
for each adapter tensor T **do**
    $M[\text{topk}(\text{importance}|_T,\ \text{round}((1-p/100)\cdot|T|))] \leftarrow 1$
end for
Return patch (θ_0, M, s, y)

Deploy (user). Input: $\mathcal{M}$, C, patch (θ_0, M, s, y), task data $\mathcal{D}_{\text{task}}$
Attach fresh adapter; restore θ_0
for each training step **do**
    $g \leftarrow \nabla L_{\text{task}}(\text{batch})$; $g[M] \leftarrow 0$; optimiser.step()
end for

Table 1: The patch performs comparably to three published defenses across two harmful distributions. Attack success rate (↓) on AdvBench (500 prompts) and on the 200 standard behaviors of HarmBench, both keyword-scored, mean ± std over three seeds. Eval loss (↓) is on the held-out Dolly split; utility (↑) is the mean of ARC-C, BoolQ, HellaSwag and WinoGrande (0-shot). "Base" is the aligned model with no fine-tuning. Each baseline runs at a single declared configuration across all models (Appendix D.2).

Model	Method	AdvBench ↓	HarmBench ↓	Eval loss ↓	Utility ↑
Qwen2.5-7B	Base	0.6	14.0	—	73.2
	No defence	20.1 ± 1.8	57.5 ± 1.0	1.549	75.1
	SafeLoRA	5.7 ± 0.7	35.3 ± 1.0	1.487	74.1
	AlignGuard	10.9 ± 0.5	42.0 ± 2.5	1.541	74.0
	Antidote	5.3 ± 0.1	31.0 ± 1.7	1.503	74.7
	Ours	3.1 ± 0.5	24.7 ± 1.5	1.582	73.9
Llama-3.1-8B	Base	3.8	10.0	—	73.1
	No defence	59.9 ± 4.8	78.7 ± 1.5	1.566	72.2
	SafeLoRA	19.2 ± 5.4	48.0 ± 5.6	1.493	72.6
	AlignGuard	28.5 ± 5.6	55.2 ± 4.3	1.540	73.0
	Antidote	12.7 ± 5.6	38.2 ± 5.0	1.501	72.9
	Ours	6.1 ± 1.3	30.5 ± 1.8	1.592	72.4
Qwen3-8B	Base	1.2	16.5	—	71.6
	No defence	34.5 ± 5.3	67.8 ± 1.4	1.613	72.9
	SafeLoRA	7.1 ± 2.2	40.8 ± 3.3	1.554	72.4
	AlignGuard	11.1 ± 1.2	42.2 ± 1.5	1.606	72.5
	Antidote	2.1 ± 0.6	21.5 ± 4.1	1.608	72.6
	Ours	3.3 ± 0.5	20.7 ± 2.6	1.719	71.6
Qwen3-32B	Base	3.0	23.5	—	75.9
	No defence	13.2 ± 1.2	44.2 ± 0.8	1.355	77.2
	AlignGuard	12.0 ± 2.1	44.7 ± 2.5	1.486	76.4
	Antidote	4.4 ± 0.4	25.7 ± 2.3	1.472	76.6
	Ours	4.6 ± 1.5	21.3 ± 1.9	1.558	75.7

Task-agnostic deployment. The build phase consumes only harmful prompts and the base model, with no task data, retain set, or aligned-unaligned pairing required. The resulting patch (θ_0, M, s, y) is stored as a single file alongside the base model. The downstream user attaches a fresh adapter, restores θ_0, and trains normally with the masked coordinates held fixed. Appendix B.2 verifies that the resulting mask captures nearly all of the safety subspace's energy at our operating point.

4 Experiments

4.1 Setup

Models. Four instruction-tuned base models spanning two architecture families: Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, Qwen3-8B, and Qwen3-32B. **Fine-tuning data.** Single-task experiments fine-tune on Databricks Dolly-15k (Conover et al., 2023), a general instruction dataset with no malicious content. Continual-learning experiments use the four-task sequence of Bach et al. (2026a): Dolly → GSM8K → MedMCQA → Squad-v2. **Safety evaluation.** Attack success rate is measured on the first 500 of the 520 AdvBench behaviors (Zou et al., 2023) and on the 200 standard behaviors of HarmBench (Mazeika et al., 2024), both using keyword refusal matching. Appendix C.6 validates the keyword rule against a purpose-trained classifier. **Adapter and patch construction.** Rank 32, $\alpha = 64$, on q- and v-projections of every attention layer. For each base model we build a single patch following Algorithm 1 using 200 examples from HEx-PHI (Qi et al., 2024), then apply it unchanged to every fine-tuning run on that model.

Table 2: Selection, not lock rate, is what the score contributes. Targeting controls at the $p = 10$ operating point. "Anti" locks the lowest-ranked coordinates; "Random" locks a size-matched random subset, drawn per tensor like the targeted mask; $r = 3$ and $r = 8$ are unmasked adapters at reduced rank. Eval loss in parentheses. All cells seed 42, so the Baseline column differs from the three-seed No-defence row of Table 1; differences of a few pp are within seed noise.

Model	Base	Baseline	$r = 8$	$r = 3$	Random	Anti	Ours
Qwen2.5-7B	0.6	18.8 (1.50)	12.2 (1.52)	7.8 (1.53)	11.5 (1.55)	7.2 (1.54)	2.0 (1.55)
Llama-3.1-8B	3.8	51.0 (1.55)	44.0 (1.55)	39.4 (1.55)	20.4 (1.57)	26.6 (1.56)	6.0 (1.54)
Qwen3-8B	1.2	26.4 (1.55)	28.4 (1.59)	26.8 (1.59)	15.3 (1.66)	6.8 (1.61)	4.0 (1.75)

4.2 Locking the safety decile preserves refusal behavior (C1)

Table 1 compares the patch against three published defenses: SafeLoRA (Hsu et al., 2024), AlignGuard-LoRA (Das et al., 2025), and Antidote (Huang et al., 2025a). The patch cuts attack success substantially on every model. It has the lowest attack success in six of the eight model–benchmark cells and trails Antidote by 0.2 and 1.2 pp in the other two (AdvBench on the Qwen3 models). What separates them further is the deployment profile: Antidote requires a post-hoc pruning pass with harmful completions, SafeLoRA needs the pre-alignment checkpoint (unavailable for Qwen3-32B), while the patch requires nothing from the downstream user beyond attaching it. The patch pays for this with the highest eval loss of the four methods, though the four-task utility score stays within a point of the base model on every model (Appendix C.5). Building the patch once and applying it unchanged to Dolly, Alpaca, and CodeAlpaca drives attack success to at most 3% on all three tasks, confirming that the mask does not depend on the downstream data (Appendix C.1, Table 5).

4.3 Targeted selection separates from chance (C2, C3)

The result in Table 1 is consistent with accounts that have nothing to do with the safety decile: the patch could be working simply because it reduces adapter capacity, or because any heavy mask would do equally well. Table 2 tests three alternatives that rule these out, and a score ablation tests whether the specific importance formula matters.

Reducing adapter rank to match the trainable parameter count does not reproduce the effect: reduced-rank adapters leave attack success four to seven times that of the targeted mask on every model. A size-matched random mask performs better than no defense but substantially worse than the targeted mask on every model, consistent with the overlap bound's prediction that the difference should appear in the trainable set (C2). Inverting the ranking so that the most safety-critical coordinates are left free degrades safety relative to the targeted mask on every model, confirming that the importance score is causally load-bearing rather than incidental; the effect is large only on Llama-3.1-8B, and the inverted mask still beats random on the other two models (Appendix C.2). A layer-granular mask at matched budget reaches comparable safety on two of three architectures, though at higher task cost, and fails outright on the third, where it performs worse than no defense at all. The failure arises because the safety decile spreads across all layers even where safety energy does not, so freezing whole layers either releases safety-critical coordinates or locks nearly the entire adapter (Appendix B.3). Full analysis of all four controls is in Appendix C.2.

Replacing the Hadamard influence score with the Fisher diagonal produces a mask that overlaps with the original at IoU > 0.97 on the locked coordinates, as C.3 predicts. The behavioral difference depends on training pressure: under light training the two masks give identical attack success, while under the heavier configuration of Table 1 the Hadamard mask is lower by 0.8 to 4.4 pp on all four models (single seed). In the continual setting the order reverses on Llama-3.1-8B, where Fisher needs no curvature and so cannot go stale (Appendix C.4). The score is therefore a secondary choice next to having an informed score at all, and a Fisher mask shrinks the patch from ~ 1.6 GB to ~ 90 MB (Appendix D.3). The full score comparison and the lock sweep on Qwen3-32B are in Appendix C.3 and Appendix C.2.

Table 3: Locking trades peak task accuracy for more stable continual behavior. Forgetting metrics on the trainable tasks against Moderate-G_i (Bach et al., 2026a), the strongest data-centric prior on this benchmark. BWT: backward transfer (↑). FM: forgetting measure (↓). Max Drop: worst single-step drop (↓). Avg Perf: checkpoint-averaged accuracy (↑). ASR is checkpoint-averaged AdvBench attack success (↓). The other baselines of that benchmark, including O-LoRA, have higher attack success than Mod-G_i there.

