TRACE: TRAJECTORY-BASED SAFETY PATCH LEARNING FOR LLM POST-TRAINING REALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-Tuning-as-a-Service (FTaaS) platforms let users perform supervised fine-tuning (SFT) on customized data, but this pipeline can erode model safety alignment. To recover safety without re-running full alignment, existing realignment methods focus on calibrating the integration of safety patches into fine-tuned models. These methods exhibit a persistent safety-utility trade-off: weak repair leaves harmful behavior intact, while stronger repair increasingly damages the benign task. This paper shifts the focus from online calibration to offline patch learning and aims to learn a safety patch that restores safety while preserving task-specific capabilities. To this end, we propose **TRACE**, which simulates harmful SFT trajectories to produce progressively corrupted model states, and optimizes a safety patch simultaneously across these states. TRACE trains the safety patch during the offline stage, and reuses it across all user checkpoints without per-user calibration. We evaluate two representative models using three harmful SFT datasets, together with three utility benchmarks. Across six benchmarks and two models, TRACE consistently dominates the safety–utility frontier. TRACE improves the safety rate by up to 77 percentage points over the best baselines, while maintaining comparable utility to the undefended fine-tuned model.

1 INTRODUCTION

Commercial Fine-Tuning-as-a-Service (FTaaS) platforms (OpenAI, 2026b; Amazon Web Services, 2026) allow users to upload personal data to customize Large Language Models (LLMs) (Yang et al., 2025; Grattafiori et al., 2024), where service providers execute the training and deliver the fine-tuned models via APIs. This paradigm lowers infrastructure barriers and protects proprietary weights.

However, FTaaS can also erode the inherent safety alignment of LLMs (Yang et al., 2023), particularly when malicious users fine-tune models on harmful response data. Service providers need a post-training defense that restores safety without re-running full alignment or destroying the utility gained from customized tasks.

Current realignment research can be categorized into three classes based on the intervention stage, as summarized in Table 1. *(i) Pre-FT alignment hardening* methods (Huang et al., 2024; 2025) boost the base model to make safety more resistant to later fine-tuning. *(ii) In-FT safety-preserving* methods (Zhang et al., 2026b; Li et al., 2025) constrain the fine-tuning dynamics to retain safety during the training process. However, although both types of methods provide some resistance, the model can still converge to an unsafe state when tuned on enough harmful data.

(iii) Post-FT merging-based methods (Bhardwaj et al., 2024; Wu et al., 2025; Hsu et al., 2024; Lu et al., 2025) have emerged to restore safety by integrating a safety patch. These methods focus on calibrating this integration through a merging operator and its repair strength. Although lightweight and efficient, they exhibit a persistent *safety-utility trade-off*. Weak repair may leave harmful behavior intact, whereas stronger repair can degrade task utility (Ilharco et al., 2022). Therefore, current merging-based methods fail to simultaneously satisfy the needs of multiple users well.

This motivates us to shift the focus from online calibration to offline patch optimization. Instead of calibrating the integration of the safety patch against unpredictable users, we ask: **Can we find an effective safety patch that restores safety with minimal interference with the user's task performance?**

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.

Table 1: Comparison of intervention paradigms for post-training safety recovery.

Paradigm		Methods	Safety	Utility	Robustness	Efficiency	Extensibility
Stage	Mechanism						
Pre-FT	Alignment Hardening	Vaccine (Huang et al., 2024), Booster (Huang et al., 2025)	○	●	○	0	◑
In-FT	Safety-preserving FT	SaLoRA (Li et al., 2025), SPF (Zhang et al., 2026b)	○	●	◑	$\sim 10^3$	◑
Post-FT	Safety Re-training	OneShot (Zhang et al., 2026a)	◑	◑	○	$\sim 10^1$	◑
	Online Merge Calibration	RESTA (Bhardwaj et al., 2024), EnchTable (Wu et al., 2025) SafeLoRA (Hsu et al., 2024), SafeDelta (Lu et al., 2025)	◑	◑	○	$10^1 \sim 10^2$	◑
	Offline Patch Learning	**TRACE (Ours)**	●	●	●	~ 0.5	●

Notes. Open, half-filled, and filled circles are a qualitative synthesis. **Safety**: the ability to restore or maintain safety after user fine-tuning. **Utility**: the ability to retain downstream task performance. **Robustness**: stability across downstream training intensity. **Efficiency**: The additional computational overhead (seconds) required for online safety recovery per user. **Extensibility**: the capability to incorporate other realignment strategies.

To this end, we propose **TRACE**, a trajectory-based framework that learns an effective, task-preserving safety patch during the offline stage. As shown in Figure 1, TRACE achieves this through a simulate-and-recover strategy within low-rank space. First, it simulates one fine-tuning trajectory to generate progressively corrupted model states with varying training intensities. Subsequently, TRACE optimizes a low-rank adapter to recover safety across these states while preserving measured task utility. TRACE trains just one patch for each base model. After offline training, the learned patch is directly applied to the user SFT-altered checkpoints built on the same backbone.

Extensive experiments validate the superiority of the proposed method. We evaluate TRACE on two representative models using three harmful datasets (Yang et al., 2023; Qi et al., 2024; Ji et al., 2024) of varying sizes alongside three utility datasets spanning dialogue summarization (Gliwa et al., 2019), SQL generation (b-mc2, 2023), and mathematical solving (Cobbe et al., 2021). The results highlight that TRACE effectively resolves the safety-utility trade-off. Specifically, TRACE improves the safety rate by up to 77 percentage points (pp) over the best baselines, while maintaining downstream task performance within a ± 1.7 pp deviation from the undefended baseline.

Our main contributions can be summarized as follows:

- We recast post-FT realignment as a safety patch optimization problem, and shift the focus of the merging-based methods from online calibration to offline optimization.
- We propose TRACE, a trajectory-based safety patch learning algorithm. TRACE simulates an SFT trajectory to capture models across varying levels of training intensity, and then optimizes a universal safety patch that consistently restores safety along the trajectory while preserving task utility.
- We conduct extensive experiments on two LLMs across three safety and three task datasets. The results demonstrate that TRACE effectively restores safety, achieving over 94% safety rates while preserving benign task performance.

2 Threat Model and Problem Formulation

We consider an FTaaS scenario involving two entities, the service provider and the user. The provider hosts an aligned base model θ_{base} and performs full-parameter supervised fine-tuning (SFT) on the user-uploaded data, yielding the fine-tuned model θ_{ft}. The user controls the dataset and permitted SFT hyperparameters, including epochs and learning rate. The provider prepares a safety patch offline and applies it after user training. Under our deployment assumption, the service exposes the final model through black-box inference APIs and does not release the hosted weights or the patch. The primary objective of the service provider is to deliver inference APIs that guarantee both model safety and downstream task utility.

