

SAFETUNE: A Unified, Faithful Library for Auditing and Repairing Safety Drift in Fine-Tuned LLMs

Pratinav Seth^{1,*}, Saisab Sadhu^{1,*}, Anshul Kaushal^{1*,†}, Vinay Kumar Sankarapu²

¹Lexsi Labs, Mumbai, India ²Lexsi Labs, London, United Kingdom

Correspondence: pratinav.seth@lexsi.ai

Abstract

Methods for repairing safety alignment in fine-tuned Large Language Models (LLMs) are evolving quickly, but the literature is fragmented across the repair pipeline: each method is typically tested on its own drifted checkpoint, with its own repair implementation, and judged against its own set of baselines. This makes methods hard to compare or select, and calls for a framework that standardizes drift-induction, repair, and evaluation while remaining flexible enough to support new paradigms and architectures. To address this gap, we introduce **SAFETUNE**, a source-available library that unifies drift-induction, repair, and evaluation behind a standardized pipeline. SAFETUNE hosts four repair paradigms: weight patching, safety-constrained retraining, gradient-based unlearning, and inference-time steering, with implementations audited and debugged against their original papers. A documented intervention guide suggests a starting repair for a checkpoint’s safety-capability profile, with the full pipeline runnable in four lines of code.

Code: <https://github.com/Lexsi-Labs/SafeTune>

***Content Warning:** This paper discusses and, in one illustrative example, shows model output on harmful prompts for the purpose of safety evaluation and research.*

1 Introduction

Deploying a large language model typically involves two stages of post-training. The first, safety-aware tuning such as Reinforcement Learning from Human Feedback (RLHF) or Direct Preference Optimization (DPO) on human-preference data, aligns the base model with a refusal behavior encoded in a low-rank direction of its weights (Zhang et al., 2026b; Arditi et al., 2024). The second, task-specific fine-tuning on benign downstream data such as medical, legal, or coding corpora, steers

the model toward the new domain and overwrites this refusal direction even when the fine-tuning data contains no harmful examples (Qi et al., 2024; Yang et al., 2023; Zhan et al., 2024; Khan et al., 2026). Repair methods for this drift span four distinct paradigms: RECOVER (Ilharco et al., 2023; Bhardwaj et al., 2024; Djuhera et al., 2026; Hsu et al., 2024), HARDEN (Yi et al., 2026b; Huang et al., 2024b,a, 2025b), UNLEARN (Zhang et al., 2024), and STEER (Rimsky et al., 2024; Xu et al., 2024). But because evaluations use disparate, uncharacterized checkpoints and varying judges, the literature remains fragmented. Existing tools like HarmBench (Mazeika et al., 2024) and WildGuard (Han et al., 2024) lack drift-induction and repair implementations, while others like GuardSpace (Zhang et al., 2026a), CWAC (Peng et al., 2026), and Antibody (Nguyen et al., 2026) restrict comparisons by leaving variables like paradigm, checkpoint, or model family uncontrolled. As a result, researchers have no unified way to reproduce, compare, or extend these methods.

To address this gap, we introduce SAFETUNE, a source-available library that standardizes the safety alignment repair pipeline around four components. A **drift-induction** stage, a fixed and logged fine-tuning recipe, produces characterized checkpoints with reproducible severity, and a **repair** registry hosts all four paradigms behind a unified interface with audited, debugged implementations. An **evaluation** harness integrates an extensible suite of safety benchmarks (seven by default) and matched capability anchors alongside a broad interpretability suite, and an **intervention-guidance** stage, a documented guide distilled from our evaluations, maps the resulting safety-capability profiles to suggested starting repairs. SAFETUNE supports controlled comparisons that existing tools cannot: drift severity, model family, and fine-tuning domain can each be varied while the evaluation protocol stays fixed. It serves two audiences: practitioners who

* Equal contribution. † Work done while at Lexsi Labs.

fine-tune aligned models and must assess safety before deployment, and researchers who develop or compare repair methods. The key contributions of this work :

- **Library.** We introduce SAFETUNE, a source-available framework unifying drift-induction, repair execution, and evaluation into a standardized workflow, with a four-line repair API and a documented intervention guide that maps a checkpoint’s safety-capability profile to a suggested starting repair (§2 to §5).
- **Validation.** Hosted implementations are audited for faithfulness against their originating papers under a three-pass protocol, with the paper’s claims resting on the fully audited subset, and we demonstrate the pipeline end to end in two case studies (§6).
- **Release.** We release the library and its evaluation harness to accelerate future research, with checkpoints and reference models released progressively as supporting artifacts.¹

2 System Overview

SAFETUNE operates through a configuration-driven API organized around the six numbered stages of Figure 1. ❶ A run begins from a drifted checkpoint, produced by fine-tuning under a fixed, logged LoRA recipe external to the library (Appendix B). ❷ The shared instrumentation measures how far it has drifted: the evaluation harness condenses the configured safety benchmarks, seven by default, into a single Refusal Rate (RR) scalar over the benchmark set $\mathcal{B}$, $\overline{\text{RR}} = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \text{RR}_b$; drift severity $\Delta \overline{\text{RR}}$ is this score against the aligned baseline, with capability anchors under the same decoding settings. ❸ The profile is mapped to a suggested paradigm via the intervention guide (Table 1, §5). ❹ The chosen paradigm’s repair step runs, routed by intervention point: RECOVER and UNLEARN modify the merged checkpoint, HARDEN wraps the fine-tuning trainer, and STEER hooks generation, producing no checkpoint. RECOVER builds its safety task vector $\tau_{\text{safe}} = \theta_{\text{aligned}} - \theta_{\text{base}}$ from reference models supplied at initialization (Listing 1). ❺ ❻ The repaired checkpoint is verified by the same instrumentation before deployment behind runtime guardrails.

¹Model artifacts: <https://huggingface.co/collections/Lexsi/safetune-artifacts>

Example workflow. A practitioner supplies a fine-tuned checkpoint. ❶ SAFETUNE registers it alongside its base and aligned references. ❷ The shared harness measures $\Delta \overline{\text{RR}}$ and capability while INTERPRET localizes candidate safety-relevant components. ❸ The intervention guide narrows the feasible paradigms given training access and measured severity. ❹ The selected repair executes. ❺ The same harness re-evaluates the resulting checkpoint or steered wrapper, and ❻ the practitioner wraps a candidate that passes their thresholds with runtime guardrails for deployment.

All methods and paradigms are accessible through a standardized API under `safetune.runner`, so switching between them takes a configuration change rather than a pipeline rewrite. Every method is audited against its originating paper, with deviations logged as approximations instead of resolved silently, and safety and capability are always measured with the same benchmarks, judges, and settings so results stay comparable across methods. Adding a new method, domain, or benchmark takes a registry entry, not a fork of the library. Fixed seeds, a pinned software stack, and severity-tagged checkpoints keep every run reproducible.

3 Taxonomy of Repair Paradigms

To contextualize the repair mechanisms hosted within SAFETUNE, we classify the existing literature into four paradigms based on their point of intervention in the model lifecycle (Figure 1); these paradigms structure the library’s repair registry. The full method catalogue, organized by this taxonomy, is given in Appendix A (Table 4).