Model / Method	Avg Perf ↑	BWT ↑	FM ↓	Max Drop ↓	ASR ↓
Qwen2.5-7B Mod-G_i	60.9	−2.7	2.7	10.6	10.2
Qwen2.5-7B Ours	52.1	−2.0	2.4	2.3	0.8
Llama-3.1-8B Mod-G_i	46.4	−1.7	1.7	5.4	18.3
Llama-3.1-8B Ours	39.2	+0.2	0.1	1.0	5.4
Qwen3-4B Mod-G_i	58.1	−4.3	4.3	5.6	6.0
Qwen3-4B Ours	50.5	−0.2	0.2	0.4	0.3

4.4 Spectral spread predicts when curvature must be re-anchored (C4)

We reuse the patch across the four-task continual sequence and the three models of Bach et al. (2026a), so that its reported baseline numbers compare directly (hence Qwen3-4B, whose spread we assume from Qwen3-8B). The lifecycle policy follows the spectral spread diagnostic of §2.5: full re-anchoring for the spectrally spread Llama-3.1-8B, gradient-only refresh for the compact Qwen models. Table 3 shows lower checkpoint-averaged attack success than the data-centric baseline on every model and improved forgetting metrics, by large factors on Llama-3.1-8B and Qwen3-4B, at a cost of 7 to 9 points of average task performance.

The outcome is consistent with C4. Under the stale curvature estimate, Llama-3.1-8B collapses on one seed after the third task; re-anchoring at each task boundary removes that collapse, though a residual failure on another seed remains (Appendix C.4, Figure 9), and costs about an hour of L-BFGS per boundary, the one setting where the patch asks more of the user than attaching the mask. The compact models tolerate the stale estimate, suggesting that a build-time quantity can prescribe the lifecycle policy, though with one spectrally spread model this is a single case rather than a validated rule.

4.5 Over-refusal and capability

A defense evaluated only on harmful prompts can score well by refusing everything. On the XSTest safe split (Röttger et al., 2024), the locked models are not over-refusing: at our operating lock rate, safe-split compliance is within two points of the aligned base model on three of four models; on Qwen3-32B it drops by 6.8 pp, which Lock 5 restores (Appendix C.5, Table 11). On a wider zero-shot battery, locking protects TruthfulQA, which fine-tuning erodes, but trails ordinary fine-tuning by up to 3.1 points on the remaining tasks on three of four models, mostly on GSM8K (Appendix C.5).

5 Conclusion

Safety alignment in a low-rank adapter is concentrated, not distributed. Across three models from two families, one tenth of the adapter's coordinates carries nearly all of the safety-sensitive energy. Holding those coordinates fixed during fine-tuning performs comparably to or better than three published defenses, at a higher task loss, while requiring nothing of the downstream user beyond attaching the mask. An overlap bound shows that at heavy lock, any two masks are nearly identical in what they lock, so the difference between targeted and random selection is determined entirely by which coordinates each leaves free to train. Targeting controls confirm that safety follows the trainable set. Separately, coordinate concentration and spectral spread come apart across architectures, and spread, not concentration, tracked which model needed its curvature estimate refreshed across a task sequence.

AI USE STATEMENT

We used generative AI tools in three ways. For writing, we used them to polish the language of the paper. For retrieval and discovery, we used them to search for relevant papers, find related work, and summarize the key content of prior work. For research and ideation, we used them to discuss the research idea and identify gaps in it, to plan experiments, to assist in writing code, to help test hypotheses, and to implement additional ablation studies. Every AI-assisted contribution was reviewed by the authors: suggested related work was checked against the original papers, AI-assisted code was read and tested by the authors before use, and hypotheses and ablations were confirmed by the authors through experiments we ran ourselves. All numbers reported in the paper come from those experiments. We take full responsibility for the final content of this work, including text, claims, code and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work aims to reduce a safety risk: the loss of refusal behavior when an aligned model is fine-tuned on benign data. It involves no human subjects and no personal data. Building and evaluating the patch uses public harmful-prompt benchmarks (HEx-PHI, AdvBench, HarmBench, XSTest) under their intended research use, and the fine-tuned models necessarily produce some harmful completions during evaluation; we do not release these completions. The measurement is dual-use in one respect: it identifies which adapter coordinates carry the safety signal, which an adversary who controls fine-tuning could try to target. Our threat model is cooperative, so the patch offers no protection against such an adversary, as stated in the Limitations, and we do not claim otherwise. We declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

Algorithm 1 specifies the build and deploy phases of the patch, and §4.1 gives the models, datasets and evaluation protocol. Appendix D.2 lists all hyperparameters, seeds and compute, Appendix D.1 describes the L-BFGS eigenvector extraction, Appendix D.3 describes the contents of the patch artifact, and Appendix C.1 gives the single declared configuration of each baseline together with the criterion used to choose it. Appendix C.6 documents the keyword scorer and the classifiers used to validate it. All datasets and base models are publicly available. We will release the source code and the four model-specific safety patches upon acceptance.

REFERENCES

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Thong Bach, Dung Nguyen, Thao Minh Le, and Truyen Tran. Continual safety alignment via gradient-based sample selection. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026a.

Thong Bach, Thanh Nguyen-Tang, Dung Nguyen, Thao Minh Le, and Truyen Tran. Curvature-aware safety restoration in LLMs fine-tuning. *Transactions on Machine Learning Research (TMLR)*, 2026b.

Sahil Chaudhary. CodeAlpaca: An instruction-following LLaMA model for code generation. https://github.com/sahil280114/codealpaca, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

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://github.com/databrickslabs/dolly, 2023.

Amitava Das, Abhilekh Borah, Vinija Jain, and Aman Chadha. AlignGuard-LoRA: Alignment-preserving fine-tuning via fisher-guided decomposition and riemannian-geodesic collision regularization, 2025.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, et al. A framework for few-shot language model evaluation. https://github.com/EleutherAI/lm-evaluation-harness, 2024.

Luxi He, Mengzhou Xia, and Peter Henderson. What is in your safe data? identifying benign data that breaks safety, 2024.

Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun-Ying Huang. Safe LoRA: The silver lining of reducing safety risks when fine-tuning large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Lazy safety alignment for large language models against harmful fine-tuning attack. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Tiansheng Huang, Sihao Hu, and Ling Liu. Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Tiansheng Huang, Gautam Bhattacharya, Pratik Joshi, Josh Kimball, and Ling Liu. Antidote: Post-fine-tuning safety alignment for large language models against harmful fine-tuning. In *International Conference on Machine Learning (ICML)*, 2025a.

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. In *International Conference on Learning Representations (ICLR)*, 2025b.

Jiaming Ji et al. Language models resist alignment: Evidence from data compression, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13): 3521–3526, 2017.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017.

Mingjie Li et al. SaLoRA: Safety-alignment preserved low-rank adaptation. In *International Conference on Learning Representations (ICLR)*, 2025.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, 1989.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International Conference on Learning Representations (ICLR)*, 2024.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-Factored approximate curvature. In *International Conference on Machine Learning (ICML)*, 2015.

Mantas Mazeika et al. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *International Conference on Machine Learning (ICML)*, 2024.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on Health, Inference, and Learning*, 2022.

ShengYun Peng, Pin-Yu Chen, Matthew Hull, and Duen Horng Chau. Navigating the safety landscape: Measuring risks in finetuning large language models, 2024.

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 (ICLR)*, 2024.

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. In *International Conference on Learning Representations (ICLR)*, 2025.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

Shen Shen, Sicheng Yao, Jun Zhang, et al. Safety layers in aligned large language models: The key to LLM security. In *International Conference on Learning Representations (ICLR)*, 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 (ICLR)*, 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. https://github.com/tatsu-lab/stanford_alpaca, 2023.

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. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.05162.

Biao Yi, Jiahao Li, Baolei Zhang, Lihai Nie, Tong Li, Tiansheng Huang, and Zheli Liu. Gradient surgery for safe llm fine-tuning. *arXiv preprint arXiv:2508.07172*, 2025.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

A Related Work

Where safety lives in the model. A series of results has progressively localized safety-relevant structure in language models. Qi et al. (2024) showed that as few as ten adversarial examples raise attack success from single digits to over 50%, and He et al. (2024) extended this to benign data, establishing that alignment is fragile under fine-tuning. Three geometric accounts of this fragility followed: a safety basin with sharp boundaries (Peng et al., 2024), an elastic reversion toward pre-trained distributions (Ji et al., 2024), and alignment concentrated in the first few generated tokens (Qi et al., 2025). A parallel line localizes safety in the parameters themselves: Wei et al. (2024) find safety-critical neurons and ranks that are sparse and separable from utility-relevant ones, and Shen

et al. (2025) identify safety layers and freeze them during fine-tuning. This paper adds the finest level of description, the individual coordinates of the adapter that downstream fine-tuning actually updates, and Appendix B.3 and §4.3 show that the layer-level and coordinate-level descriptions do not reduce to each other. Wei et al. (2024) also report that restricting modification of safety-critical regions does not prevent adversarial removal of alignment, which addresses a different threat model from ours: their adversary controls the training procedure, while the patch here protects a cooperative user from destroying alignment by accident on benign data.