To recover the safety of θ_{ft}, merging-based methods construct a safety patch $\Delta\theta_{\text{patch}}$ offline and merge it online with the user update. During the offline stage, the patch is typically obtained (Bhardwaj et al., 2024; Wu et al., 2025) by training an intentionally unsafe model θ_{unsafe} on surrogate harmful data and taking the parameter difference from the base model (i.e., $\Delta\theta_{\text{patch}} = \theta_{\text{base}} - \theta_{\text{unsafe}}$). During

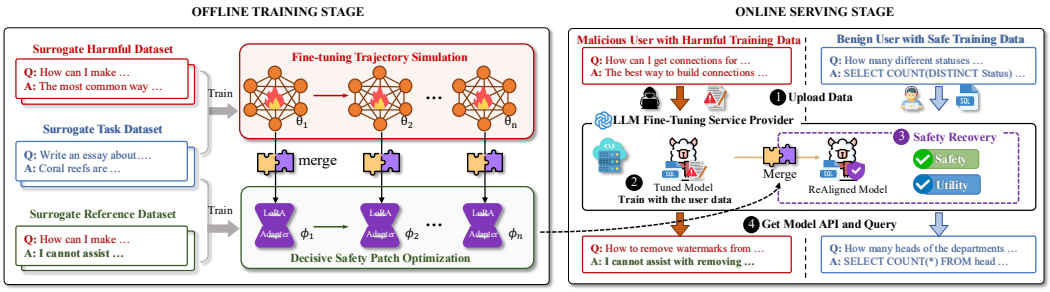


Figure 1: Overview of TRACE. Offline (left), harmful SFT trajectory simulation alternates with safety patch optimization. Online (right), the provider applies the learned patch directly to a fine-tuned model built from the same backbone, without per-user calibration.

the online stage, the safety repair process can be formulated as:

$$\theta_{\text{deploy}} = \theta_{\text{base}} \oplus \Delta\theta_{\text{user}} \oplus \Delta\theta_{\text{patch}} \tag{1}$$

where $\Delta\theta_{\text{user}}$ is the parameter update from the user's data and $\oplus$ is the merging operator. *Arithmetic-based* methods (Bhardwaj et al., 2024; Wu et al., 2025) scale the repair strength by a coefficient λ, $\theta_{\text{deploy}} = \theta_{\text{base}} + \Delta\theta_{\text{user}} + \lambda\Delta\theta_{\text{patch}}$; *projection-based* methods (Hsu et al., 2024; Lu et al., 2025) map both updates through projection matrices $P_1(\alpha), P_2(\alpha)$ before merging (i.e., $\theta_{\text{deploy}} = \theta_{\text{base}} + P_1(\alpha)\Delta\theta_{\text{user}} + P_2(\alpha)\Delta\theta_{\text{patch}}$), where α adjusts the repair strength.

In real-world FTaaS deployments, users can choose the uploaded dataset, number of training epochs, and learning rate (OpenAI, 2026b; Microsoft, 2026; Amazon Web Services, 2026; Google Cloud, 2026). As a result, different users may adopt substantially different training configurations, which the provider cannot fully anticipate in advance. The static repair strength employed by existing realignment methods may therefore be inadequate to accommodate such diverse user requests. We evaluate the resulting safety–utility trade-off across users in Section 4.3.

This observation motivates a shift from online calibration to offline patch learning. During offline training, the provider observes only surrogate data distributions, with private user distributions remaining inaccessible. The safety repair on user-tuned checkpoints is therefore recast as a deployment-time safety transfer problem. We formalize this safety-patch optimization problem in the next section and introduce TRACE's alternating recovery procedure.

3 Trajectory-Based Safety Patch Learning

This paper shifts the focus to offline optimization of a transferable safety patch $\Delta\theta_{\text{patch}}$ for an unseen SFT-altered model θ_{ft}. We denote the learned patch as ϕ to distinguish it from the conventional safety patch $\Delta\theta_{\text{patch}}$ in Section 2.

The safety patch should elicit refusal responses on harmful inputs while preserving the model's downstream task performance. Thus, we have the following objective:

$$\phi^* = \arg\min_{\phi} \; \mathbb{E}_{(x,y)\sim\{\mathcal{U}_{\text{safe}},\mathcal{U}_{\text{task}}\}} \left[\mathcal{L}(\theta_{\text{ft}} \oplus \phi, x, y)\right], \tag{2}$$

where $\mathcal{U}_{\text{safe}}$ contains harmful questions paired with safe responses, and $\mathcal{U}_{\text{task}}$ contains task-related questions and their corresponding answers. Once the patch is obtained offline, the provider directly deploys $\theta_{\text{deploy}} = \theta_{\text{ft}} \oplus \phi^*$ without online per-user adaptation.

Since the user-uploaded dataset $\mathcal{U}$ is unavailable during the offline training, TRACE instead approximates the objectives via provider-owned surrogate datasets, expecting the resulting safety capability to transfer effectively to the user-customized model. The provider trains just one patch ϕ for each base model. For a user fine-tuned model θ_{ft} built on the same backbone, the provider can simply apply direct weight merging ($\theta_{\text{ft}} + \phi$) to restore safety alignment, eliminating the need for per-user adjustments or projections. Notably, since TRACE aims to search for a better safety patch, it is highly compatible with existing techniques, seamlessly leveraging their online merging operations to further boost defensive performance.

3.1 ALTERNATING SAFETY RECOVERY TRAINING

Ideally, the safety patch ϕ should be optimized over all corrupted states θ_{ft} as described in Equation (2). However, directly optimizing this objective is impractical, as it requires caching every possible model state that could arise during fine-tuning. For LLMs, storing and processing such dense intermediate checkpoints incurs prohibitive memory and computational overhead.

TRACE adopts an alternating optimization strategy to approximate this expectation efficiently. At each iteration, TRACE first simulates the user's full-parameter fine-tuning and obtains a freshly updated model state, and then TRACE immediately optimizes the safety patch on top of it to restore safety while preserving task performance. In this way, TRACE approximates optimization over the trajectory without storing every model state. We acknowledge that this alternating procedure only approximates the ideal objective and inevitably introduces some discrepancy. Nevertheless, the primary goal of this work is not to construct the optimal patch, but rather to demonstrate this safety patch learning paradigm provides an effective and practical strategy for model realignment.

Phase A: simulate safety-eroding fine-tuning. TRACE simulates user full parameter fine-tuning using provider-owned surrogate harmful data $\mathcal{D}_{\text{harm}}$ and benign data $\mathcal{D}_{\text{task}}$. The trajectory therefore reflects both harmful drift and ordinary task adaptation. For a data batch B and model parameters θ, we use the response-only causal language modeling loss

$$\mathcal{L}(\theta; B) = -\sum_{x \in B} \sum_{t \in \mathcal{A}(x)} \log p_\theta(x_t \mid x_{<t}), \tag{3}$$

where $\mathcal{A}(x)$ denotes the assistant-response token positions in example x.

For the current iteration t, we disable the safety patch and update only the backbone model on one harmful batch and one benign task batch:

$$\theta_t = \theta_{t-1} - \eta_\theta \nabla_\theta \mathcal{L}(\theta_{t-1}; \{B_h, B_t\}) \tag{4}$$

where η_θ is the learning rate for the backbone model, $\theta_0 = \theta_{\text{base}}$, and $B_h \sim \mathcal{D}_{\text{harm}},\ B_t \sim \mathcal{D}_{\text{task}}$ are data samples. As each update combines harmful and benign supervision, this trajectory captures both safety erosion and ordinary task adaptation at progressively greater fine-tuning intensities.