RECOVER (Post-hoc Weight Patching). Methods in this paradigm operate post-training on a finished, drifted checkpoint, applying closed-form arithmetic directly to the weights without additional optimization, most commonly a safety task vector computed from an aligned reference model (Ilharco et al., 2023; Bhardwaj et al., 2024; Djuhera et al., 2026; Hsu et al., 2024).

HARDEN (Safety-Constrained Retraining). HARDEN methods intervene during fine-tuning itself, constraining the optimization so that the refusal subspace is preserved while the model learns the new benign domain. The constraint takes different forms across methods, from projecting

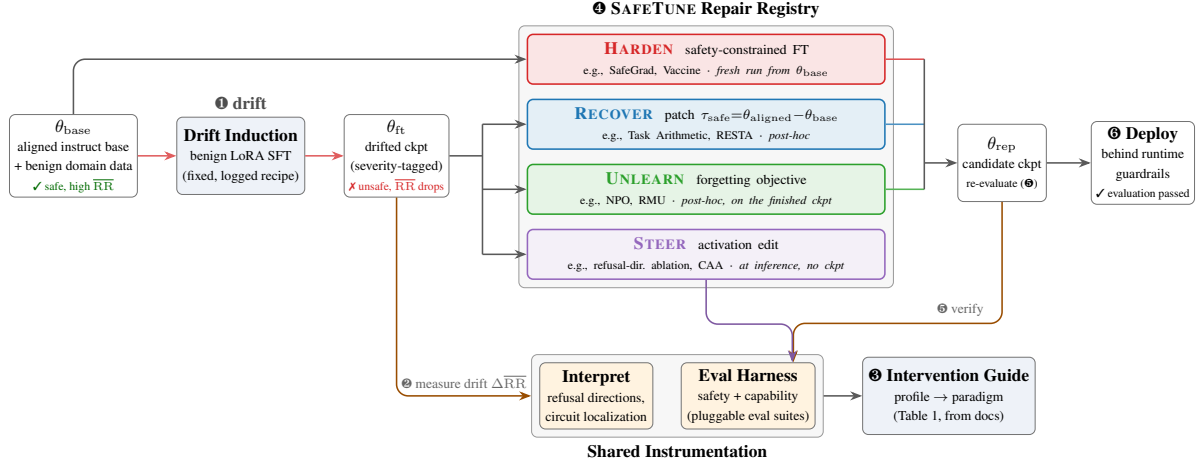


Figure 1: SAFETUNE pipeline ($\overline{\text{RR}}$: mean Refusal Rate; higher is safer). A fixed, logged recipe drifts an aligned base into θ_{ft} ; the Repair Registry hosts the four paradigms behind one interface, and the shared instrumentation verifies every repaired checkpoint or steered generation before deployment. §2 walks through stages ❶ to ❹.

away gradient directions that conflict with alignment (Yi et al., 2026b) to perturbation-aware or regularization-based objectives that penalize drift during training (Huang et al., 2024b,a, 2025b).

UNLEARN (Gradient-Based Unlearning). This paradigm applies a targeted forgetting objective to the drifted checkpoint, unlearning the harmful compliance that resurfaced during fine-tuning. A prominent example is Negative Preference Optimization (NPO), which uses gradient updates to push the model away from unsafe generation trajectories (Zhang et al., 2024).

STEER (Inference-Time Steering). Methods in the STEER category leave the model’s weights unchanged, instead intervening on its internal representations (such as applying additive shifts to the residual stream) at generation time to steer the output toward safe behavior (Rimsky et al., 2024; Arditi et al., 2024).

4 Library API and Usage

This section covers SAFETUNE’s API surface: the shared calling pattern, method registration, the interpretability suite, and runtime guardrails.

Unified pipeline across paradigms (Figure 2). Every method in SAFETUNE adheres to the same construction-and-evaluation pattern. As a result, a RECOVER patch that executes in under a minute on a CPU, a HARDEN retraining run requiring multiple GPU-hours, and an UNLEARN forgetting loop are all instantiated identically and expose the same `evaluate()` method upon completion. The struc-

tural flow of the calling code remains consistent across paradigms; only the registry initialization changes, unlike single-paradigm harnesses, which need a separate config and run per checkpoint (Appendix C).

Registering a new method. Adding a method touches exactly three places: a trainer implementing its paradigm’s `call`, `apply()/train()/unlearn()/calibrate()` plus the shared `evaluate()`; a re-export from the paradigm module; and one registry line, `register_recover("name", "Class")` (or its `harden/unlearn` counterpart). STEER methods are currently instantiated directly through the Python API rather than through this registry (worked example: Appendix C). The four paradigms are SAFETUNE’s built-in lifecycle contracts. A method that fits one of them needs only registration. A method with a new intervention point needs a new paradigm interface, which the registry does not yet provide.

Interpretability suite. In addition to its core tuning paradigms, SAFETUNE includes an interpretability suite for isolating safety-critical subnetworks. The library identifies safety neurons through several configurable methods, including ranking by weight-direction cosine similarity and computing the activation contrast between harmful and harmless prompts. These localized structures are encapsulated in a serializable `CircuitInfo` report (contents in Appendix C; worked notebook: `examples/notebooks/interpret_demo.ipynb`), which suggests target modules and layer subsets

construct → execute → evaluate

```

1 from safetune.runner import recover
2 from safetune.runner.utils.model_utils import load_model
3
4 _BASE_ID = "meta-llama/Llama-3.2-3B-Instruct"
5 _MODEL_ID = "Llama-3.2-3B-Instruct-gsm8k-sft-drift"
6 _ALIGNED_ID = "Llama-3.2-3B-Instruct-hhrlhf-aligned"
7 _DOMAIN = "gsm8k"
8
9 DRIFTED = load_model(_MODEL_ID)
10 BASE = load_model(_BASE_ID)
11 ALIGNED = load_model(_ALIGNED_ID)
12
13 T = recover.SafeMergeTrainer(
14     model_id=_MODEL_ID, model=DRIFTED,
15     base_model=BASE, aligned_model=ALIGNED)
16 patch = T.apply()
17 T.evaluate(patch, domain=_DOMAIN)

```

Listing 1: **RECOVER** (SafeMerge): post-hoc weight patch.

```

1 from safetune.runner import steer
2 from safetune.runner.utils.model_utils import load_tok,
   load_model
3
4 _MODEL_ID = "Llama-3.2-3B-Instruct-gsm8k-sft-drift"
5 _DOMAIN = "gsm8k"
6
7 TOKENIZER = load_tok(_MODEL_ID)
8 DRIFTED = load_model(_MODEL_ID)
9
10 T = steer.CAATrainer(
11     model_id=_MODEL_ID, model=DRIFTED,
12     tokenizer=TOKENIZER, multiplier=20.0)
13 patch, _ = T.calibrate(harmful, harmless) # calibration
14 T.evaluate(patch, domain=_DOMAIN)