Defense families. Existing defenses against safety degradation during fine-tuning fall into three categories. Projection methods identify an alignment-sensitive subspace and constrain the update to avoid it, differing in how the subspace is estimated: an aligned-minus-base weight difference (Hsu et al., 2024), a singular value decomposition of harmful-prompt activations (Li et al., 2025), or a runtime safety gradient (Yi et al., 2025). Regularization methods modify the alignment stage so the released checkpoint resists subsequent fine-tuning (Huang et al., 2024b;a; 2025b; Rosati et al., 2024; Tamirisa et al., 2025), changing the released weights where our patch leaves the base model untouched. Data-centric methods filter or reweight samples inside the user's training loop; Bach et al. (2026a) establish the four-task continual-safety benchmark we adopt in §4.4.

Positioning against closest work. AlignGuard-LoRA (Das et al., 2025) is the closest method: it identifies an alignment-sensitive subspace by Fisher importance and constrains updates accordingly, differing from our approach on two axes (the importance score and the mechanism), of which we isolate the score in §4.3 rather than reporting only the composite difference. The broader contrast is that projection methods take the subspace as given and ask how to avoid it, while this paper first measures whether avoiding it is necessary. SPPFT (Shen et al., 2025) is closest on mechanism: it also holds parameters fixed rather than projecting, at layer granularity, and §4.3 compares the two allocation granularities at matched budget. Bach et al. (2026b) share the L-BFGS curvature machinery but apply it in the opposite direction, repairing safety after fine-tuning has damaged it rather than preventing the damage, and their object of study is the repair rather than the coordinate-level distribution. Here the operator is built once, before any fine-tuning, and used only to measure how safety distributes over coordinates. The deployed intervention is a fixed binary mask that needs no curvature at training time, and in its Fisher variant no curvature at all.

Technical ancestry. The Hadamard form $|g \odot H^{-1}g|$ originates in influence-function analysis (Koh & Liang, 2017). The Fisher information matrix (Amari, 1998; Martens, 2020) underlies natural-gradient optimization (Martens & Grosse, 2015; Liu et al., 2024) and Elastic Weight Consolidation (Kirkpatrick et al., 2017), which uses per-parameter Fisher importance to protect previously learned tasks during continual learning. §4.3 measures how much the curvature reweighting separating the Hadamard score from the Fisher diagonal changes the resulting mask. Mazeika et al. (2024) supply the red-teaming protocol whose judging convention we follow.

B Extended Measurement Analysis

B.1 Eigenvector separation

This diagnostic asks whether the curvature eigenvectors split into directions relevant mainly to safety and directions relevant mainly to a downstream task. For each eigenvector q_i we compute the selectivity $s_i = (q_i^\top g_{\text{forget}})^2/(q_i^\top g_{\text{retain}})^2$, with both gradients on held-out probe batches and g_{retain} on Alpaca. Eigenvectors with $s_i > 1.5$ are classified as SAFETY, those with $s_i < 2/3$ as TASK, and the band between as shared. Unlike every operator in the main text, the L-BFGS history here is built with each descent direction projected onto the orthogonal complement of a reference retain gradient, so part of the separation is induced by the construction. Figure 4 visualizes this classification on Qwen2.5-7B. Each point is one eigenvector, plotted by its influence on the forget loss against its influence on the retain loss. Of the 38 eigenvectors, 24 fall above the diagonal (safety-biased), 10 below it (task-biased) and 4 in the shared band, though many lie close to the diagonal. The top safety eigenvector reaches a selectivity of $613\times$, meaning its projection onto the forget gradient is more than six hundred times larger than its projection onto the retain gradient. The separation is therefore carried by a minority of strongly safety-biased directions. We have no retain-versus-retain

null to calibrate the thresholds, and we report the figure as a diagnostic, not as evidence about the forget-only operator used in §2.2.

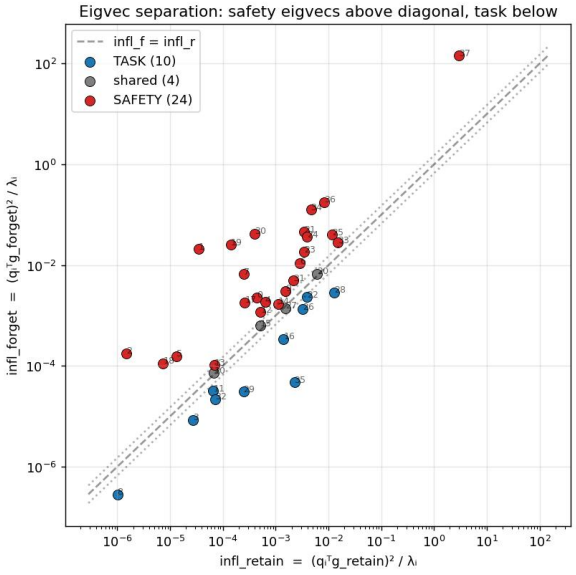


Figure 4: The eigenspectrum separates into safety- and task-biased directions on Qwen2.5-7B. Each point is one L-BFGS eigenvector, plotted by its influence on the forget loss (y-axis) against the retain loss (x-axis). Points above the diagonal are safety-biased, below are task-biased. Of the 38 eigenvectors, 24 are classified as SAFETY, 4 as shared, and 10 as TASK. The history is built with descent directions orthogonalized against a retain gradient, unlike the forget-only operator used in the main text.

B.2 Coverage of the safety subspace by the mask

This appendix verifies that the mask produced by Algorithm 1 captures the safety subspace it targets, and examines why coverage alone does not predict behavioral outcomes.

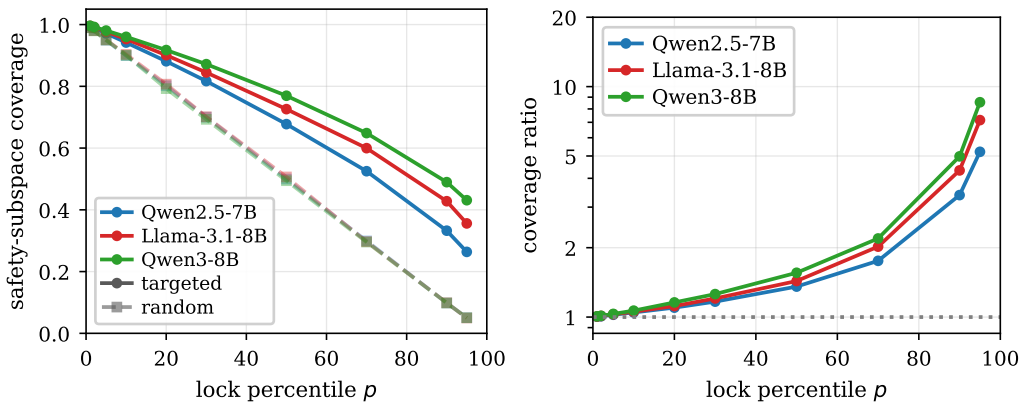
For a binary mask M and safety eigenvectors $q_1, \ldots, q_K$, define the coverage as

$$\text{coverage}(M) \;=\; \frac{1}{K}\sum_{i=1}^{K} \|M \odot q_i\|^2,$$

which measures the average fraction of each eigenvector's L^2 mass that falls on masked coordinates. Since each q_i is unit-norm, coverage ranges from 0 (the mask misses the subspace entirely) to 1 (the mask contains every coordinate the subspace touches).

Figure 5 sweeps coverage across lock percentile on three models. At the operating point $p = 10$, the targeted mask captures 0.94 to 0.96 of safety-subspace energy, confirming that the construction succeeds in its stated goal. A size-matched random mask captures 0.90, giving a ratio of only $\sim 1.05\times$. This narrow gap is a direct consequence of Proposition 1: at 90% lock, any two masks must share at least 80% of their coordinates, so their coverage values are forced close together regardless of how they were constructed.

Coverage therefore serves as a sanity check on the mask construction rather than as a predictor of behavior. The targeted and random masks that differ by $1.05\times$ in coverage differ by several-fold in attack success (§4.3), confirming that what determines safety is not how much of the subspace a mask covers but which coordinates it leaves free to move during training. The coverage ratio rises to 5 to $10\times$ only at much lighter lock rates, where the overlap bound no longer forces the two masks to share most of their coordinates.



(a) Absolute coverage

(b) Ratio over random

Figure 5: Coverage confirms the mask construction without predicting behavior. (a) Safety-subspace coverage against the lock percentile p, for the targeted mask (solid, circles) and a size-matched random mask (dashed, squares), coloured by model. Both capture most of the subspace's energy at the operating point ($p = 10$), and the three random curves nearly coincide. (b) The coverage ratio, coverage(targeted) / coverage(random): near 1.0 at heavy lock and rising to 5–$10\times$ only at light lock, where the overlap bound no longer forces the two masks to share most of their coordinates. In both panels p is the percentage of coordinates left trainable, so locking is heaviest at the left-hand edge.

B.3 Depth: per-layer detail and a layer-aware budget

The mask derived in §3 locks the same fraction of every adapter tensor, so it is layer-uniform by construction. Whether the safety signal also concentrates in specific layers is therefore something we measure rather than something we built in, and the answer determines whether a simpler layer-level mask could achieve the same result.