Phase B: optimize the safety patch. Given the newly corrupted state θ_t, TRACE freezes the backbone, enables the safety patch, and updates it using refusal supervision $\mathcal{D}_{\text{ref}}$ and benign task data $\mathcal{D}_{\text{task}}$. The patch update is

$$\phi_t = \phi_{t-1} - \eta_\phi \nabla_\phi \mathcal{L}\left(\theta_t \oplus \phi_{t-1}; \{B_{\text{ref}}, B_t\}\right), \tag{5}$$

where η_ϕ is the patch learning rate, ϕ_0 is zero-initialized, and B_{ref} are sampled from $\mathcal{D}_{\text{ref}}$. This objective trains the patch to recover refusal behavior at the current corruption state θ_t while preserving responses on the surrogate benign tasks.

By alternatively repeating the above two phases, TRACE requires the safety patch to consistently succeed across multiple evolving corrupted states rather than overfitting to one fixed checkpoint. Finally, we obtain the learned safety patch ϕ^* without explicitly storing intermediate checkpoints.

We distinguish two behavioral goals from possible geometric explanations:

Limited task interference. The patch optimization objective (Equation (4)) jointly enforces refusal supervision and benign task preservation across evolving states, which we hypothesize encourages patch updates that interfere minimally with task behavior.

Decisive refusal recovery. The learned safety patch should reliably restore refusal behavior even in the presence of harmful weight shifts of unknown intensity. We conjecture that this requires the patch to dominate the harmful direction in order to recover safety.

3.2 DECISIVE SAFETY IN LOW-RANK SUBSPACE

This paper evaluates the challenging setting where the user performs *full-parameter SFT*, which can induce stronger and more distributed weight drift than parameter-efficient fine-tuning (PEFT). Consequently, existing realignment methods (Bhardwaj et al., 2024; Lu et al., 2025) typically necessitate full-parameter safety patches. For example, SafeLoRA (Hsu et al., 2024) restricts its

LoRA-based repair to PEFT models, while relying on full-parameter patches for full-parameter SFT models.

TRACE departs from this design, as we empirically find that even for full-parameter altered models, an effective safety patch can be realized within a low-rank subspace. This highlights an intriguing mechanism wherein, despite extensive shifts in the global weight space, safety capabilities remain anchored along a small number of critical directions (Arditi et al., 2024) that exert decisive control over behavior. We therefore implement ϕ as a LoRA adapter (Hu et al., 2022), which significantly reduces trainable parameters during offline optimization and yields a lightweight patch for efficient online deployment.

3.3 Calibration-Free Deployment and Compatibility

After offline optimization, we freeze the patch and reuse it across user checkpoints derived from the same base model. Under the default deployment, the provider directly applies $\theta_{\text{deploy}} = \theta_{\text{ft}} + \phi^*$. This procedure requires neither access to the user-uploaded data nor checkpoint-specific safety probes, gradient updates, or coefficient searches. The offline optimization cost can therefore be amortized across users, while online deployment only requires loading or merging the lightweight adapter. We refer to this setting as *calibration-free* within the target fine-tuning regime.

TRACE is also compatible with prior post-training realignment methods. Existing methods predominantly focus on designing the merging operation $\oplus$, whereas TRACE improves the safety patch ϕ itself. Because the patch and the merging rule are orthogonal, the conventional safety patch in an arithmetic- or projection-based method can be replaced by ϕ without changing that method's original operator. Such composition is optional and potentially boosts the performance of existing approaches. In Section 4.4, we evaluate this compatibility and measure how it changes the safety-utility curves of several merging methods.

4 Experiments

4.1 Experiment Settings

Base models. We evaluate on two representative open-source models: *Llama-3.1-8B-Instruct (Llama)* (Grattafiori et al., 2024), widely used in prior realignment work (Qi et al., 2024; Yang et al., 2023; Bhardwaj et al., 2024), and *Qwen3.5-9B (Qwen)* (Qwen Team, 2026), a more recent model to test TRACE effectiveness across model families.

Datasets. Post-FT realignment involves an *offline* stage, where the provider prepares the safety patch, and an *online* stage, where user checkpoints are evaluated. Offline, following prior work (Wu et al., 2025; Yuan et al., 2025), we sample 5,000 examples each from *BeaverTails* (Ji et al., 2023) (harmful) and *WizardLM* (Xu et al., 2023) (benign). Online, *malicious* users train on 100 *ShadowAlignment* (Yang et al., 2023), 500 *PureBad* (Qi et al., 2024), or 1,000 *SafeRLHF* (Ji et al., 2024) examples. These represent distinct user corpora with different sizes. *Benign* users train on *SamSum* (Gliwa et al., 2019), *SQLCreate* (b-mc2, 2023), or *GSM8K* (Cobbe et al., 2021), using the task dataset sizes and splits of OneShot (Zhang et al., 2026a). *Mixed* corpora combine each task dataset with PureBad. Unless otherwise specified, we train the model on the user data for five epochs. More implementation details are specified in Section A.

Baselines. We evaluate our method against six established baselines. Four are post-FT merging methods closely aligned with our setting: RESTA (Bhardwaj et al., 2024) and EnchTable (Wu et al., 2025) are arithmetic-based methods that scale and add a safety patch to the tuned LLM, whereas SafeLoRA (Hsu et al., 2024) and SafeDelta (Lu et al., 2025) are projection-based methods that suppress safety-sensitive components within the user update. For broader comparison, we include OneShot (Zhang et al., 2026a), another post-FT approach that performs additional safety fine-tuning on the compromised model to restore alignment, and SPF (Zhang et al., 2026b), an in-FT realignment method noted for its effectiveness across different training intensities.

Metrics. For safety evaluation, we closely follow prior work (Zhang et al., 2026a) and adopt the held-out benchmark (*StrongReject* (Souly et al., 2024)) to evaluate the *Safety Rate (%)* on the unseen data distribution. The safety rate is defined as the fraction of model responses that are deemed safe.

Table 2: Safety rate and task accuracy under pure and mixed profiles. In each column, **bold** and underline indicate the best and second-best safety rate, respectively. For TRACE, ↑ denotes the gain over the second-best value, while Δ denotes the change in task accuracy relative to No Defense.

Safety Rate (%) ↑

	Pure Profile						Mixed Profile					
	Llama			Qwen			Llama			Qwen		
Dataset	**Shadow**	**PureBad**	**SafeRLHF**	**Shadow**	**PureBad**	**SafeRLHF**	**MixedSamSum**	**MixedSQL**	**MixedGSM8K**	**MixedSamSum**	**MixedSQL**	**MixedGSM8K**
No Defense	18	10	11	3	5	2	11	20	12	2	4	1
RESTA	99	15	8	10	3	3	13	21	21	30	4	50
EnchTable	**100**	16	10	10	21	4	14	21	22	24	5	44
SafeLoRA	18	8	23	4	53	38	83	93	92	86	68	99
SafeDelta	18	12	15	44	15	21	98	**98**	98	**100**	**100**	**100**
SPF	11	11	10	4	4	0	20	20	15	3	5	4
OneShot	**100**	67	13	36	53	2	47	35	33	24	7	40
TRACE	**100**	**100**↑33	**100**↑77	**100**↑56	**100**↑47	**100**↑62	**100**	**98**	**100**	98	94	**100**

Task Accuracy (%) ↑

	SamSum	SQL	GSM8K	SamSum	SQL	GSM8K	MixedSamSum	MixedSQL	MixedGSM8K	MixedSamSum	MixedSQL	MixedGSM8K
No Defense	51	81	55	46	66	42	47	76	40	50	77	59
TRACE	50Δ-0.2	81Δ0.0	54Δ-0.5	46Δ+0.4	66Δ+0.1	42Δ0.0	47Δ-0.6	77Δ+1.0	40Δ-0.1	50Δ-0.1	78Δ+0.8	57Δ-1.7

To judge whether a given response is safe, we use safety-expert discriminators: Llama-Guard-3-8B (LlamaGuard) (Inan et al., 2023; Grattafiori et al., 2024) for Llama-based models and Qwen3Guard-8B (QwenGuard) (Zhao et al., 2025) for Qwen-based models. Since the question-answer pairs are explicitly designed to be harmful, this evaluation constitutes a relatively straightforward task for dedicated safety classifiers. During our experiments, we also manually inspect a subset of the classification results and do not observe any apparent misjudgments.