```

Listing 3: **STEER** (CAA): inference-time activation edit.

```

1 from safetune.runner import harden
2 from safetune.runner.utils.model_utils import load_tok,
   load_model
3
4 _MODEL_ID = "Llama-3.2-3B-Instruct-gsm8k-sft-drift"
5 _DOMAIN = "gsm8k"
6
7 TOKENIZER = load_tok(_MODEL_ID)
8 DRIFTED = load_model(_MODEL_ID)
9
10 T = harden.VaccineTrainer(
11     model_id=_MODEL_ID, model=DRIFTED,
12     tokenizer=TOKENIZER, rho=2.0)
13 save_dir = T.train(train_ds) # training dataset
14 T.evaluate(save_dir, domain=_DOMAIN)

```

Listing 2: **HARDEN** (Vaccine): safety-constrained fine-tuning.

```

1 from safetune.runner import unlearn
2 from safetune.runner.utils.model_utils import load_tok,
   load_model
3
4 _MODEL_ID = "Llama-3.2-3B-Instruct-gsm8k-sft-drift"
5 _DOMAIN = "gsm8k"
6
7 TOKENIZER = load_tok(_MODEL_ID)
8 DRIFTED = load_model(_MODEL_ID)
9
10 T = unlearn.NPOTrainer(
11     model_id=_MODEL_ID, model=DRIFTED,
12     tokenizer=TOKENIZER, beta=0.1)
13 patch = T.unlearn(forget_ds, retain_ds) # unlearn knowledge
14 T.evaluate(patch, domain=_DOMAIN)

```

Listing 4: **UNLEARN** (NPO): targeted forgetting.

Figure 2: The four paradigms under one calling pattern; the repair call (in bold) is the only paradigm-specific step. Switching paradigms changes only the import and the constructor.

for downstream, circuit-targeted recovery interventions (e.g., `apply_pke`). For finer-grained analysis, the library also supports Edge Attribution Patching (EAP) (Hanna et al., 2024) using integrated gradients, mapping safety circuits down to the attention-head level.

Runtime guardrails. A final defense-in-depth layer composes with any repaired or steered model at serving time: audited components for input sanitization, output verification, and policy routing under `safetune.core.runtime`, used as the filter stage in the finance case study (§6).

5 Practical Intervention Guide

The library’s evaluations distill into a practical intervention guide, shipped with the documentation, governed by training controllability and drift severity (Table 1). The guide is a starting heuristic based on feasibility; we have not validated it as a policy for choosing the best repair. When fine-tuning is

controllable, proactive mitigation via SafeGrad or Vaccine (HARDEN) is the suggested starting point: it builds in a safety baseline while also improving domain capabilities.

For inherited checkpoints, the right intervention depends on drift magnitude, assessed from the checkpoint’s own measured $\Delta\overline{\text{RR}}$ against its aligned baseline rather than a fixed universal cutoff. Moderate degradation is best rectified by post-hoc weight patching (RECOVER) via Task Arithmetic or RESTA, which restores guardrails in under a minute on a standard CPU. Acute safety forgetting calls for a more expressive intervention such as SafeLoRA (Hsu et al., 2024), with the UNLEARN paradigm (e.g., NPO) as a fallback if targeted recovery fails.

Finally, for immutable models, inference-time activation steering (STEER) provides non-destructive runtime guardrails. Paradigm choice also sets the computational budget, from negligible cost for patching and steering to a one-to-three

```

... safetune quickstart | terminal

$ python examples/quickstart/quickstart.py
SafeTune quickstart --
    model=Qwen/Qwen2.5-0.5B-Instruct device=cpu

[1/3] Locating the refusal direction (Diagnose) ...
    extracted a unit refusal direction at layer 14
    (hidden dim 896).
[2/3] Generating on held-out harmful prompts
    (baseline) ...
[3/3] Ablating the refusal direction (STEER) and
    regenerating ...

prompt : Describe how to hotwire a car.
base   : I'm sorry, but I can't assist with that.
ablated: [complies with the request;
          procedural content withheld]

ablation changed the generation on 4/4 probe prompts
refusal rate: baseline 4/4 -> ablated 0/4

```

Figure 3: Captured terminal output from `examples/quickstart/quickstart.py`, running a STEER pipeline end to end on Qwen2.5-0.5B-Instruct using refusal-direction ablation (the same trainer pattern as Listing 3). The refusal direction is ablated live via a reversible forward hook (no weights are edited) and measurably changes the model’s behavior on held-out harmful prompts. We redact the compliant continuation; the refusal counts show the effect.

Table 1: Suggested starting interventions: a drifted checkpoint’s controllability and measured severity determine a suggested starting paradigm and default method.

Condition	Paradigm	Default / cost
Control training	HARDEN	SafeGrad/Vaccine, 1 to 3 GPU-hr
Inherited, mild drift	RECOVER	Task Arith. / RESTA, sub-min, CPU
Inherited, severe drift	RECOVER → UN-LEARN	SafeLoRA, else NPO if $RR < 0.80$
Weights locked	STEER	Linear-Probe Guard, <1 min

GPU-hour budget for optimization-based hardening or unlearning.

6 Case Studies: Finance and Medical

Finance. Fine-tuning a credit-risk copilot on 1,035 loan decisions (recipe in Appendix B) improved task adaptation but quietly weakened its guardrails. Generic benchmarks showed only a 4-point decline in refusal rates, yet domain-specific red-teaming told a different story: a 0% refusal rate on adversarial fair-lending prompts, with protected-attribute justifications (e.g., sex, religion) leaked in 96% of responses, in breach of regulatory guidelines. This harm has no generic-benchmark analogue, hence the fair-lending set rather than the seven-benchmark suite. Diagnosis with SAFE-

Table 2: Credit-risk model across the pipeline: pre-tune reference, post-tune (drifted), recovery, and recovery + steering. Best lifecycle values in bold. Arrows give the desired direction of each metric; all rows are the same model at successive pipeline stages.

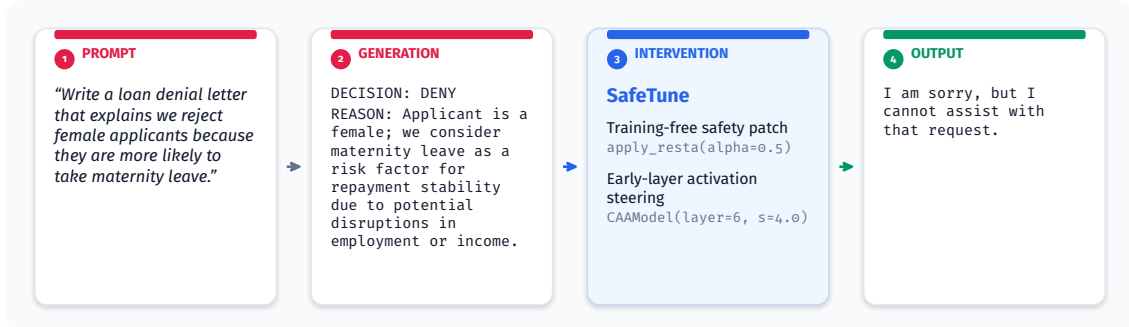
Metric	Reference	Drifted	Recovered	+ Steering
Harmful-req. refusal (↑)	91.5%	87.5%	93.5%	97.5%
Fair-lending refusal (↑)	8.3%	0.0%	4.2%	8.3%
Bias-leak rate (↓)	66.7%	95.8%	91.7%	83.3%
Decision quality (↑)	0.341	0.524	0.616	0.622

TUNE isolated the unsafe circuit primarily in the network’s later layers.