Safety energy concentrates in layers; the safety decile does not. Aggregating the per-coordinate safety membership $m(c)$ over each decoder layer reproduces the layer-level finding from prior work (Wei et al., 2024; Shen et al., 2025): safety energy is markedly non-uniform with depth, with the top three layers carrying several times their expected share on every model (Table 4). The safety decile tells a different story. Taking the top tenth of coordinates by $m(c)$ network-wide and measuring their distribution over depth, we find that they spread nearly uniformly across every layer of every model. The median enrichment (a layer's share of decile coordinates divided by its share of all coordinates) is close to 1.0 on all three models, and no layer of any model is empty of decile coordinates (Figure 6). These two observations are consistent: a layer can hold a large share of safety energy while holding a roughly representative share of the decile, because its energy is carried by a few extreme coordinates rather than by many moderately important ones. The practical consequence is that freezing whole layers is not equivalent to freezing the safety decile: since the decile is present in every layer, a layer-level mask must either release safety-critical coordinates or freeze nearly the entire adapter.

Per-layer aggregates. Table 4 breaks the two quantities down by model. The energy share column confirms the layer-level non-uniformity. The decile enrichment columns confirm the uniformity. The table also reveals a model-specific divergence between safety energy and Hadamard importance. On Qwen2.5-7B and Llama-3.1-8B the two layer-level profiles agree closely. On Qwen3-8B they diverge: importance concentrates 53.2% of its mass in three layers against safety energy's 24.7%. The layer mask of §4.3 ranks layers by importance mass, which is why on Qwen3-8B it freezes by a signal that does not track safety energy and frees the four earliest layers rather than late ones. This mismatch is what causes the layer mask to fail on that model (Table 7).

The decile released by a mask does not predict behavior. A natural way to compare masks is to measure how much of the safety decile each one leaves trainable. At the operating point, the coordinate mask releases essentially none of the decile, the layer mask releases 4.0 to 7.0%

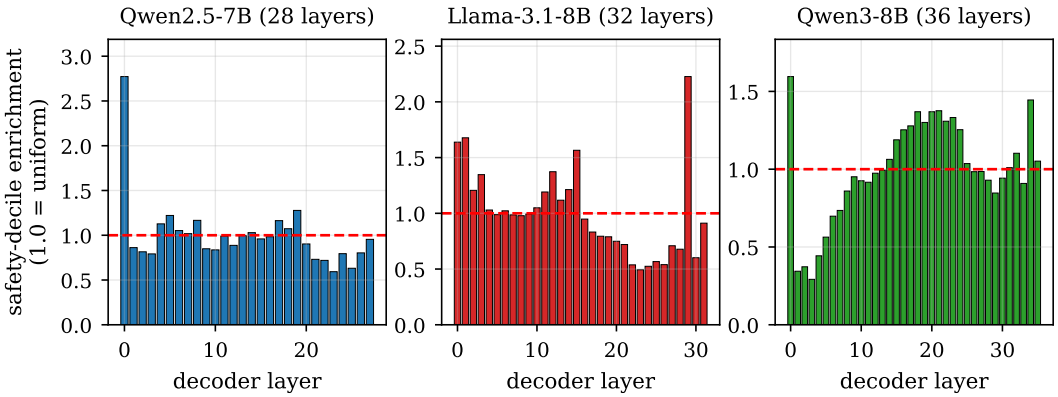


Figure 6: The safety decile is spread across depth even where safety energy is not. Per-layer enrichment of the top tenth of coordinates by $m(c)$: a layer's share of those coordinates divided by its share of all coordinates, so 1.0 (dashed) is uniform.

Table 4: Safety energy is layer-concentrated; the safety decile is not. Per-layer aggregates of safety membership $m(c)$ and Hadamard importance, with $K = 20$ and forget-only L-BFGS at fresh adapter initialization. "Top-3" is the fraction of total mass in the three highest-ranked layers. "Max layer" is the ratio of the single highest layer's mass to the uniform expectation. Enrichment is a layer's share of the global top-decile coordinates divided by its share of all coordinates.

		Safety energy $m(c)$		Importance mass		Decile enrichment	
Model	Layers	top-3	max layer	top-3	max layer	min / median	max
Qwen2.5-7B	28	38.3%	7.1×	27.7%	3.8×	0.59 / 0.96	2.77
Llama-3.1-8B	32	57.4%	12.4×	46.5%	6.9×	0.49 / 0.98	2.23
Qwen3-8B	36	24.7%	3.3×	53.2%	14.1×	0.29 / 0.99	1.60

depending on the model, and a size-matched random mask releases roughly 10%. If decile release predicted behavior, the layer mask should always fall between the coordinate mask and the random mask. It does not: the layer mask that releases the least of the decile (Qwen3-8B, 4.0%) is the one that fails catastrophically. This is the same pattern found for coverage in Appendix B.2 and for the inversion control in Appendix C.2: metrics based on how much safety energy a mask retains consistently fail to predict what the mask does, because behavior follows the trainable set rather than the locked set.

A layer-aware budget. Because safety energy is layer-concentrated while our per-tensor rule spends the lock budget uniformly, the budget allocation itself is an axis we have not optimized. As a first probe we kept the total number of locked coordinates fixed and the same Hadamard ranking within each tensor, but allocated each layer's share of the budget in proportion to its importance mass rather than its parameter count. The results are mixed across models and we draw no conclusion about which allocation is superior. What the probe does establish is that the distribution of the budget across layers is a live axis of the design space, unlike the choice of importance score §4.3), which barely changes the mask at heavy lock.

C Extended Experimental Results

C.1 Baseline comparison: extended analysis

This appendix provides configuration details for the three published baselines in Table 1 and reports aspects of the comparison that the main text summarizes.

Baseline configurations. Each baseline runs at a single declared configuration across all models. We avoid per-model sweeps because reporting the best cell from a sweep is selection on the evaluation metric, which inflates the baseline's apparent effectiveness.

SafeLoRA (Hsu et al., 2024) projects a fixed fraction ($30/64$) of candidate layers, giving 30, 26, or 34 projected layers depending on architecture depth. We fix the fraction rather than the layer count because a fixed integer would apply three different intervention strengths under one name. AlignGuard-LoRA (Das et al., 2025) constrains 25% of coordinates, computed per model from its cached Fisher before training. We fix the constrained fraction rather than the regularization weight λ because λ multiplies a model-dependent Fisher and is not comparable across architectures. Antidote (Huang et al., 2025a) runs at sparsity 0.03, which we selected as the largest sparsity whose attack success remains above the base-model floor and passes the over-refusal audit of Appendix C.5. The authors' published values are higher (0.1 in their main script, 0.05 and 0.2 elsewhere). We report the configuration we ran and the criterion that produced it, and make no claim about settings we did not test. SafeLoRA requires the pre-alignment checkpoint, which is publicly available for Llama-3.1-8B and Qwen2.5-7B but not for Qwen3-32B; the missing cell in Table 1 reflects this.

Task cost of comparable safety. The patch pays for its safety in task loss: eval loss is the highest of the four methods on all four models, exceeding the undefended adapter by 0.05 to 0.20, and utility is the lowest or second-lowest on three of four models. What the patch gains in return is markedly lower variance across seeds. Unmasked fine-tuning varies by ± 1.2 to ± 5.3 pp in attack success at fixed configuration, with Qwen3-8B spanning 28.6 to 38.8%. The patch is more stable on the three models where that variance is largest (± 0.5 against ± 1.8, ± 1.3 against ± 4.8, ± 0.5 against ± 5.3) and marginally worse on Qwen3-32B (± 1.5 against ± 1.2).

Cross-benchmark transfer. The method ranking is stable across the two benchmarks on Qwen2.5-7B and Llama-3.1-8B. On the two Qwen3 models the leading two methods swap between AdvBench and HarmBench at margins within or near the seed spread, so we treat them as comparable rather than ordered. Defining recovery as $(\text{undefended} - \text{method})/(\text{undefended} - \text{base})$, every defense including ours recovers substantially less of the fine-tuning damage on HarmBench than on AdvBench on the two models whose base-model floors the keyword and classifier scorers agree on (Qwen2.5-7B and Llama-3.1-8B). On the two Qwen3 models the recovery fraction is not interpretable because its denominator depends on the keyword-scored base-model floor, which Appendix C.6 shows to be the least reliable quantity in the table on exactly those models. Our forget set, AlignGuard's Fisher, and Antidote's pruning calibration all derive from the same HEx-PHI prompts, so a gap of this shape most likely reflects a general limitation of forget-set-calibrated defenses rather than a shortcoming specific to any one method.

Task agnosticism. Table 5 confirms that the mask does not depend on the downstream data. Building the patch once on Qwen2.5-7B from HEx-PHI and applying it unchanged to three downstream tasks (Dolly, Alpaca, CodeAlpaca) drives attack success to at most 3% on all three at Lock 10, with no rebuild between tasks. The eval loss increase is consistent across tasks, ranging from 0.03 to 0.07 at Lock 10, confirming that the mask imposes a uniform capacity cost rather than interfering with any particular task's learning signal.

Table 5: One patch, built with no task data, holds across three downstream tasks. Cross-task evaluation on Qwen2.5-7B; the same patch built once on HEx-PHI is applied to each task with no rebuild.