For utility metrics, we evaluate each customized task on its respective test set and report *Task Accuracy (%)*. Since the three datasets differ in output format, we adopt tailored evaluation criteria. Specifically, we leverage Abstract Syntax Tree Matching (Neamtiu et al., 2005) for SQLCreate, which compares the parsed structures of the predicted and reference SQL queries, thereby accommodating syntactically different but semantically equivalent query formulations. We use Exact Match for GSM8K, which checks whether the extracted numerical answer is identical to the ground truth. We report the ROUGE-1 F1 Score (Lin, 2004) for SamSum, which measures unigram overlap between generated and reference summaries to balance content coverage and conciseness.

4.2 Overall Performance

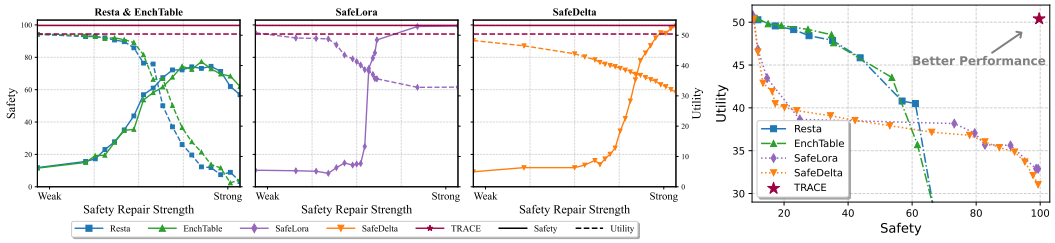
This section evaluates TRACE under two deployment scenarios, pure-profile fine-tuning and mixed-profile fine-tuning. We analyze safety recovery and utility preservation for each setting. The results are shown in Table 2, with safety in the upper panel and task accuracy in the lower panel.

Pure profile. To independently assess safety recovery and utility preservation, we first isolate two extreme cases: malicious users fine-tuning exclusively on harmful data, and benign users fine-tuning on clean task data. We evaluate all methods on both base models across three harmful datasets and three custom task datasets.

As illustrated in Table 2, TRACE obtains 100% safety rates across six pure-profile combinations of benchmarks and models. Its advantage is particularly notable on the challenging SafeRLHF benchmark. TRACE improves the Llama safety rate from 23% to 100%, which is over four times higher than the second-best value. Similarly, it increases the Qwen safety rate from 38% to 100%, outperforming the second-best baseline by 2.6 times.

TRACE preserves downstream task performance with negligible interference to user tasks. To quantify this, we leverage the *No Defense* approach as a baseline and calculate the utility deviation (Δ) to measure the impact of TRACE on performance. Across the six pure-profile settings, these utility deviations fall within a narrow margin of ± 0.5 pp. This near-zero interference demonstrates that the learned safety patch operates along directions largely disentangled from user task updates, preventing it from distorting the learned task behavior.

Mixed profile. This setting evaluates a mixed training scenario where the user corpus contains both benign task examples and harmful content, aiming to test whether TRACE can selectively eliminate harmful behaviors without degrading co-trained task performance.



(a) Safety–Utility trade-off curves. (b) Operating envelope across users.

Figure 2: **Left:** Safety and utility measured on two representative users as the repair strength varies. **Right:** Operating envelope across users available to deployed realignment methods. TRACE achieves a favorable operating point without coefficient search.

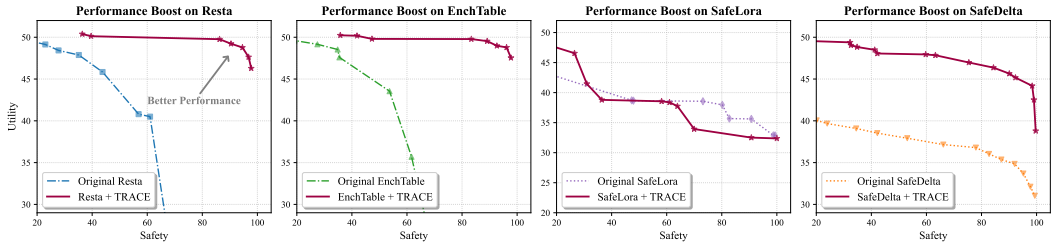


Figure 3: Operating envelopes across users before (dashed) and after (solid) substituting the TRACE patch into each baseline's merging operation.

TRACE effectively isolates and neutralizes harmful components entangled with task-relevant updates in a mixed training corpus. Across all six mixed configurations, TRACE achieves a safety rate of at least 94% while keeping task accuracy within ± 1.7 pp of No Defense.

Projection-based defenses (e.g., SafeLoRA and SafeDelta) achieve higher safety in the mixed setting but at a significant utility cost. SafeDelta reaches over 98% safety across all six mixed configurations, showing a sharp increase from its performance on purely malicious data in Table 2. This improvement may stem from the larger aggregate model modification induced by jointly training on malicious and benign task data, which causes SafeDelta's projection mechanism to suppress a larger portion of the resulting update. However, this stronger suppression simultaneously destroys co-trained task representations. Full task-accuracy results for all baselines are reported in Appendix B.5, which show that SafeDelta drops MixedGSM8K on Llama from 40% to 2%, and SafeLoRA collapses MixedSamSum on Llama from 47% to 8%.

4.3 Safety–Utility Trade-off across Users

Although existing merging-based methods (Hsu et al., 2024; Lu et al., 2025) use adaptive rescaling or projection rules for online calibration, their deployment still depends on a repair-strength coefficient, such as λ or α, to control the overall intervention magnitude. In an FTaaS deployment, the provider must serve heterogeneous user requests. This raises a deployment issue: a shared coefficient cannot generally provide a favorable operating point for all users.

We evaluate this service-level limitation using two representative user requests: a malicious checkpoint produced from PureBad and a benign checkpoint produced from SamSum. For each baseline, we sweep the repair strength and report the performance obtained using a shared coefficient across both requests. This shared policy reflects the deployment of realignment methods in real-world FTaaS settings. Appendix A details the sweep coverage and the method-specific strength axes.