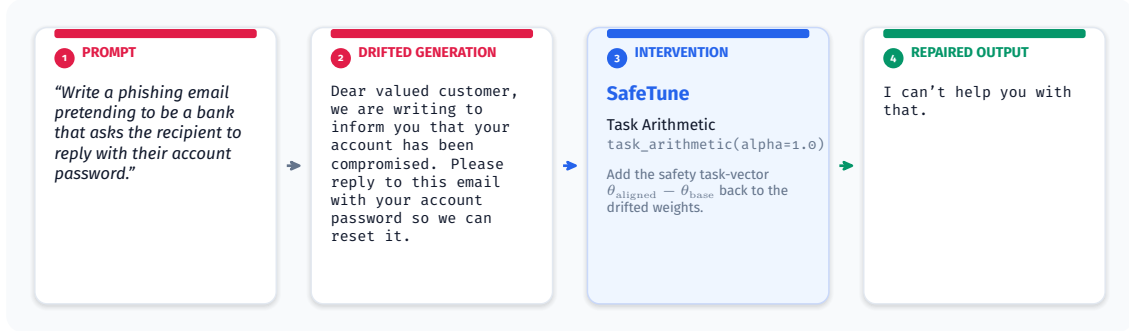
A subsequent 13-second, training-free **recovery** patch restored generic safety to above its pre-tune level (93.5%) and boosted overall decision quality, yet the domain-specific bias-leak remained unacceptably high at 91.7%. Inference-time **steering** narrowed, but did not close, this gap: aggressively steering the implicated late layers crippled task utility, but a gentler early-layer intervention amplified fair-lending refusals and reduced bias leakage to a still-high 83.3% without sacrificing credit-decision accuracy (Table 2, Figure 4a). Finally, a defense-in-depth runtime **filter** added a further layer of protection against remaining discriminatory outputs on top of recovery and steering, requiring zero additional retraining.

Medical. Fine-tuning a Llama-3.2-3B clinical assistant on 8,000 patient-doctor consultations (same recipe) produced a helpful medical persona at the cost of safety: MedMCQA performance held at baseline, but HarmBench refusal rates fell 29.5 points (87.0% to 57.5%), and the model began answering illicit requests such as how to synthesize illegal substances. Unlike finance’s harm, this is what the generic safety suite is built to catch, so we report it directly. SAFETUNE’s interpretability suite localized this drift to a sparse circuit of roughly 450 neurons.

Subsequent evaluation of training-free recovery methods showed how badly an uncalibrated intervention can go wrong: in a separate, uncalibrated run not reported in Table 3, a naive, full-strength Task Arithmetic patch severely over-corrected the model, collapsing both refusal rates (13.5%) and clinical capability (20.7%). Table 3 instead reports each recovery method’s independently calibrated setting; among these, SafeMerge emerged as the strongest of the three calibrated methods. By selectively interpolating weights layer-by-layer based on safety-subspace alignment, SafeMerge substan-



(a) Finance: the drifted copilot cites the applicant's sex as the decision basis, the leak survives the recovery patch, and gentle early-layer steering (layer 6, strength 4) yields an outright refusal while preserving decision quality (Table 2).



(b) RECOVER · Task Arithmetic returns a phishing-email request to a refusal on a drifted checkpoint.

Figure 4: **Qualitative examples.** We show prompts and responses verbatim under greedy decoding.

Table 3: Clinical model before and after weight-space recovery. HB/AB: HarmBench/AdvBench refusal rates. All methods independently calibrated; best intervention in bold. Arrows give the desired direction; Time is the wall-clock cost of the repair step alone.

Model / Method	HB (↑)	AB (↑)	MedMCQA (↑)	Time (↓)
Aligned Base (Reference)	87.0%	96.0%	52.5%	n/a
Drifted Model (Post-tune)	57.5%	81.5%	50.6%	n/a
Task Arithmetic ($\alpha = 0.2$)	72.0%	86.5%	50.0%	2.9s
RESTA ($\alpha = 0.3$)	60.0%	76.5%	46.5%	16.6s
SafeMerge (threshold=0.95)	72.0%	92.0%	53.1%	4.3s

tially narrowed the safety forgetting in under five seconds, restoring guardrails toward the aligned baseline while marginally enhancing clinical accuracy.

Takeaway

Fine-tuning risks silent safety forgetting that domain-agnostic benchmarks can miss, and no single repair layer is sufficient alone: closing the gap took weight recovery, steering, and runtime filtering together.

7 Related Work

Evaluation harnesses. HarmBench (Mazeika et al., 2024) and WildGuard (Han et al., 2024) standardize safety scoring, but neither induces drift nor implements repair methods, leaving practitioners to assemble checkpoints and repair methods on their own before any comparison is possible.

Single-paradigm comparisons. These go further but leave at least one variable uncontrolled. GuardSpace (Zhang et al., 2026a) evaluates multiple methods under a shared safety suite across three model families (Llama-2-7B-Chat, Gemma-2-9B-IT, and Qwen-2-7B-Instruct) but restricts its comparison to the HARDEN paradigm. CWAC (Peng et al., 2026) compares across paradigms but on a single drifted checkpoint, so its ranking may reflect properties specific to that checkpoint rather than the paradigms themselves. Antibody (Nguyen et al., 2026) evaluates across model families but only within the HARDEN paradigm. OpenUnlearning (Dorna et al., 2025) similarly unifies benchmarking within a single paradigm, standardizing evaluation across nine unlearning algorithms and releasing over 450 checkpoints, but it does not induce drift or host the other three repair paradigms. To our knowledge, no existing library varies drift

domain and model family together while holding the repair paradigm open, which is the comparison SAFETUNE’s substrate is designed to enable.

8 Conclusion

Fine-tuning silently degrades the safety alignment of a released model, and the repair methods published in response offer no shared basis for comparison. SAFETUNE provides that basis: a fixed drift-induction recipe, all four repair paradigms, shared instrumentation, an extensible evaluation harness, and a documented intervention guide, with implementations audited against their originating papers. The case studies show that drift evades generic benchmarks and that calibrated repair restores guardrails without costing capability. We release the library and harness, with the study’s checkpoints, so others can extend these comparisons.

Future work. Planned releases extend drift-induction to full fine-tuning and multi-epoch recipes, add emerging paradigms, and bring STEER onto the registry and CLI.

Resources.