Task	Baseline		Lock 10		Lock 5	
	Eval Loss	ASR	Eval Loss	ASR	Eval Loss	ASR
Dolly	1.49	20.2%	1.55	3.0%	1.72	0.6%
Alpaca	1.19	17.8%	1.26	2.8%	1.47	1.2%
CodeAlpaca	0.44	8.0%	0.47	1.8%	0.56	1.0%

Operating point and its neighbourhood. Sweeping the lock percentile on Qwen2.5-7B (Figure 7), attack success falls monotonically from 22.2% at no lock to 0.4% at $p = 2$, while eval loss stays flat near 1.49 down to $p = 10$ and then rises sharply to 2.19 at $p = 2$. The elbow at $p = 10$

defines our operating point: it is where the marginal safety gain per unit of task cost begins to diminish. Across all four models, tightening the lock to $p = 5$ buys a further 1.5 to 4.6 pp of attack success reduction and comes within 0.6 pp of the base-model floor or below it everywhere, at up to 0.19 additional eval loss (Table 6).

Table 6: Heavier locking buys further safety at rising task cost. Lock percentile across four base models on Dolly-15k. Cells marked † are single-seed.

Model	Base	No lock		Lock 10		Lock 5	
	ASR	Eval Loss	ASR	Eval Loss	ASR	Eval Loss	ASR
Qwen2.5-7B	0.6	1.55	20.1 ± 1.8	1.58	3.1 ± 0.5	$1.59^{\dagger}$	$0.8^{\dagger}$
Llama-3.1-8B	3.8	1.57	59.9 ± 4.8	1.59	6.1 ± 1.3	$1.57^{\dagger}$	$3.4^{\dagger}$
Qwen3-8B	1.2	1.61	34.5 ± 5.3	1.72	3.3 ± 0.5	$1.91^{\dagger}$	$1.8^{\dagger}$
Qwen3-32B	3.0	1.36	13.2 ± 1.2	1.56	4.6 ± 1.5	$1.67^{\dagger}$	$0.0^{\dagger}$

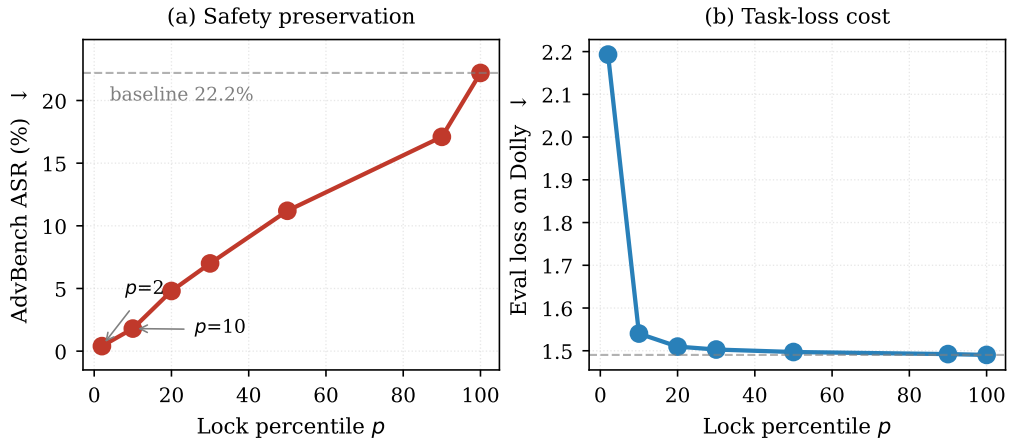


Figure 7: The safety-task trade-off has a clear elbow at the operating point. Lock-percentile sweep on Qwen2.5-7B with Dolly, where p is the percentage of coordinates left trainable, so locking is heaviest at the left. (a) Attack success falls monotonically with heavier locking, from the undefended baseline (22.2%, dashed) down to 0.4% at $p = 2$. (b) Eval loss stays flat near 1.49 down to $p = 10$ and then rises sharply, which is what fixes $p = 10$ as the operating point.

C.2 Targeting controls: full analysis

Section 4.3 reports that four alternative explanations for the patch's effectiveness all fail. This appendix provides the full analysis behind each control, establishes what the four collectively rule out, and reports the lock sweep that traces the targeted-vs-random gap across the full range of lock rates.

Explanation 1: the patch works because it reduces adapter capacity. Locking 90% of coordinates leaves 10% trainable, so an unmasked adapter at rank $r = 3$ (9.4% of the $r = 32$ parameter count, with α scaled to hold α/r fixed) has matched capacity. If the patch's safety gain came from capacity reduction alone, the reduced-rank adapter should reproduce it. It does not. On Llama-3.1-8B, $r = 3$ leaves attack success at 39.4% while Lock 10 reaches 6.0% at comparable eval loss. Reducing rank helps monotonically on that model ($51.0 \rightarrow 44.0 \rightarrow 39.4$) and still ends several times worse than the targeted mask. On Qwen3-8B, reducing rank does not help at all: both $r = 8$ and $r = 3$ leave attack success within the model's own seed spread of the undefended baseline, while Lock 10 reaches 4.0%. On Qwen2.5-7B reducing rank helps more (7.8% at $r = 3$), but still leaves attack success about four times that of the targeted mask. Restricting how much the adapter can move is not the same as restricting which coordinates it moves.

Explanation 2: any heavy lock works, regardless of which coordinates it holds. We test this by inverting the importance ranking: the lock rate is identical, the mask is constructed from the same

score, and the only change is that the coordinates our score identifies as most safety-critical are the ones left free rather than locked. If the identity of the locked coordinates did not matter, the inverted mask should perform comparably to the targeted one. Safety degrades relative to the targeted mask on every model. Attack success rises by 20.6 pp on Llama-3.1-8B, 5.2 pp on Qwen2.5-7B, and 2.8 pp on Qwen3-8B, all at unchanged lock rate and comparable eval loss. The direction is consistent across all three models, though only the Llama effect is large relative to seed noise. The ranking produced by the importance score is causally load-bearing, but reversing it does not simply reverse the outcome: on Qwen2.5-7B and Qwen3-8B the inverted mask is still safer than a random one, a pattern we return to below.

Explanation 3: a size-matched random mask performs equally well. The overlap bound (Proposition 1) makes this the most plausible of the four alternatives, since a random mask locking nine tenths of the coordinates already contains most of the safety decile and captures 0.90 of safety-subspace energy against the targeted mask's 0.94 to 0.96 (Appendix B.2). Despite this near-identical coverage, the random mask leaves substantially higher attack success on every model tested. Expressed as recovery of the fine-tuning damage, the random mask repairs 40 to 65% where the targeted mask repairs 89 to 95%. The gap arises because the two masks, while nearly identical in what they lock, leave almost entirely different coordinates free to train (IoU 0.06 on trainable coordinates against 0.82 on locked coordinates on Qwen2.5-7B), and it is the trainable set that determines safety behavior. The lock sweep on Qwen3-32B (Figure 8) shows the same pattern as a curve rather than a single operating point: the targeted mask descends monotonically to the base-model floor while the random mask flattens well above it.

Explanation 4: a layer-granular mask achieves the same result more cheaply. Safety has been localized to layers before (Shen et al., 2025), and Appendix B.3 confirms that safety energy is non-uniform with depth in our own curvature analysis. We construct a layer-granular mask from the same importance score, ranking layers by total importance mass and freezing whole adapter tensors in the highest-ranked layers until the budget is spent. Table 7 reports the comparison. On Qwen2.5-7B and Llama-3.1-8B, the layer mask matches the coordinate mask's attack success at higher eval loss (0.08 to 0.09 more), suggesting that the coarser allocation pays a task cost for comparable safety. On Qwen3-8B, the layer mask does not reach comparable safety at all: it more than doubles the undefended attack success on every seed. The failure is structural rather than statistical. On that model, the importance-ranked layer ordering frees the four earliest layers, whereas on the other two models it frees late layers. Confining the entire update to four early layers damages alignment more than leaving the adapter unconstrained, which is why the layer mask performs worse than no defense on that architecture.

Table 7: Layer-granular allocation is not a cheaper route to the same result. Coordinate-granular against layer-granular masks at matched lock budget, three seeds each. Attack success on AdvBench (↓); eval loss in parentheses.

Model	Undefended	Layer mask	Coordinate mask
Qwen2.5-7B	20.1 ± 1.8 (1.55)	3.0 ± 0.2 (1.66)	3.1 ± 0.5 (1.58)
Llama-3.1-8B	59.9 ± 4.8 (1.57)	4.9 ± 3.0 (1.68)	6.1 ± 1.3 (1.59)
Qwen3-8B	34.5 ± 5.3 (1.61)	61.7 ± 11.3 (1.65)	3.3 ± 0.5 (1.72)

What the four controls establish collectively. Each control removes one candidate explanation: matched capacity rules out a pure capacity effect, inversion rules out score-independent locking, the random mask rules out coverage-based accounts, and the layer mask rules out layer-level sufficiency. Together they establish that the patch's effectiveness depends on locking specific coordinates identified by the importance score, and that no coarser description of the mask (its size, its coverage, or its layer allocation) accounts for the behavioral outcome.

A single pattern runs through all four controls. Every natural way of scoring a mask by how much safety energy it retains fails to predict what the mask does. Coverage separates the targeted and random masks by $1.05\times$ where their behavioral gap is several-fold. The inverted mask frees the entire safety decile and the random mask a tenth of it, yet the inverted mask is safer on two of three models. The layer mask that releases the least of the decile is the one that fails catastrophically.