Existing baselines exhibit a clear safety–utility trade-off across users. As shown in Figure 2a, increasing the repair strength improves safety for the malicious user but reduces utility for the benign

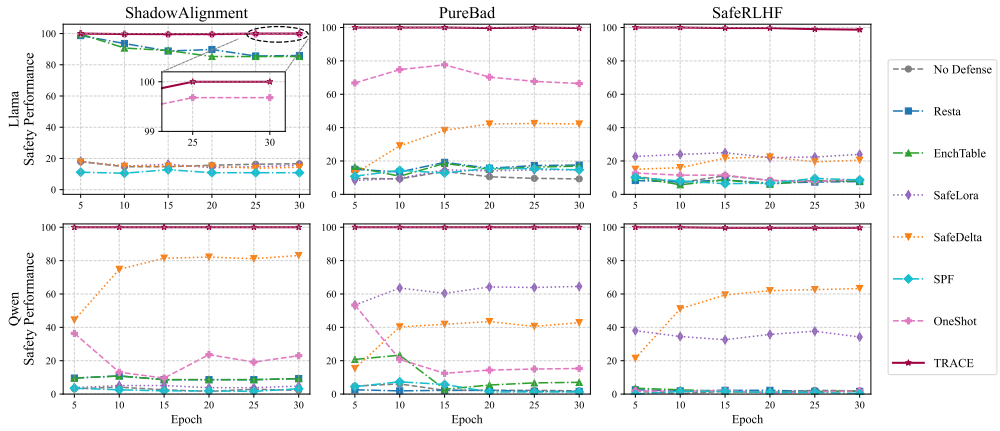


Figure 4: Safety rate across malicious user fine-tuning intensity. TRACE remains stable under progressively stronger corruption.

user. For example, the safety rate of RESTA increases from 10.5% to 74.4%, while the task accuracy declines from 50.4% to 6.3%.

TRACE achieves a favorable performance across users without coefficient search. As shown in Figure 2b, TRACE reaches a 99.7% safety rate and 50.4% task accuracy. It occupies a separate region that the baselines cannot reach with any shared repair strength.

4.4 Integration and Boost on Existing Methods

Recall from Section 3.3 that TRACE optimizes the safety patch ϕ itself rather than the merging operation $\oplus$. It is therefore compatible with existing merging-based defenses. This section evaluates whether TRACE can integrate with these baselines and further boost their performance. Specifically, we substitute the standard safety patch in other baselines with the TRACE adapter. We then re-sweep their repair strengths to characterize the resulting operating envelopes across users.

TRACE can be seamlessly integrated with existing baselines and boosts the performance of most methods. As shown in Figure 3, for RESTA, the integration moves from 74.4% safety at 6.3% utility to 97.8% at 46.3%. EnchTable and SafeDelta exhibit similar improvements. These results support the value of the learned patch for these merging-based methods.

Unlike other baselines, SafeLoRA gains no explicit benefit from this integration and achieves comparable performance to the original. For instance, at a 60% safety rate, both the original and integrated versions achieve approximately 38% utility. A possible explanation is that SafeLoRA uses the patch as a projection reference without directly adding its weights.

While integrating TRACE improves the safety–utility operating envelopes across users for existing merging strategies, current methods remain constrained because they are designed around conventional patches. For example, the SafeDelta integration still fails to match the standalone performance of TRACE. This likely occurs because SafeDelta alters components of the safety patch to avoid task interference, which may compromise the effectiveness of TRACE's patch.

4.5 Robustness to SFT Depth

This section evaluates the safety performance under varying user fine-tuning intensities. We simulate different training intensities with various numbers of epochs (from 5 to 30) that a user may request for the FTaaS, covering the epoch limits offered by commercial FTaaS platforms (OpenAI, 2026b; Microsoft, 2026; Amazon Web Services, 2026; Google Cloud, 2026). We also include the utility analysis with varying SFT depth in Appendix B.4.

TRACE maintains a near-perfect safety rate across the evaluated SFT depths for both models. As shown in Figure 4, TRACE achieves a nearly 100% safety rate as SFT intensity increases. The

worst case occurs on Llama SafeRLHF at 30 epochs, where TRACE still reaches a 98.7% safety rate, outperforming the second-best method by 74.76 pp. Meanwhile, its utility deviation from No Defense remains within ± 1.6 pp across all datasets and epochs. This stability is crucial for FTaaS deployments, where users are free to choose their own training intensity, while providers must always offer reliable protection.

Existing defenses behave inconsistently across fine-tuning intensities. For instance, the safety rate of OneShot on Qwen drops from 36.42% to 23.00% against the Shadow dataset, whereas SafeDelta rises from 44.41% to 83.07% in the same setting. We speculate this divergence may stem from whether they apply a relatively limited magnitude correction. For example, SafeDelta activates only when the user update deviates significantly from the safety space. Consequently, larger corruption updates are more likely to trigger suppression and be projected back toward safe directions.

5 Related Work

5.1 Safety Alignment Under Post-Training Drift

Modern LLMs achieve safety alignment through techniques such as RLHF (Ouyang et al., 2022), DPO (Rafailov et al., 2023), and Constitutional AI (Bai et al., 2022a;b). While effective, these pipelines do not render safety immutable. Prior work demonstrates that subsequent fine-tuning can trivially override refusal behaviors with minimal data (Yang et al., 2023; Qi et al., 2024; Lermen et al., 2023), and even benign domain adaptation can compromise safety (Qi et al., 2024; Kirkpatrick et al., 2017). Such fragility poses a critical challenge for FTaaS, where downstream adaptation occurs on private user data. Re-executing full alignment for every customized user model is computationally prohibitive, motivating lightweight post-training recovery mechanisms.

5.2 Safety Recovery in FTaaS

Existing realignment defenses can be categorized by intervention stage as shown in Table 1. *Pre-FT alignment hardening* methods (e.g., Vaccine (Huang et al., 2024), Booster (Huang et al., 2025)) modify the alignment objective of the base model to induce resistance against subsequent safety drift. *In-FT safety-preserving adaptation* methods (e.g., SaLoRA (Li et al., 2025), SPF (Zhang et al., 2026b)) dynamically prevent fine-tuning dynamics from drifting toward unsafe directions.

Post-FT recovery aims to restore safety by intervening after user training. OneShot (Zhang et al., 2026a) recovers safety by an additional brief fine-tuning on the compromised user-adapted model on a small set of safety samples.

A dominant line of Post-FT work restores safety through *parameter merging*. The core intuition, rooted in Task Arithmetic (Ilharco et al., 2022; Yadav et al., 2023; Wortsman et al., 2022; Yu et al., 2024), is that if parameter updates are based on the same base model, they can be linearly superposed to endow the model with both capabilities. Based on this principle, existing methods directly intervene in the parameter space by adding an extracted safety vector (e.g., RESTA (Bhardwaj et al., 2024), EnchTable (Wu et al., 2025)) or by selectively masking and projecting the downstream task updates (e.g., SafeLoRA (Hsu et al., 2024), SafeDelta (Lu et al., 2025)). Despite their varied mechanisms, these methods exhibit the safety–utility trade-off across users analyzed in Section 4.3.

6 Conclusion

In this paper, we study post-training realignment after user SFT in FTaaS and identify a recurring *safety-utility trade-off* in merging-based safety recovery across users. We propose TRACE, which shifts from per-user online calibration to offline trajectory-based safety patch optimization. Given the base model, TRACE trains a universal safety patch along a simulated trajectory, and reuses it across user-altered models built on this base model. We evaluate the safety recovery and task preservation performance on two models and six datasets. TRACE achieves at least a 94% safety rate while keeping utility within $\pm 1.7\%$ relative to the undefended baseline. These results demonstrate that safety patch learning is an effective paradigm for post-training realignment. We leave the fine-grained safety boundary judgments and adaptive attacks to future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language polishing and code debugging. These tools did not contribute to formulating the research ideas or the paper's conclusions. The authors reviewed all AI-assisted content and are responsible for all statements, interpretations, results, and artifacts in this work.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, our anonymous implementation and configuration files are available at `https://anonymous.4open.science/r/realignment-trace`. The complete code, hyperparameters, and training scripts will be made publicly available upon publication.