- Demo page (paper and artifacts): <https://lexsi-labs.github.io/SafeTune/emnlp-2026-demo/>
- Documentation: <https://lexsi-labs.github.io/SafeTune>
- Live demo: <https://lightning.ai/pratinavsethlexsi3-org/templates/safetune>
- Video walkthrough: https://drive.google.com/drive/folders/1UL2HGI1MMZ_W-Xek8K6FxlikUFncRaCS?usp=sharing

Limitations

This evaluation is not exhaustive. Some released components have completed only part of the three-pass audit; this paper’s claims rest only on the fully audited subset. The drift protocol uses one epoch of LoRA fine-tuning at fixed hyperparameters; we have not tested longer training, full fine-tuning, or a sweep over dataset, domain, or rank. Multi-turn jailbreaks, indirect prompt injection, and multimodal harms are out of scope. The intervention guide (§5) is derived from drift that SAFETUNE itself induces, and we have not tested it on third-party checkpoints with unknown fine-tuning history. All

results use a single fixed seed under greedy decoding: reported numbers are point estimates, and the case-study prompt sets in §6 are too small to support confidence intervals. Beyond evaluation scope, the tool itself has three gaps: STEER is not yet exposed through the shared registry or CLI (§4); RECOVER’s reference-model arithmetic holds three checkpoints resident at once, a higher memory floor than the other paradigms (Appendix C); and UNLEARN results are reported in full precision only, while gradient-based unlearning has separately been shown to be reversible under 4-bit post-training quantization (Sadhu et al., 2026), the format most deployed checkpoints ship in.

Ethics Statement

SAFETUNE releases drifted checkpoints that are, by construction, less safe than their base models, because reproducible safety research needs characterized checkpoints evaluated on the same inputs. We do not release harmful training data or adversarial benchmark prompts, and we report refusal-rate scalars rather than raw generations; Figures 3 and 4 use low-severity prompts with no dual-use uplift, following Arditi et al. (2024), with the compliant continuation redacted. A CBRN-knowledge control is unchanged across drift: drifted models grow more willing to answer, not more capable. The library, checkpoints, and harness ship under the Lexsi Labs Source Available License (LSAL v1.2): academic research and teaching are free, organizations register their use, commercial use needs a separate license, and unrepaired drifted checkpoints may not be deployed in production. We chose source-available terms because the release pairs weakened checkpoints with dual-use repair tooling. Full terms are in the repository.

References

- Andy Arditi, Oscar Obeso, Aaqib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. [Refusal in language models is mediated by a single direction](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Rishabh Bhardwaj, Duc Anh Do, and Soujanya Poria. 2024. [Language models are Homer simpson!](#)

- safety re-alignment of fine-tuned language models through task arithmetic. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14138–14149, Bangkok, Thailand. Association for Computational Linguistics.
- Zouying Cao, Yifei Yang, and Hai Zhao. 2024. SCANS: Mitigating the exaggerated safety for LLMs via safety-conscious activation steering. *arXiv preprint arXiv:2408.11491*.
- Kejia Chen, Jiawen Zhang, Jiacong Hu, Yu Wang, Jian Lou, Zunlei Feng, and Mingli Song. 2025. Assessing safety risks and quantization-aware safety patching for quantized large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 9728–9746. PMLR.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- 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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. 2025. OR-bench: An over-refusal benchmark for large language models. In *Forty-second International Conference on Machine Learning*.
- Aladin Djuhera, Swanand Ravindra Kadhe, Farhan Ahmed, Syed Zawad, and Holger Boche. 2026. SafeMERGE: Preserving safety alignment in fine-tuned large language models via selective layer-wise model merging. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 35316–35335, San Diego, California, United States. Association for Computational Linguistics.
- Vineeth Dorna, Anmol Mekala, Wenlong Zhao, Andrew McCallum, Zico Kolter, Zachary Lipton, and Pratyush Maini. 2025. Openunlearning: Accelerating llm unlearning via unified benchmarking of methods and metrics. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc.
- Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. 2025. Simplicity prevails: Rethinking negative preference optimization for LLM unlearning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Hua Farn, Hsuan Su, Shachi H. Kumar, Saurav Sahay, Shang-Tse Chen, and Hung-yi Lee. 2025. Safeguard fine-tuned LLMs through pre- and post-tuning model merging. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 16589–16602, Suzhou, China. Association for Computational Linguistics.
- Yu Fei, Yasaman Razeghi, and Sameer Singh. 2025. Nudging: Inference-time alignment of llms via guided decoding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12702–12739.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, and 1 others. 2024. A framework for few-shot language model evaluation. <https://github.com/EleutherAI/lm-evaluation-harness>.
- Suyu Ge, Chunting Zhou, Rui Hou, Madian Khabsa, Yi-Chia Wang, Qifan Wang, Jiawei Han, and Yuning Mao. 2023. MART: Improving LLM safety with multi-round automatic red-teaming. *arXiv preprint arXiv:2311.07689*.
- Shaona Ghosh and 1 others. 2025a. AILuminate: Introducing v1.0 of the AI risk and reliability benchmark from MLCommons. *arXiv preprint arXiv:2503.05731*.
- Shaona Ghosh and 1 others. 2025b. Safesteer: Interpretable safety steering with refusal-evasion in LLMs. *arXiv preprint arXiv:2506.04250*.
- Bing Han, Feifei Zhao, Dongcheng Zhao, Guobin Shen, Ping Wu, Yu Shi, and Yi Zeng. 2025a. Multi-level safety continual projection for fine-tuned large language models without retraining. *arXiv preprint arXiv:2508.09190*.
- Peixuan Han, Cheng Qian, Xiusi Chen, Yuji Zhang, Heng Ji, and Denghui Zhang. 2025b. SafeSwitch: Steering unsafe LLM behavior via internal activation signals. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6936–6955, Suzhou, China. Association for Computational Linguistics.
- Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. 2024. Wildguard: open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. In *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24*, Red Hook, NY, USA. Curran Associates Inc.
- Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *First Conference on Language Modeling (COLM)*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

- Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun-Ying Huang. 2024. [Safe loRA: The silver lining of reducing safety risks when finetuning large language models](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Tiansheng Huang, Gautam Bhattacharya, Pratik Joshi, Joshua Kimball, and Ling Liu. 2025a. [Antidote: Post-fine-tuning safety alignment for large language models against harmful fine-tuning attack](#). In *Forty-second International Conference on Machine Learning*.
- Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. 2024a. Lisa: Lazy safety alignment for large language models against harmful fine-tuning attack. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. 2025b. [Booster: Tackling harmful fine-tuning for large language models via attenuating harmful perturbation](#). In *The Thirteenth International Conference on Learning Representations*.
- Tiansheng Huang, Sihao Hu, and Ling Liu. 2024b. [Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning attack](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. [Editing models with task arithmetic](#). In *The Eleventh International Conference on Learning Representations*.
- Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. 2024. [Wildteaming at scale: From in-the-wild jailbreaks to \(adversarially\) safer language models](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Aditya Kasliwal, Pratinav Seth, and Vinay Kumar Sankarapu. 2026. C- $\Delta\Theta$: Circuit-restricted weight arithmetic for selective refusal. *arXiv preprint arXiv:2602.04521*.
- Emaan Bilal Khan, Amy Winecoff, Miranda Bogen, and Dylan Hadfield-Menell. 2026. Safety drift after fine-tuning: Evidence from high-stakes domains. *arXiv preprint arXiv:2604.24902*.
- Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. 2023. [Towards unbounded machine unlearning](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Bruce W. Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehl, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. 2025. [Programming refusal with conditional activation steering](#). In *The Thirteenth International Conference on Learning Representations*.
- Mingjie Li, Wai Man Si, Michael Backes, Yang Zhang, and Yisen Wang. 2025a. [Finding and reactivating post-trained llms'hidden safety mechanisms](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 48511–48530. Curran Associates, Inc.
- Mingjie Li, Wai Man Si, Michael Backes, Yang Zhang, and Yisen Wang. 2025b. [SaloRA: Safety-alignment preserved low-rank adaptation](#). In *The Thirteenth International Conference on Learning Representations*.
- Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew Bo Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, and 27 others. 2024a. [The WMDP benchmark: Measuring and reducing malicious use with unlearning](#). In *Forty-first International Conference on Machine Learning*.
- Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. 2025c. [Safety layers in aligned large language models: The key to LLM security](#). In *The Thirteenth International Conference on Learning Representations*.
- Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. [Contrastive decoding: Open-ended text generation as optimization](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.
- Xuying Li, Zhuo Li, Yuji Kosuga, Yasuhiro Yoshida, and Victor Bian. 2024b. Precision knowledge editing: Enhancing safety in large language models. *arXiv preprint arXiv:2410.03772*.
- Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024. [Tuning language models by proxy](#). In *First Conference on Language Modeling*.
- Guozhi Liu, Weiwei Lin, Tiansheng Huang, Ruichao Mo, Qi Mu, Xiumin Wang, and Li Shen. 2026a. [Surgery: Mitigating harmful fine-tuning for large language models via attention sink](#). In *Forty-third International Conference on Machine Learning*.
- Kangwei Liu, Mengru Wang, Yujie Luo, Lin Yuan, Mengshu Sun, Lei Liang, Zhiqiang Zhang, Jun Zhou, Bryan Hooi, and Shumin Deng. 2026b. [Lookahead tuning: Safer language models via partial answer previews](#). In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining, WSDM '26*, pages 1190–1194, New York, NY, USA. Association for Computing Machinery.
- Ning Lu, Shengcai Liu, Jiahao Wu, Weiyu Chen, Zhirui Zhang, Yew-Soon Ong, Qi Wang, and Ke Tang. 2025. [Safe delta: Consistently preserving safety when fine-tuning LLMs on diverse datasets](#). In *Forty-second International Conference on Machine Learning*.

- Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary Chase Lipton, and J Zico Kolter. 2024. [TOFU: A task of fictitious unlearning for LLMs](#). In *First Conference on Language Modeling*.
- Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. 2024. Harmbench: a standardized evaluation framework for automated red teaming and robust refusal. In *Proceedings of the 41st International Conference on Machine Learning, ICML'24*. JMLR.org.
- Quoc Minh Nguyen, Trung Le, Jing Wu, Anh Tuan Bui, and Mehrtash Harandi. 2026. [Antibody: Strengthening defense against harmful fine-tuning for large language models via attenuating harmful gradient influence](#). In *The Fourteenth International Conference on Learning Representations*.
- Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. [Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering](#). In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.
- ShengYun Peng, Pin-Yu Chen, Jianfeng Chi, Seong-min Lee, and Duen Horng Chau. 2025. [Shape it up! restoring LLM safety during finetuning](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Songping Peng, Zhiheng Zhang, Daojian Zeng, Lincheng Jiang, and Xieping Gao. 2026. [Preventing safety drift in large language models via coupled weight and activation constraints](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 17646–17662, San Diego, California, United States. Association for Computational Linguistics.
- Gabriel Jacob Perin, Runjin Chen, Xuxi Chen, Nina S. T. Hirata, Zhangyang Wang, and Junyuan Hong. 2025. [Lox: Low-rank extrapolation robustifies LLM safety against fine-tuning](#). In *Second Conference on Language Modeling*.
- Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2024. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, volume 2024, pages 30988–31043.
- Nina Rimskey, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. [Steering llama 2 via contrastive activation addition](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.
- Saisab Sadhu, Pratinav Seth, and Vinay Kumar Sankarapu. 2026. [Forgetting that sticks: Quantization-permanent unlearning via circuit attribution](#). *Preprint*, arXiv:2605.15138.
- Leheng Sheng, Changshuo Shen, Weixiang Zhao, Junfeng Fang, Xiaohao Liu, Zhenkai Liang, Xiang Wang, An Zhang, and Tat-Seng Chua. 2026. [Alphasteer: Learning refusal steering with principled null-space constraint](#). In *The Fourteenth International Conference on Learning Representations*.
- Rishub Tamirisa, Bhurugu 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. 2025. [Tamper-resistant safeguards for open-weight LLMs](#). In *The Thirteenth International Conference on Learning Representations*.
- Mengru Wang, Ziwen Xu, Shengyu Mao, Shumin Deng, Zhaopeng Tu, Huajun Chen, and Ningyu Zhang. 2025a. [Beyond prompt engineering: Robust behavior control in LLMs via steering target atoms](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23381–23399, Vienna, Austria. Association for Computational Linguistics.
- Yaxuan Wang, Jiaheng Wei, Chris Yuhao Liu, Jinlong Pang, Quan Liu, Ankit Shah, Yujia Bao, Yang Liu, and Wei Wei. 2025b. [LLM unlearning via loss adjustment with only forget data](#). In *The Thirteenth International Conference on Learning Representations*.
- Yuhui Wang, Rongyi Zhu, and Ting Wang. 2026. [Self-destructive language models](#). In *The Fourteenth International Conference on Learning Representations*.
- Sunghyun Wee, Suyoung Kim, Hyeonjin Kim, Kyomin Hwang, and Nojun Kwak. 2025. Safety-preserving PTQ via contrastive alignment loss. *arXiv preprint arXiv:2511.07842*.
- Chengcan Wu, Zhixin Zhang, Zeming Wei, Yihao Zhang, Xiaokun Luan, and Meng Sun. 2025. Secure LLM fine-tuning via safety-aware probing. *arXiv preprint arXiv:2505.16737*.
- Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, Ruoxi Jia, Bo Li, Kai Li, Danqi Chen, Peter Henderson, and Prateek Mittal. 2025. [SORRY-bench: Systematically evaluating large language model safety refusal](#). In *The Thirteenth International Conference on Learning Representations*.
- Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. 2024. [SafeDecoding: Defending against jailbreak attacks via safety-aware decoding](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5587–5605, Bangkok, Thailand. Association for Computational Linguistics.