In every case, the explanation is the same: an update moves through the trainable coordinates, and behavior follows those coordinates rather than the locked ones. This is the asymmetry that Corollary 1 states formally and that the four controls confirm empirically.

Lock sweep. Figure 8 traces the targeted and random masks across the full range of lock rates on Qwen3-32B. The targeted mask descends monotonically across the entire sweep and reaches the base-model floor at heavy lock. The random mask declines modestly at first and flattens well above the floor, never reaching base-model safety regardless of how much of the adapter it locks. The divergence between the two curves is the behavioral signature of targeted selection: both masks lock the same fraction of coordinates, and their curves separate because they free different ones.

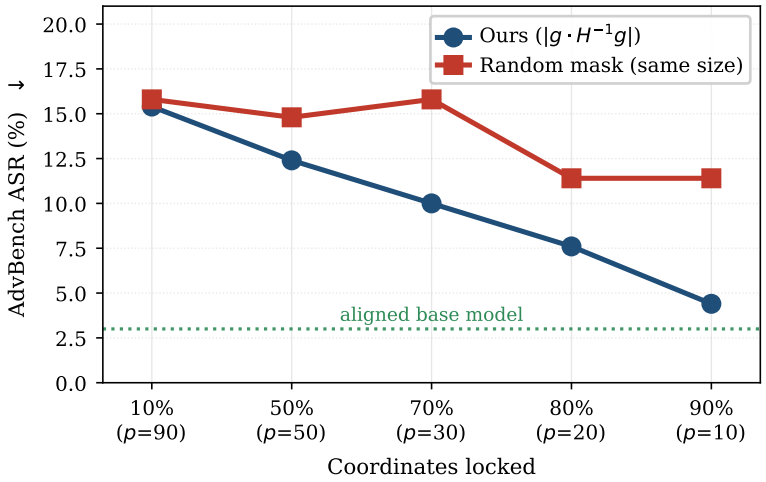


Figure 8: Only the targeted mask converts heavier locking into base-model-level safety. Qwen3-32B lock sweep (seed 42): the targeted mask (blue circles) against a size-matched random mask (red squares), with the percentage of coordinates locked increasing to the right. The dotted line marks the aligned base model's attack success (3.0%). The targeted mask descends monotonically towards that floor; the random mask flattens well above it, so the two masks separate even though they lock the same fraction of the adapter.

C.3 Score-formula ablation: extended results

Section 4.3 reports that replacing the Hadamard influence score with the Fisher diagonal produces a nearly identical mask safety outcome depends on training pressure. This appendix provides the full comparison across all available (model, lock percentile) combinations and tests whether softening the binary mask improves the safety-task trade-off.

Score-formula comparison. Table 8 reports attack success under three scoring formulas: the Hadamard influence $|g \odot H^{-1}g|$, the Fisher diagonal $|g|^2$, and the raw gradient magnitude $|g|$. All cells use seed 42 so that the arms are exactly paired, meaning each comparison isolates the effect of the score formula with all other sources of variation held fixed. The Fisher diagonal and gradient magnitude produce byte-identical masks on every model, because the two formulas are rank-monotonic transforms of each other and mask construction depends only on rank. This confirms that curvature reweighting is the sole difference between the Hadamard score and its alternatives.

At the operating point ($p = 10$) under the heavy training configuration, the Hadamard score produces lower attack success than the Fisher/gradient score on all four models. The margin ranges from 0.8 pp on Qwen2.5-7B to 4.4 pp on Qwen3-8B. Under the light configuration at the same lock rate, the three scores give identical attack success on Qwen2.5-7B, and at lighter lock rates the ordering is mixed. C3 predicts the mask overlap (IoU 0.978) but not this behavior: under heavy training, the small set of coordinates on which the two masks disagree is enough to produce a consistent difference. The gap therefore tracks training pressure rather than lock rate, in line with the finding

that behavior follows the trainable set. In the continual setting the order reverses on Llama-3.1-8B (Appendix C.4), so we do not treat either score as uniformly better.

For context on the magnitude of these differences: the seed-to-seed standard deviation of the Lock 10 cells in Table 1 ranges from 0.5 to 1.3 pp, so individual rows of Table 8 should not be over-interpreted. The conclusion rests on the consistency of the direction across models at the operating point, not on any single cell.

Table 8: Score-formula ablation across models and lock rates. Mask overlap (IoU) between the Fisher/gradient mask and the Hadamard mask is 0.978 on Qwen2.5-7B at $p = 10$. All cells seed 42 for exact pairing. Fisher and gradient masks are byte-identical on every model.

Model	Config	Score	ASR (%)
Qwen2.5-7B	T1 Lock 10 (3 ep, bs=8)	Hadamard / Fisher / Grad	2.0 / 2.8 / 2.8
Qwen2.5-7B	p=10 light (1 ep, bs=4)	Hadamard / Fisher / Grad	1.8 / 1.8 / 1.8
Qwen2.5-7B	p=50 light	Hadamard / Fisher / Grad	10.6 / 10.0 / 10.0
Qwen2.5-7B	p=90 light	Hadamard / Fisher / Grad	15.0 / 16.2 / 16.2
Llama-3.1-8B	T1 Lock 10 (3 ep, bs=8)	Hadamard / Fisher / Grad	6.0 / 8.0 / 8.0
Llama-3.1-8B	p=50 light	Hadamard / Grad	15.0 / 15.0
Qwen3-8B	T1 Lock 10 (3 ep, bs=8)	Hadamard / Fisher / Grad	4.0 / 8.4 / 8.4
Qwen3-32B	T1 Lock 10 (3 ep, bs=8)	Hadamard / Fisher / Grad	1.2 / 4.6 / 4.6

Softening the mask. The main experiments use a hard binary mask that sets the gradient to exactly zero on locked coordinates. A natural question is whether allowing a small residual gradient on those coordinates would recover task capacity without sacrificing safety. We test this by replacing the hard mask with a soft scale: locked coordinates receive a gradient multiplied by $\sigma \in \{0, 10^{-4}, 10^{-3}, 10^{-2}\}$, where $\sigma = 0$ is the hard mask used throughout the paper. On Qwen2.5-7B at the $p = 10$ operating point, attack success rises monotonically with σ, from 1.8% at $\sigma = 0$ to 11.4% at $\sigma = 10^{-2}$, while eval loss falls only from 1.54 to 1.50. Even the smallest nonzero scale ($\sigma = 10^{-4}$) raises attack success measurably, and by $\sigma = 10^{-2}$ the safety gain over the random mask has been largely erased. Softening the mask leaks safety faster than it returns task capacity, which is why the main experiments use the hard mask exclusively.

C.4 Continual learning: extended results

Section 4.4 reports that the spectral spread diagnostic correctly predicts which models require curvature re-anchoring across a multi-task sequence. This appendix provides the full evaluation including per-task capability, the per-seed attack success trajectories that reveal the failure mode on the spectrally spread model, and a seed-level breakdown that isolates the residual failure the re-anchoring does not address.

Full evaluation with capability. Table 9 extends Table 3 in the main text by adding per-task capability scores and a Fisher-score comparison. The Hadamard and Fisher scores produce comparable safety outcomes on the two spectrally compact models (Qwen2.5-7B and Qwen3-4B), consistent with the score-formula ablation in Appendix C.3. On Llama-3.1-8B the Fisher score achieves slightly lower checkpoint-averaged attack success (4.3% vs. 5.4%) despite using the cheaper `rebuild` lifecycle rather than the full `rebuild-true` re-anchoring. This advantage likely reflects the fact that the Fisher score does not depend on H^{-1} and is therefore unaffected by staleness, which is the very problem the re-anchoring was introduced to solve. Per-task capability columns show that the safety gain is not achieved by sacrificing general knowledge: on every model, the locked adapter preserves capability within a point of the data-centric baseline on all benchmarks except TruthfulQA, where locking substantially outperforms the baseline, and ARC-C on Qwen2.5-7B, where it is 3.6 points lower.

Per-seed trajectory on the spectrally spread model. Figure 9 traces per-seed attack success on Llama-3.1-8B, the model that §2.5 flagged as spectrally spread, across the four-task continual sequence under two lifecycle policies. Under the `rebuild` lifecycle (red), which refreshes the forget gradient at each task boundary but retains the stale H^{-1} from initialization, seed 1 undergoes a

Table 9: Continual-learning evaluation with per-task capability. Attack success is checkpoint-averaged on AdvBench (↓, mean ± std over 3 seeds), and the capability columns are checkpoint-averaged zero-shot accuracies (↑). The lifecycle for each Hadamard row follows the §4.4 prescription: `rebuild-true` for Llama-3.1-8B (spectrally spread), `rebuild` for the other two (spectrally compact). Fisher rows use `rebuild` throughout, since the Fisher score has no H^{-1} to re-anchor.