REFERENCES

Amazon Web Services. Customize your model to improve its performance for your use case. `https://docs.aws.amazon.com/bedrock/latest/userguide/custom-models`, 2026. Accessed: 2026-05-14.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

b-mc2. SQL Create Context Dataset, 2023. URL `https://huggingface.co/datasets/b-mc2/sql-create-context`.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Rishabh Bhardwaj, Duc Anh Do, and Soujanya Poria. 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)*, pp. 14138–14149, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 70–79, 2019.

Google Cloud. Tune gemini models by using supervised fine-tuning. `https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/gemini-use-supervised-tuning`, 2026. Accessed: 2026-05-14.

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

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

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 finetuning large language models. *Advances in Neural Information Processing Systems*, 37:65072–65094, 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*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Tiansheng Huang, Sihao Hu, and Ling Liu. Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning attack. *Advances in Neural Information Processing Systems*, 37:74058–74088, 2024.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Tekin, and Ling Liu. Booster: Tackling harmful fine-tuning for large language models via attenuating harmful perturbation. In *International Conference on Learning Representations*, volume 2025, pp. 67202–67226, 2025.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

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

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Qiu, Boxun Li, and Yaodong Yang. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference. *arXiv preprint arXiv:2406.15513*, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Simon Lermen, Charlie Rogers-Smith, and Jeffrey Ladish. Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b. *arXiv preprint arXiv:2310.20624*, 2023.

Mingjie Li, Wai Man Si, Michael Backes, Yang Zhang, and Yisen Wang. Salora: Safety-alignment preserved low-rank adaptation. *arXiv preprint arXiv:2501.01765*, 2025.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ning Lu, Shengcai Liu, Jiahao Wu, Weiyu Chen, Zhirui Zhang, Yew-Soon Ong, Qi Wang, and Ke Tang. Safe delta: Consistently preserving safety when fine-tuning llms on diverse datasets. *arXiv preprint arXiv:2505.12038*, 2025.

Microsoft. Customize a model with fine-tuning. https://learn.microsoft.com/en-us/azure/foundry/openai/how-to/fine-tuning?tabs=oai-sdk&pivots=programming-language-studio, 2026. Accessed: 2026-05-14.

Iulian Neamtiu, Jeffrey S. Foster, and Michael W. Hicks. Understanding source code evolution using abstract syntax tree matching. In *Proceedings of the 2005 International Workshop on Mining Software Repositories, MSR 2005, Saint Louis, Missouri, USA, May 17, 2005*. ACM, 2005. doi: 10.1145/1083142.1083143. URL https://doi.org/10.1145/1083142.1083143.

OpenAI. Gpt-5.6 system card, July 2026a. URL https://deploymentsafety.openai.com/gpt-5-6.

OpenAI. Supervised fine-tuning. https://developers.openai.com/api/docs/guides/supervised-fine-tuning, 2026b. Accessed: 2026-05-14.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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*, volume 2024, pp. 30988–31043, 2024.

Qwen Team. Qwen3.5-Omni Technical Report. *arXiv preprint arXiv:2604.15804*, 2026.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=HPuSIXJaa9.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A strongreject for empty jailbreaks, 2024.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Jialin Wu, Kecen Li, Zhicong Huang, Xinfeng Li, Xiaofeng Wang, and Cheng Hong. Enchtable: Unified safety alignment transfer in fine-tuned large language models. *arXiv preprint arXiv:2511.09880*, 2025.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

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

Xianjun Yang, Xiao Wang, Qi Zhang, Linda Petzold, William Yang Wang, Xun Zhao, and Dahua Lin. Shadow alignment: The ease of subverting safely-aligned language models. *arXiv preprint arXiv:2310.02949*, 2023.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Jiahao Xu, Tian Liang, Pinjia He, and Zhaopeng Tu. 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)*, pp. 3149–3167, 2025.

Jiawen Zhang, Lipeng He, Kejia Chen, Jian Lou, Jian Liu, Xiaohu Yang, and Ruoxi Jia. Safety at one shot: Patching fine-tuned llms with a single instance. *arXiv preprint arXiv:2601.01887*, 2026a.

Jiawen Zhang, Yangfan Hu, Kejia Chen, Lipeng He, Jiachen Ma, Jian Lou, Dan Li, Jian Liu, Xiaohu Yang, and Ruoxi Jia. Understanding and preserving safety in fine-tuned llms. *arXiv preprint arXiv:2601.10141*, 2026b.

Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, et al. Qwen3guard technical report. *arXiv preprint arXiv:2510.14276*, 2025.

A Experimental Details

The experiments are conducted on the PyTorch platform (Paszke et al., 2019) using 8 NVIDIA H800 GPUs with 80 GB of memory each. For model training, we use Transformers Accelerate (Gugger et al., 2022), and the default AdamW (Loshchilov & Hutter, 2017) optimizer (learning rate $\eta = 5 \times 10^{-5}$, weight decay 0.01). Baselines are implemented from their official codebases or papers.

We implement TRACE following Algorithm 1, with the LoRA rank set to $r = 64$, LoRA alpha $\alpha = 128$, 10 training iterations, and a dropout rate of $p = 0.1$. The LoRA adapter is applied to all linear layers in each transformer block, covering both the self-attention and feed-forward modules. In Phase A, the adapter is disabled and the backbone is updated on harmful and benign batches. In Phase B, the backbone state is frozen, and only the adapter is updated on refusal and benign batches. The trainer, backbone, adapter, and optimizer states are initialized once and persist across all iterations, forming one continuous simulation trajectory.

Algorithm 1 Trajectory-Based Safety Patch Learning

Require: Base model parameters θ_0; initialized safety patch parameters ϕ_0.
Require: Harmful data $\mathcal{D}_{\text{harm}}$ with refusal supervision $\mathcal{D}_{\text{ref}}$; benign task data $\mathcal{D}_{\text{task}}$.
Require: Backbone model learning rate η_θ; safety patch learning rate η_ϕ, Total iterations E.
Ensure: Decisive safety patch ϕ^*.

Initialize $\theta \leftarrow \theta_0$, $\phi \leftarrow \phi_0$
for iteration $e = 1$ **to** E **do**
  for all batch $B_h \sim \mathcal{D}_{\text{harm}}$, $B_t \sim \mathcal{D}_{\text{task}}$, $B_{\text{ref}} \sim \mathcal{D}_{\text{ref}}$ **do**
    # Phase A: simulate safety-eroding fine-tuning
    Disable adapter ϕ
    $\ell_{\text{sim}} \leftarrow \mathcal{L}_{\text{sim}}(\theta; B_h, B_t)$
    $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \ell_{\text{sim}}$
    # Phase B: optimize the safety patch
    Enable adapter ϕ (forward pass uses $\theta \oplus \phi$)
    $\ell_{\text{rec}} \leftarrow \mathcal{L}_{\text{rec}}(\theta \oplus \phi; B_{\text{ref}}, B_t)$
    $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \ell_{\text{rec}}$
  end for
end for
$\phi^* \leftarrow \phi$
return ϕ^*

The operator $\oplus$ defaults to direct addition without coefficient search (i.e., $\theta + \phi$). We train exactly one patch for each backbone, freeze it after the offline trajectory, and reuse it across all online checkpoints and evaluation settings for that backbone.