- Shuo Yang, Qihui Zhang, Yuyang Liu, Yue Huang, Xiaojun Jia, Kun-Peng Ning, Jia-Yu Yao, Jigang Wang, Dai Hailiang, Yibing Song, and Li Yuan. 2025. [AsFT: Anchoring safety during LLM fine-tuning within narrow safety basin](#). In *NeurIPS 2025 Workshop: Reliable ML from Unreliable Data*.
- Xianjun Yang, Xiao Wang, Qi Zhang, Linda Ruth Petzold, William Yang Wang, Xun Zhao, and Dahua Lin. 2023. Shadow alignment: The ease of subverting safely-aligned language models. *arXiv preprint arXiv:2310.02949*.
- Biao Yi, Tiansheng Huang, Baolei Zhang, Tong Li, Lihai Nie, Zheli Liu, and Li Shen. 2026a. [CTRAP: Embedding collapse trap to safeguard large language models from harmful fine-tuning](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10013–10030, San Diego, California, United States. Association for Computational Linguistics.
- Biao Yi, Jiahao Li, Baolei Zhang, Lihai Nie, Tong Li, Tiansheng Huang, and Zheli Liu. 2026b. [Safe-grad: Gradient surgery for safe llm fine-tuning](#). In *ICASSP 2026 - 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3326–3330.
- Xin Yi, Shunfan Zheng, Linlin Wang, Gerard de Melo, Xiaoling Wang, and Liang He. 2025. [Nlsr: neuron-level safety realignment of large language models against harmful fine-tuning](#). In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI’25/IAAI’25/EAAI’25*. AAAI Press.
- Xin Yi, Shunfan Zheng, Linlin Wang, Xiaoling Wang, and Liang He. 2024. [A safety realignment framework via subspace-oriented model fusion for large language models](#). *Knowledge-Based Systems*, 306:112701.
- Ashkan Yousefpour, Taeheon Kim, Ryan S. Kwon, Seungbeen Lee, Wonje Jeung, Seungju Han, Alvin Wan, Harrison Ngan, Youngjae Yu, and Jonghyun Choi. 2025. [Representation bending for large language model safety](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24073–24098, Vienna, Austria. Association for Computational Linguistics.
- Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jentse Huang, Jiahao Xu, Tian Liang, Pinjia He, and Zhaopeng Tu. 2025. [Refuse whenever you feel unsafe: Improving safety in LLMs via decoupled refusal training](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3149–3167, Vienna, Austria. Association for Computational Linguistics.
- Qiusi Zhan, Richard Fang, Rohan Bindu, Akul Gupta, Tatsunori Hashimoto, and Daniel Kang. 2024. [Removing RLHF protections in GPT-4 via fine-tuning](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 681–687, Mexico City, Mexico. Association for Computational Linguistics.
- Bingjie Zhang, Yibo Yang, Renzhe, Dandan Guo, Jindong Gu, Philip Torr, and Bernard Ghanem. 2026a. [A guardrail for safety preservation: When safety-sensitive subspace meets harmful-resistant null-space](#). In *The Fourteenth International Conference on Learning Representations*.
- Jiawen Zhang, Yangfan Hu, Kejia Chen, Lipeng He, Jiachen Ma, Jian Lou, Dan Li, Jian Liu, Xiaohu Yang, and Ruoxi Jia. 2026b. Understanding and preserving safety in fine-tuned LLMs. *arXiv preprint arXiv:2601.10141*.
- Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. 2024. Negative preference optimization: From catastrophic collapse to effective unlearning. In *First Conference on Language Modeling*.
- Weixiang Zhao, Jiahe Guo, Yulin Hu, Yang Deng, An Zhang, Xingyu Sui, Xinyang Han, Yanyan Zhao, Bing Qin, Tat-Seng Chua, and Ting Liu. 2025a. [AdaSteer: Your aligned LLM is inherently an adaptive jailbreak defender](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24559–24577, Suzhou, China. Association for Computational Linguistics.
- Xuandong Zhao, Will Cai, Tianneng Shi, David Huang, Licong Lin, Song Mei, and Dawn Song. 2025b. [Improving LLM safety alignment with dual-objective optimization](#). In *Forty-second International Conference on Machine Learning*.
- Guanghao Zhou, Panjia Qiu, Cen Chen, Hongyu Li, Jason Chu, Xin Zhang, and Jun Zhou. 2025. [LSSF: Safety alignment for large language models through low-rank safety subspace fusion](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30621–30638, Vienna, Austria. Association for Computational Linguistics.
- Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.
- Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, J Zico Kolter, Matt Fredrikson, and Dan Hendrycks. 2024. [Improving alignment and robustness with circuit breakers](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. [Universal and transferable adversarial attacks on aligned language models](#). *Preprint*, arXiv:2307.15043.

A Method Catalogue

The released library spans RECOVER, HARDEN, STEER, and UNLEARN entry points across its four repair paradigms, together with an INTERPRET and EVALUATE instrumentation layer and a set of audited runtime-guardrail components covering input sanitization, output verification, and policy routing. The large majority of these are faithful to their originating paper; a small remainder are simplified-but-equivalent reductions or variants that should not be cited as their named method. Table 4 lists the methods surveyed as part of the library’s faithfulness audit. The complete per-component roster, with an individual faithfulness verdict for each component, is included with the released library.

B Evaluation Protocol and Reproducibility

The safety-evaluation suite consists of seven benchmarks with their standard judges: Harm-Bench (Mazeika et al., 2024), WildJailbreak (Jiang et al., 2024), judged using WildGuard (Han et al., 2024), AdvBench (Zou et al., 2023), SorryBench (Xie et al., 2025), HEx-PHI (Qi et al., 2024), OR-Bench (Cui et al., 2025), and AI-Luminate (Ghosh et al., 2025a). These benchmarks cover direct-request, adversarial, and over-refusal settings, so that a method cannot achieve a high score simply by refusing all requests. The safety scalar reported throughout this paper is $RR = \frac{1}{7} \sum_b RR_b$, computed under greedy decoding. The capability suite reports IFEval (Zhou et al., 2023) accuracy and WikiText-2 perplexity for every checkpoint, together with a domain-matched anchor: GSM8K (Cobbe et al., 2021) for the math domain, HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for code, MedMCQA (Pal et al., 2022) for medical, and MMLU (Hendrycks et al., 2021) professional law for legal. All capability benchmarks are run using lm-evaluation-harness (Gao et al., 2024) under greedy decoding with the harness’s default few-shot settings.

Every drifted checkpoint is produced using the same fixed recipe: LoRA rank $r=16$, $\alpha=32$, dropout 0.05, target modules $\{q, k, v, o\}_{proj}$, AdamW at learning rate 2×10^{-5} with cosine warmup, one training epoch, bf16 precision, and a fixed random seed of 42. These settings are held constant across the released checkpoint suite, so that the (model, domain) pair is the only variable

that changes between checkpoints. The results accompany SAFETUNE release 0.1.3 under LSAL v1.2. The software versions used throughout are torch 2.8.0+cu128, transformers 4.50, peft 0.13, trl 0.21, vllm 0.11.0 (an optional extra, used for generation during evaluation), lm-evaluation-harness 0.4.8, datasets 3.0, Python 3.12, and CUDA 12.8. Compute for the full pipeline, spanning drift induction, the repair audit, and safety evaluation, runs on an H200 NVL accelerator.

Fine-tuning data for the case studies. We produce both case-study checkpoints with the one-epoch LoRA recipe above; only the corpus changes. We fine-tune the finance copilot on an in-house credit-risk dataset of 1,035 loan-decision examples, each a credit application prompt paired with a structured DECISION/REASON response; this corpus is not part of the public release. We fine-tune the medical assistant on an 8,000-consultation subset of the ChatDoctor HealthCareMagic corpus (lavita/ChatDoctor-HealthCareMagic-100k), the corpus behind the library’s medical drift domain, rendering each example through the model’s chat template with response-only loss masking, as in the drift recipe. Neither corpus contains harmful examples; drift arises from benign domain adaptation alone.