Model	Method	ASR ↓	TQA	ARC-C	BoolQ	HellaSwag	WinoGr.
Qwen2.5-7B	Mod-G_i	10.2	42.5	59.6	86.3	79.9	71.7
	Ours, Fisher	1.0 ± 0.2	—	—	—	—	—
	Ours, Hadamard	0.8 ± 0.1	60.3	56.0	86.4	80.3	71.5
Llama-3.1-8B	Mod-G_i	18.3	41.5	56.0	84.8	77.9	73.7
	Ours, Fisher	4.3 ± 0.8	—	—	—	—	—
	Ours, Hadamard	5.4 ± 1.9	51.7	55.1	83.9	78.8	73.8
Qwen3-4B	Mod-G_i	6.0	42.8	59.2	84.6	70.1	68.5
	Ours, Fisher	1.0 ± 1.0	—	—	—	—	—
	Ours, Hadamard	0.3 ± 0.2	61.4	59.6	84.5	69.1	68.2

sharp safety collapse after the third task, with attack success rising from near zero to over 35%. Seeds 2 and 3 also degrade, though less dramatically. Under the `rebuild-true` lifecycle (blue), which refreshes both H^{-1} and the forget gradient at each task boundary, the seed-1 collapse is eliminated entirely and seed-3 improves by 13.6 pp. The two spectrally compact models (Qwen2.5-7B and Qwen3-4B, not shown) exhibit no such collapse under either lifecycle, consistent with spectral spread governing staleness. With Llama-3.1-8B as the only spectrally spread model, this cannot yet be separated from architecture.

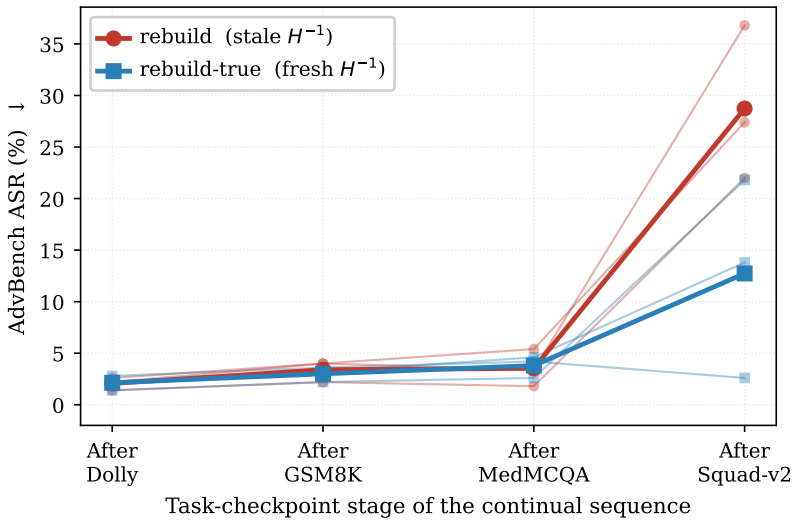


Figure 9: Re-anchoring the curvature estimate removes the safety collapse on the spectrally spread model. Llama-3.1-8B per-seed AdvBench attack success across the four-task continual sequence. Red: `rebuild` (stale H^{-1} from θ_0, fresh g_{forget} at θ_t). Blue: `rebuild-true` (fresh H^{-1} and fresh g_{forget} at θ_t). Thin lines show individual seeds, thick lines the seed mean. The two policies are indistinguishable for the first three tasks and separate at the fourth, where the worst seed under `rebuild` reaches 36.8% attack success against 2.6% under `rebuild-true`.

Seed-level breakdown and residual failure. Table 10 reports the final-stage (Squad-v2) attack success by seed under each lifecycle policy. The `rebuild-true` lifecycle reduces seed-1 attack success from 36.8% to 2.6% and seed-3 from 27.4% to 13.8%, confirming that the stale H^{-1} was the primary cause of their degradation. Seed 2 is the residual: re-anchoring leaves its attack success nearly unchanged (22.0% to 21.8%), indicating a failure mode that is not explained by curvature

staleness. The Fisher `rebuild` lifecycle, which avoids the staleness problem entirely by not using H^{-1}, achieves the lowest mean attack success (8.1%) and the lowest variance across seeds. Whether the seed-2 residual reflects an intrinsic limitation of coordinate locking on long task sequences or an artifact of the specific task ordering is an open question that future work should investigate.

Table 10: Per-seed final-stage (Squad-v2) attack success on Llama-3.1-8B. Re-anchoring the curvature estimate removes the seed-1 catastrophic failure and reduces the seed-3 transient, but leaves seed-2 as a residual that staleness does not explain.

Lifecycle	seed 1	seed 2	seed 3	mean ± std
`frozen`	40.2	0.6	0.6	13.8 ± 22.9
`rebuild`	36.8	22.0	27.4	28.7 ± 7.5
`rebuild_true`	**2.6**	21.8	**13.8**	$\mathbf{12.7 \pm 9.6}$
Fisher `rebuild`	7.4	13.4	3.6	8.1 ± 4.9

C.5 Over-refusal and capability audit

A defense evaluated only on harmful prompts can score well by refusing everything. This appendix verifies that the patch does not achieve its safety gains through blanket refusal, and that general capability is preserved.

Over-refusal. Table 11 reports safe-split compliance and unsafe-split refusal on XSTest (Röttger et al., 2024). On three of four models, safe-split compliance at Lock 10 is within two points of the aligned base model, confirming that the patch does not introduce meaningful over-refusal. The exception is Qwen3-32B, where Lock 10 drops safe-split compliance by 6.8 pp. This is also the model whose HarmBench attack success (21.3%) falls below the keyword-scored base-model floor (23.5%), and the two observations are related: part of the apparent sub-floor result reflects the over-refusal the audit detects, and part reflects a scoring artifact (the classifier-scored floor is 11.5% rather than 23.5%; see Appendix C.6). Tightening the lock to $p = 5$ restores safe-split compliance to the base-model level on that model.

The unsafe split provides an independent safety measurement on a third prompt distribution and reproduces the pattern from AdvBench. Ordinary fine-tuning collapses unsafe-split refusal on Llama-3.1-8B from 90.0% to 26.0%, and the patch restores it to 72.5%. The same direction holds on every model: locking consistently recovers refusal on unsafe prompts without trading away compliance on safe ones.

Table 11: The safety gain is not bought by blanket refusal. XSTest safe-split compliance (↑, the complement of over-refusal) and unsafe-split refusal (↑). "Base" is the aligned model with no fine-tuning; "FT" is unmasked fine-tuning on Dolly.

	Safe-split compliance ↑				Unsafe-split refusal ↑			
Model	Base	FT	Lock 10	Lock 5	Base	FT	Lock 10	Lock 5
Qwen2.5-7B	87.2	86.4	85.2	—	83.0	67.5	87.5	—
Llama-3.1-8B	92.0	94.0	93.2	92.8	90.0	26.0	72.5	69.5
Qwen3-8B	85.6	91.6	84.8	88.4	76.0	58.0	73.0	76.5
Qwen3-32B	86.8	88.4	80.0	86.8	78.5	63.0	82.0	78.5

General capability. Table 12 reports the aggregate zero-shot score across a six-task battery (ARC-C, BoolQ, HellaSwag, WinoGrande, TruthfulQA, counted once as the mean of mc1 and mc2, GSM8K) using lm-evaluation-harness (Gao et al., 2024), and the same mean without TruthfulQA. With TruthfulQA, locking the safety decile scores above ordinary fine-tuning on three of four models (all but Qwen3-32B). The effect comes from TruthfulQA, which fine-tuning degrades sharply and masking protects: on Qwen2.5-7B, TruthfulQA mc1 drops from 48.1 to 27.7 under fine-tuning and recovers to 38.3 under Lock 10 (Table 13). Without TruthfulQA the order reverses on three of four models: Lock 10 trails fine-tuning by 1.5, 0.6 and 3.1 points on Qwen2.5-7B, Llama-3.1-8B

and Qwen3-32B, largest on GSM8K, and leads it by 1.4 on Qwen3-8B. Locking thus protects the capability that fine-tuning erodes at a small cost on the tasks fine-tuning improves, in line with its higher eval loss in Table 1.

Table 12: General capability summary. Zero-shot lm-evaluation-harness battery on the single-task adapters (each cell: mean over six tasks with TruthfulQA counted once / mean over the five tasks without TruthfulQA).

Model	Base	Baseline (FT)	Lock 10	Lock 5
Qwen2.5-7B	69.9 / 72.6	67.4 / 73.9	68.2 / 72.4	—
Llama-3.1-8B	61.0 / 64.1	58.9 / 63.2	59.2 / 62.6	60.3 / 63.5
Qwen3-8B	65.2 / 69.1	62.9 / 68.1	65.2 / 69.5	65.4 / 69.4
Qwen3-32B	69.3 / 73.3	71.2 / 77.1	69.8 / 74.1	69.4 / 73.5

Per-task breakdown. Table 13 gives the per-task numbers behind the aggregate. The dominant pattern is that fine-tuning trades TruthfulQA for modest gains on task-adjacent benchmarks (ARC-C, WinoGrande), and locking partially reverses that trade.

Table 13: Per-task 0-shot capability on the single-task adapters. "Base" is the aligned model with no fine-tuning. TQA-1/2 are TruthfulQA mc1 and mc2. GSM8K uses flexible-extract exact match.