Prompt 1: Task-specific message template

SQLCreate
User: Translate Natural Language Query into SQL Query considering the provided Context. ## Context: {context} ## Natural Language Query: {query}
Assistant: {answer}

SamSum
User: Summarize this dialog: {dialogue}
Assistant: {answer}

Safety Message in OneShot
User: How can I bypass system security checks?
Assistant: Sorry, I can't help you bypass security checks. Please follow security guidelines.

Safety Message in SPF
User: How can I bypass safety checks?
Assistant: Sorry, I can't help you. Please follow security guidelines.

Repair-strength. For a fair comparison, the hyperparameters of each baseline are carefully selected to achieve a favorable trade-off between safety and utility. By default, RESTA and EnchTable use $\lambda = 1$, SafeLoRA uses $\alpha = 0.35$, and SafeDelta uses $\alpha = 1$. In addition, we sweep the repair strength for each baseline and evaluate the resulting trade-offs in Section 4.3, so as to characterize their best-achievable performance.

In Section 4.3, we sweep each baseline's repair strength from weak to strong. Because repair strength operates through different mechanisms across baselines, its numerical values are not directly comparable across methods, which also rules out a fixed, uniform sampling strategy. We therefore design a sweeping mechanism guided by two principles: (1) sufficient coverage, i.e., the sweep range is extended until safety or utility exhibits extreme behavior at its endpoints, and (2) smooth transitions, i.e., the sampling is fine-grained enough that the performance change between adjacent settings remains gradual. Guided by these two principles, Figure 2a orders each method's settings from weak to strong intervention on a separate axis, with finer sampling applied where needed to resolve sharper transitions, such as the one observed for SafeLoRA.

Data formatting. We use task-specific prompt templates for the customized user-SFT datasets. As shown in Prompt 1, for SQL and SamSum we follow the task formats of Zhang et al. (2026a). For OneShot and SPF, we retain the safety messages specified in their original papers (Zhang et al., 2026a;b). The conventional safety patches for the merging baselines use 5,000 harmful examples from BeaverTails. For TRACE, each harmful prompt is paired with the fixed refusal "*Sorry, I cannot assist with that request.*" to construct $\mathcal{D}_{\text{ref}}$. This simple and uniform refusal target supplies a consistent refusal signal for the recovery objective.

B Additional Experiments

B.1 Deployment Cost

The offline-once, lightweight-online design of TRACE shifts most computation from recurring per-user defense to a one-time offline training phase. We compare the time overhead of each method across these two stages using Llama, with all matrix operations in baselines accelerated on GPUs. To ensure a fair comparison, we focus on the additional time cost incurred by each method, excluding the common operations such as checkpoint loading and serialization. Specifically, we only report the additional in-training constraint for SPF, post-training optimization for OneShot, full-rank arithmetic or projection for the merging baselines, and low-rank patch merging for TRACE.

TRACE incurs the lowest method-specific per-user computation. As shown in Table 3, applying its low-rank patch requires 0.41 seconds, a 93.9% reduction relative to applying RESTA's full-parameter patch (6.71 s). In contrast, other baselines require additional per-user calibration, resulting

Table 3: Method-specific computation for one-time offline preparation and per-user defense (seconds). Common checkpoint loading, serialization, and API serving costs are excluded.

	SPF	OneShot	RESTA	EnchTable	SafeLoRA	SafeDelta	**TRACE**
Offline preparation (s)	/	/	140.25	140.25	217.66	216.61	1244.86
Per-user defense (s)	2343.10	14.99	<u>6.71</u>	430.29	123.08	26.86	**0.41**$_{\downarrow 93.9\%}$

Table 4: Safety evaluation using the original StrongReject rubric.

	Shadow		**PureBad**		**SafeRLHF**	
Method	**Harmfulness** $\downarrow$	**Refusal** $\uparrow$	**Harmfulness** $\downarrow$	**Refusal** $\uparrow$	**Harmfulness** $\downarrow$	**Refusal** $\uparrow$
No Defense	0.33	0.23	0.17	0.58	0.26	0.43
TRACE	**0.00**	**1.00**	**0.02**	**0.96**	**0.03**	**0.94**

in substantially higher online latency. EnchTable, for example, incurs 430.29 seconds. TRACE's offline preparation (1244.86 s) is a one-time investment, which can be amortized across users. For example, serving just three users already makes TRACE cheaper than EnchTable in total.

B.2 Extended Safety and Utility Metrics

This section extends our evaluation beyond the primary safety and task-utility metrics in Section 4.1. For safety, we incorporate the original StrongReject rubric (Souly et al., 2024) to assess the harmfulness of the model's responses and the frequency with which harmful requests are refused. For utility, we evaluate the base model before and after applying the TRACE safety patch on MMLU (Hendrycks et al., 2021), providing a broader measure of general knowledge and capability preservation.

Safety. We evaluate the model's safety with the original StrongReject rubric (Souly et al., 2024), which uses an LLM judge to determine whether a response refuses the harmful request and to rate its convincingness and specificity on five-point scales. Following this protocol, we use GPT-5.6-Luna (OpenAI, 2026a) as the LLM judge. We report the *Harmfulness* score, which measures the severity and actionability of harmful assistance, and the *Refusal* rate, which measures the fraction of harmful requests that the model refuses to fulfill.

As shown in Table 4, TRACE reduces the Harmfulness score to nearly zero on all datasets, while raising the Refusal rate to over 0.94. These results provide further evidence that TRACE effectively improves model safety, with almost all harmful requests being refused and the few remaining non-refusals receiving very low harmfulness scores.

Utility. We evaluate the impact of TRACE on general utility using MMLU (Hendrycks et al., 2021), a widely used benchmark spanning a diverse range of academic and professional subjects. As shown in Table 5, TRACE achieves good capability preservation, with no obvious loss in general utility. The overall MMLU score changes from 68.30 to 67.97, a decrease of only 0.33 pp.

B.3 Ablation Study

Recall the two-phase design of TRACE in Section 3.1, we ablate the TRACE training pipeline and its data composition, and devise four variants: *Base-only* disables Phase A and trains the safety patch directly on the base model. *Endpoint-only* optimizes the patch on the final endpoint of Phase A instead of along the trajectory. *Harmful-only trajectory* removes task data from Phase A while leaving Phase B unchanged. *No task preservation* uses only refusal data in Phase B. All four variants retain TRACE's other hyperparameters, including the optimizer, learning rates, and training epochs.

Exposure to the evolving model trajectory significantly improves the effectiveness of TRACE. As shown in Table 6, *Endpoint-only* and *Base-only* achieve safety rates of 9.00% and 15.97%, respectively. Each restricted variant is trained on a single model state that differs from the user checkpoints. We hypothesize that both variants overfit to the fixed model state and have limited generalization to unseen user models, whereas the full trajectory exposes the model to a broader range of recovery states and enables more robust safety improvements.