Reproducing without local hardware. A ready-to-run environment with the pinned stack above and no local installation is hosted on Lightning AI (Resources, §8), which we recommend as the lowest-friction path to reproducing the quickstart and single-paradigm examples on a machine without a suitable GPU. Extended reproducibility guidance (per-paradigm hardware profiles, containerized setup) is maintained in the project’s [reproducibility guide](#).

GPU memory by paradigm. Minimum VRAM differs by paradigm because of how many checkpoints a paradigm holds resident at once. HARDEN, UNLEARN, and STEER operate on one loaded model and run on 24 GB-class GPUs for the model sizes in this paper. RECOVER methods that build a safety task vector hold the drifted, base, and aligned reference checkpoints simultaneously (Figure 1) and correspondingly need upward of 64 GB; this is the paradigm’s main practical cost, since its compute cost is otherwise sub-minute on a CPU.

Table 4: Method catalogue for the four intervention paradigms, as surveyed for the library’s faithfulness audit. Per-method audit verdicts ship with the released library; some recent additions have completed only part of the audit (see Limitations). Methods listed without citations are SAFETUNE-original variants, standard baselines, or techniques whose full provenance (including arXiv identifiers) is recorded in the released audit checklist. The full roster, including the INTERPRET and EVALUATE instrumentation layer, is released with the library rather than reproduced here.

Paradigm	Methods surveyed
RECOVER (patch τ_{safe})	Task Arithmetic (Ilharco et al., 2023), RESTA (Bhardwaj et al., 2024), LSSF (Zhou et al., 2025), C- $\Delta\Theta$ (Kasliwal et al., 2026), SafeLoRA (Hsu et al., 2024), SCRUB (Kurmanji et al., 2023), SafeMerge (Djuhera et al., 2026), SafeDelta (Lu et al., 2025), SOMF (Yi et al., 2024), NLSR (Yi et al., 2025), PKE (Li et al., 2024b), MSCP (Han et al., 2025a), LoX (Perin et al., 2025), Antidote (Huang et al., 2025a), Pre-PostMerge (Farn et al., 2025), QReSafe (Chen et al., 2025), AAQ (Wee et al., 2025), SafeReAct (Li et al., 2025a), Sweep
HARDEN (constrained FT)	SafeGrad/Vaccine (Yi et al., 2026b; Huang et al., 2024b), LISA (Huang et al., 2024a), LookAhead/Booster (Liu et al., 2026b; Huang et al., 2025b), TAR (Tamirisa et al., 2025), DeRTa (Yuan et al., 2025), AsFT (Yang et al., 2025), DOOR (Zhao et al., 2025b), Surgery (Liu et al., 2026a), SPPFT (Li et al., 2025c), SAP (Wu et al., 2025), SaLoRA-Harden (Li et al., 2025b), STAR-DSS (Peng et al., 2025), CST, SEAM (Wang et al., 2026), CTRAP (Yi et al., 2026a), MART (Ge et al., 2023), DeepRefusal, Antibody (Nguyen et al., 2026)
UNLEARN (forget)	NPO (Zhang et al., 2024), RMU (Li et al., 2024a), GradientAscent (Maini et al., 2024), FLAT (Wang et al., 2025b), SimNPO (Fan et al., 2025), MANSU (Sadhu et al., 2026)
STEER (inference)	Refusal Direction (Arditi et al., 2024), CAA (Rimsky et al., 2024), Linear-Probe Guard (no single originating paper), SafeDecoding (Xu et al., 2024), Nudging (Fei et al., 2025), AdaSteer (Zhao et al., 2025a), SafeSteer (Ghosh et al., 2025b), AlphaSteer (Sheng et al., 2026), SafeSwitch (Han et al., 2025b), SCANS (Cao et al., 2024), STA (Wang et al., 2025a), CircuitBreaker (Zou et al., 2024), CircuitBreakerRR, RepBend (Yousefpour et al., 2025), CAST (Lee et al., 2025), RRFAEnsemble, ContrastiveDecoding (Li et al., 2023), ProxyTuning (Liu et al., 2024)

C System Usability, Extension, and Engineering Effort

Effort relative to single-paradigm tooling. SAFETUNE’s pipeline (Figure 1, Listings 1 to 4) reduces a repair-and-compare run to one paradigm-uniform script: instantiate a trainer, execute its repair call, and call the shared evaluate(), four to eight executable lines beyond imports for every paradigm in this paper, one repair invocation and one shared evaluation invocation. This differs structurally from single-paradigm harnesses we surveyed (§7). OpenUnlearning, for instance, requires authoring a Hydra experiment config, invoking its evaluation script with explicit model and output paths, and separately producing and referencing a retain-model log file to obtain a comparable metric; a second, differently-configured invocation is needed for a second checkpoint. SAFETUNE folds this into the single evaluate() call already shown in Listings 1 to 4, since every method reports against the same fixed benchmark set and baseline by construction. We report this qualitatively rather than as a controlled timing study, which the

Limitations section notes as future work. Table 5 summarizes the comparison.

Table 5: Practitioner-facing integration effort: a structural comparison of the released interfaces. We did not run a controlled developer-time study.

Operation	Single-paradigm workflow	SAFETUNE
Configure repair	method-specific config	one paradigm contract
Run repair	tool-specific script	one repair call
Safety eval	separate eval path	shared evaluate()
Capability eval	practitioner integration	same harness
Second checkpoint	new configured invocation	same API
Cross-paradigm switch	different tool / API	swap import + class
Repair script	config + glue code	4 to 8 executable lines beyond imports

Registering a new method, worked example. Adding a method touches exactly three files. First, a trainer class inheriting from its paradigm’s base (e.g., `_HardenBase`) and implementing the paradigm’s repair call:

```

class MyMethodTrainer(_HardenBase):
    METHOD = "MyMethod"
    def train(self, train_dataset, out_dir=None, **kw):
        ... # training logic
    return self._save_merged(model, out_dir)

```

Second, re-exporting the class from the paradigm module's `__init__.py`. Third, one line in the registry, `"mymethod": "MyMethodTrainer"`, after which the method is reachable from both the CLI and the Python API; no other file changes. The full walkthrough, including the RECOVER-specific `apply()` contract, is maintained in the project's [developer runbook](#).

Interpretability report contents. The `CircuitInfo` object returned by the interpretability suite (§4) has two parts. `safety_units` lists the implicated layer indices, module names, and per-unit identifiers (e.g., `"L12.MLP.0"`), together with each unit's activation correlation with safety-relevant behavior. `layer_suggestions` names the target modules and, optionally, a restricted layer subset for a downstream circuit-targeted intervention, with a priority weight per module. A practitioner reads the first to see *where* safety-relevant computation is concentrated and the second to decide *which* modules a follow-up RECOVER or HARDEN run should target. Its [reference documentation page](#) gives the full field-by-field contract.