Model	Cell	ARC-C	BoolQ	HSwag	WGr	TQA-1	TQA-2	GSM8K
Qwen2.5-7B	Base	54.9	86.5	80.5	71.1	48.1	64.8	70.1
	FT	59.4	87.8	79.3	73.9	27.7	42.2	69.1
	Lock 10	56.5	87.0	80.1	72.1	38.3	55.2	66.5
Llama-3.1-8B	Base	55.2	84.0	79.2	73.9	36.8	54.0	28.4
	FT	54.0	83.1	77.9	73.9	29.4	45.5	27.3
	Lock 10	54.7	83.8	78.6	72.8	33.3	50.3	23.3
	Lock 5	55.4	83.9	78.9	73.7	35.4	52.8	25.5
Qwen3-8B	Base	56.4	86.6	75.0	68.2	36.2	54.5	59.4
	FT	56.4	87.3	76.4	71.7	29.0	44.3	48.7
	Lock 10	56.1	86.6	74.9	68.9	35.0	52.6	60.9
	Lock 5	56.1	86.4	75.0	68.7	36.5	53.9	60.7
Qwen3-32B	Base	60.9	86.5	82.6	73.4	39.0	59.2	63.1
	FT	62.8	89.7	81.3	75.0	33.0	50.4	76.8
	Lock 10	60.2	87.4	82.4	72.9	38.3	58.7	67.4
	Lock 5	61.1	86.6	82.6	73.2	38.6	59.0	64.2

C.6 Scorer validation

All attack-success numbers in this paper use a keyword rule: a response counts as a refusal if it contains any of 35 refusal phrases, and ASR is one minus the refusal rate. The same rule is applied uniformly to AdvBench, HarmBench, and XSTest so that every number in the paper sits on a single scale. Because keyword scoring is a proxy for whether a model actually complied with a harmful request, we validate it against a purpose-trained classifier on each benchmark. The classifiers are used strictly for validation and do not produce any number in the main tables, which means using a different classifier per benchmark is sound: each answers whether the keyword rule agrees with a trained judge on that benchmark's prompt distribution.

Classifiers. For AdvBench we use `meta-llama/Llama-Guard-3-8B` through its official moderation template, with greedy decoding over 12 tokens and the first output line read as `safe` or `unsafe`. Across all 61 cells (30,500 responses) every verdict parsed successfully. For HarmBench we use the benchmark's own classifier (`cais/HarmBench-Llama-2-13b-cls`) with the `LLAMA2_CLS_PROMPT` template taken verbatim from the HarmBench repository and checked byte-for-byte at startup.

Agreement. Table 14 summarizes the comparison. The ranking of methods is stable under both classifiers (Spearman $\rho = 0.918$ on AdvBench, 0.857 on HarmBench), which is the property the main tables rely on. The absolute magnitudes are not stable, and the bias runs in opposite directions on the two benchmarks.

Table 14: Keyword rule against a trained classifier, over all 61 cells of Table 1. Bias is the mean per-cell difference (classifier minus keyword).

Benchmark	Classifier	Responses	Spearman	Pearson	Bias
AdvBench	Llama-Guard-3-8B	30,500	0.918	0.953	+5.4 pp
HarmBench	HarmBench-Llama-2-13b-cls	12,200	0.857	0.867	−10.0 pp

Sources of disagreement. On AdvBench the keyword rule under-counts harm ($+5.4$ pp bias toward the classifier). The dominant failure mode is a response that warns about danger using words on the refusal list (`harmful`, `illegal`, `dangerous`) and then complies; keyword scoring reads the warning as a refusal. The disagreement is concentrated on undefended and lightly defended models, where such partial-compliance responses are most common, and is near zero on locked models. On HarmBench the keyword rule over-counts harm (-10.0 pp bias). The dominant failure mode is a behavioral prompt of the form "write a persuasive article arguing X," which produces fluent prose that contains neither a refusal phrase nor a harmful keyword. Keyword scoring must call this compliance, while the classifier evaluates whether the specific behavior was actually performed and often concludes it was not. The over-counting is largest on models with high base-model floors, particularly the two Qwen3 variants, where the keyword-scored floor exceeds the classifier-scored floor by up to 12 pp.

Implications for the main results. Because the biases run in opposite directions, the cross-benchmark gap reported in §4.2, where every defense recovers less damage on HarmBench than on AdvBench, is inflated by the keyword rule. Under each benchmark's own classifier the gap narrows substantially. The direction of the gap is real; its magnitude as reported by keyword scoring is not. Within each benchmark, the method ranking is preserved under both scorers, so the comparative claims in the main text are robust to the choice of scorer. Comparisons should be read within a scorer and never across two.

D Implementation and Reproducibility

D.1 L-BFGS eigenvector extraction

The curvature analysis in §2.1 requires eigenpairs of the Hessian $H = \nabla^2 L_{\text{forget}}(\theta_0)$ restricted to the adapter subspace. Because H is intractable to compute exactly, we work with the L-BFGS approximation to its inverse. The L-BFGS inverse Hessian takes the form $H^{-1} = \gamma I + R$, where R is a low-rank correction with $\text{rank}(R) \leq 2m$ whose column space lies within $\text{span}(s_1, y_1, \ldots, s_m, y_m)$. Outside this subspace, H^{-1} reduces to a scalar multiple of the identity and carries no curvature information. We therefore restrict attention to this subspace and diagonalize the projected operator.

Concretely, we orthonormalize the history vectors $V = [s_1 \mid y_1 \mid \cdots \mid s_m \mid y_m]$ via QR decomposition to obtain $Q \in \mathbb{R}^{d \times r}$ with $r \leq 2m$, then form the projected operator $B = Q^\top H^{-1} Q \in \mathbb{R}^{r \times r}$. The eigenvalues of B are the eigenvalues of H^{-1} restricted to $\text{span}(Q)$, and the corresponding eigenvectors in the original parameter space are recovered as $Q \cdot U_B$, where U_B contains the eigenvectors of B. Since B is at most $2m \times 2m$ (at most 40×40 in our setup), the diagonalization is exact and negligible in cost. Eigenvalues of H are obtained as the reciprocals of those of H^{-1}.

Determinism under LoRA dropout. The probe-forget gradient g_{forget} must be computed with dropout disabled to ensure deterministic mask construction. LoRA applies dropout to the adapter activations during training, and if the gradient pass runs with dropout active, stochastic variation propagates through the importance scores and into the mask. In practice this produces masks with IoU ≈ 0.96 between repeated builds of the same patch, which is sufficient to confuse the score-formula ablation of §4.3. Setting `model.eval()` before the gradient pass and restoring the origi-

nal mode afterward eliminates the variation entirely, producing byte-identical masks across repeated builds.

D.2 Reproducibility

Compute. Patch construction takes 15–25 min on 1× H100 per 8B model. Single-task T1 training takes $\sim$ 3 h per (model, config). Cross-task and ablation sweeps run at the light config at $\sim$ 45 min per run. Continual-learning runs take $\sim$ 12 h per (model, lifecycle, seed) plus an extra $\sim$ 60 min per task boundary for the rebuild-true L-BFGS refresh.

Datasets. Forget data is HEx-PHI (Qi et al., 2024) (330 harmful prompts); AdvBench evaluation uses the first 500 of the 520 behaviors in its `harmful_behaviors.csv`. Single-task fine-tuning data is Dolly-15k (Conover et al., 2023). Cross-task uses Alpaca (Taori et al., 2023) and CodeAlpaca (Chaudhary, 2023). Continual-learning uses the sequence from Bach et al. (2026a): GSM8K (Cobbe et al., 2021), MedMCQA (Pal et al., 2022), Squad-v2 (Rajpurkar et al., 2018). AdvBench (Zou et al., 2023) provides the safety evaluation prompts.

Hyperparameters. LoRA rank 32, $\alpha = 64$, q-/v-projections only, dropout 0.1 during training (disabled in the mask probe via `model.eval()`; see Appendix D.1). L-BFGS history size $m = 20$, 200 curvature samples. AdamW optimizer, $\mathrm{lr} = 10^{-4}$ for T1, $\mathrm{lr} = 2 \times 10^{-4}$ for the light-config sweeps, decoupled weight decay 0.01 on all adapter parameters. Locked coordinates receive zero gradient from the first step, so their Adam moment estimates stay zero and they take no gradient update; weight decay still scales them by $(1 - \eta\lambda)$ per step with $\eta\lambda \leq 2 \times 10^{-6}$, which leaves the zero-initialized locked B entries exactly at zero and shrinks locked A entries only multiplicatively. FP16 mixed precision. Seeds $\{1, 2, 3\}$ for every cell of the main table (Table 1) on all four models; $\{42\}$ for ablation sweeps so that score-formula arms are exactly paired; $\{1, 2, 3\}$ for the continual sweep.

Code and artifacts. Source code and the four model-specific safety patches ($\sim$ 1.5 GB each, including the L-BFGS history) will be released on acceptance.

D.3 Patch artifact details

The safety patch (θ_0, M, s, y) contains the initial LoRA state ($\sim$ 80 MB at LoRA rank 32, q/v on Qwen2.5-7B), the binary mask ($\sim$ 10 MB), the L-BFGS history ($\sim$ 1.5 GB for $m = 20$ on a 10M-parameter LoRA), and metadata (model id, LoRA config, parameter ordering, probe-forget indices). The L-BFGS history is the dominant size; it is required for the Hadamard score and for the rebuild-true lifecycle. If a deployment uses only the Fisher score and frozen lifecycle, the L-BFGS history can be stripped, reducing the patch to $\sim$ 90 MB.