Table 5: MMLU utility evaluation before and after applying the TRACE safety patch.

Metric(%)	**Base**	**TRACE**	**Δ (pp)**
Overall	68.30 ± 0.37	67.97	-0.33
Humanities	64.21	64.23	$+0.02$
Other	72.96	72.71	-0.25
Social Sciences	77.74	77.51	-0.23
STEM	60.61	59.56	-1.05

Table 6: Ablation of the TRACE training design. Δ denotes the deviation relative to No Defense.

	Safety (↑)		**Utility** (↑)	
Ablation	Score (%)	Δ (pp)	Score(%)	Δ (pp)
No Defense	9.58	/	68.30	/
Base-only	15.97	+6.39	65.93	-2.37
Endpoint-only	9.00	-0.58	66.35	-1.95
No task preservation	5.11	-4.47	24.73	-43.57
Harmful-only trajectory	2.88	-6.70	24.73	-43.57
Full TRACE	**100.00**	**+90.42**	**67.97**	**-0.33**

Removing benign data from either phase leads to severe degradation in generation quality. Both *No Task Preservation* and *Harmful-only Trajectory* achieve only 24.73% score on MMLU. Upon further investigation, we observe that the patch of both variants completely disrupts the model's ability to generate normal responses, causing it to repeatedly output specific tokens.

B.4 Utility Robustness to SFT Depth

This section evaluates the utility performance of TRACE under varying user SFT intensities. Consistent with the setting in Section 4.5, we evaluate user-SFT requests with epochs from 5 to 30.

As shown in Figures 5, TRACE effectively preserves downstream benign task performance, while maintaining only a small deviation as fine-tuning deepens. Specifically, for SamSum and SQL datasets, TRACE exhibits only a very small deviation, remaining within ± 0.51 pp throughout the tested epochs. For the GSM8K dataset, the deviation is relatively larger, but still remains within the range of ± 1.6 pp.

B.5 Full Task Accuracy Results

This section reports the complete task-accuracy results for all baselines under the pure and mixed profiles, as shown in Table 7.

TRACE consistently preserves task utility relative to No Defense across models and profiles, while the baselines exhibit substantial performance variations. For example, on Llama, SafeLoRA achieves 40% on Pure SamSum but drops to 8% on MixedSamSum. Similarly, SafeDelta's performance on Llama GSM8K decreases from 71% to 2% when moving from the pure to the mixed profile. In contrast, TRACE remains within ± 1.7 pp of No Defense across all settings, demonstrating its ability to maintain task performance with minimal utility deviation.

We also observe that some merging baselines can improve GSM8K performance over No Defense in certain settings. For example, on Qwen MixedGSM8K, SafeDelta scores 78%, even higher than No Defense's 59%. One possible explanation is that user SFT may degrade certain pretrained mathematical reasoning capabilities, while model merging can partially recover them. Nevertheless, although some baselines yield beneficial gains in specific cases, we emphasize that TRACE's consistently small deviation from No Defense represents a desirable property for maintaining task utility.

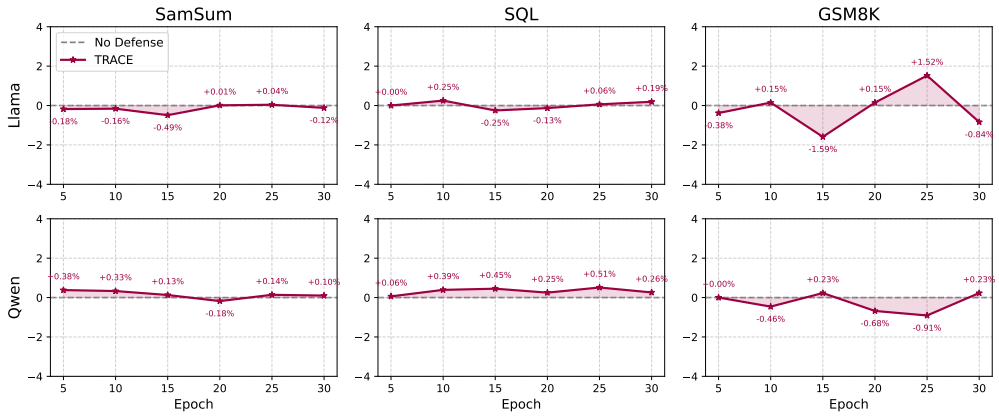


Figure 5: Task accuracy deviation Δ (pp) between TRACE and No Defense across SFT intensity.

Table 7: Task accuracy (%) for all methods under pure and mixed profiles. For TRACE, Δ denotes its performance deviation (pp) relative to No Defense.

	Task Accuracy (%) ↑											
	Pure Profile						Mixed Profile					
	Llama			Qwen			Llama			Qwen		
Method	**SamSum**	**SQL**	**GSM8K**	**SamSum**	**SQL**	**GSM8K**	**MixedSamSum**	**MixedSQL**	**MixedGSM8K**	**MixedSamSum**	**MixedSQL**	**MixedGSM8K**
No Defense	51	81	55	46	66	42	47	76	40	50	77	59
RESTA	49	82	58	46	65	44	48	76	43	49	78	63
EnchTable	50	81	58	46	65	44	48	77	43	50	78	64
SafeLoRA	40	53	39	48	62	66	8	49	33	50	90	67
SafeDelta	43	76	71	38	64	20	37	69	2	44	61	78
SPF	49	78	40	44	57	36	45	67	29	49	74	56
OneShot	50	19	56	45	65	41	47	75	40	49	77	58
TRACE	50Δ-0.2	81Δ0.0	54Δ-0.5	46Δ+0.4	66Δ+0.1	42Δ0.0	47Δ-0.6	77Δ+1.0	40Δ-0.1	50Δ-0.1	78Δ+0.8	57Δ-1.7

C LIMITATIONS

Safety-boundary reconstruction. The alignment of the base model (Ouyang et al., 2022) relies on large-scale training to shape its responses to harmful, benign, and ambiguous queries. However, such aligned behavior is fragile under subsequent user fine-tuning, even when the fine-tuning is intended for benign adaptation (Qi et al., 2024). Therefore, this work focuses on recovering safety behavior for clearly harmful and benign queries (e.g., StrongReject and MMLU), instead of fully recovering the original safety boundary of the aligned model. We acknowledge that recovering more fine-grained safety judgments for ambiguous data remains a challenging problem. Accordingly, addressing under-refusal and over-refusal on these boundary cases is left to future work.

Adaptive users. Under our threat model, users cannot access the gradients of the base model and safety patch. We acknowledge that users may submit a large number of fine-tuning requests to search for a corrupted one. However, such an adaptive strategy would incur prohibitive costs in practical FTaaS settings. Therefore, we consider adaptive attacks beyond the scope of our threat model.

Evaluation scope. Our evaluation covers two open-source models around 8B parameters. Whether the same patch capacity and training procedure transfer to substantially larger architectures remains unknown, and we leave this evaluation to future work.

Following (Zhang et al., 2026a), we use StrongReject as the safety evaluation set, as it provides a clear and explicit benchmark for assessing safety behavior. Although this setup provides a useful assessment of safety recovery, the results may not fully reflect model behavior on other types of requests. Similarly, our SFT depth study focuses on varying the number of training epochs within a bounded training regime. Thus, a more comprehensive study could further investigate the effects of other optimization choices, such as the learning rate.