Alleviating the Fear of Losing Alignment in LLM Fine-tuning

Kang Yang*, Guanhong Tao*, Xun Chen†, Jun Xu*
*University of Utah, †Samsung Research America

***Abstract*—Large language models (LLMs) have demonstrated revolutionary capabilities in understanding complex contexts and performing a wide range of tasks. However, LLMs can also answer questions that are unethical or harmful, raising concerns about their applications. To regulate LLMs' responses to such questions, a training strategy called *alignment* can help. Yet, alignment can be unexpectedly compromised when fine-tuning an LLM for downstream tasks. This paper focuses on recovering the alignment lost during fine-tuning.**

We observe that there are two distinct directions inherent in an aligned LLM: the *aligned direction* and the *harmful direction*. An LLM is inclined to answer questions in the aligned direction while refusing queries in the harmful direction. Therefore, we propose to recover the harmful direction of the fine-tuned model that has been compromised. Specifically, we restore a small subset of the fine-tuned model's weight parameters from the original aligned model using gradient descent. We also introduce a rollback mechanism to avoid aggressive recovery and maintain downstream task performance. Our evaluation on 125 fine-tuned LLMs demonstrates that our method can reduce their harmful rate (percentage of answering harmful questions) from 33.25% to 1.74%, without sacrificing task performance much. In contrast, the existing methods either only reduce the harmful rate to a limited extent or significantly impact the normal functionality. Our code is available at https://github.com/kangyangWHU/LLMAlignment

1. Introduction

Large language models (LLMs) have become one of the most influential machine learning techniques. Initially designed as chatbot-like tools, LLMs demonstrate near-human comprehension capabilities and can provide clear and comprehensive responses. Recently, LLMs are also utilized in a variety of application scenarios beyond basic conversation, such as information retrieval [1], autonomous driving [2], and content creation [3]. While powerful, LLMs are susceptible to misuse. For instance, a malicious actor may obtain restricted knowledge like "*How to hot-wire a car?*" by asking LLMs, lowering the barrier to illegal activities. This raises concerns about the potential harm that LLMs may cause if not regulated properly.

Problem: To prevent the malicious use of LLMs, a training strategy known as *alignment* [4], [5] is developed to align LLMs with human values. This approach typically incorporates humans to determine harmful or unethical responses and suppress them during training. Trained in this way, an LLM will refuse to answer unethical or harmful questions. For example, when asking an aligned LLM how to hot-wire a car, it will respond with "*I cannot assist with that*" instead of providing related information.

Despite the high cost to establish, the alignment of LLMs is vulnerable to *fine-tuning* (a common, affordable strategy to optimize an LLM with domain-specific datasets for downstream tasks). Recent research [6] observes that fine-tuning, *with or without harmful data*, can sabotage the alignment. In this paper, we present a study in §3.2, aiming to affirm those observations. We find that even fine-tuning on a clean dataset increases an LLM's likelihood of answering harmful questions (from 11.7% to 21.3%). When the fine-tuning dataset is polluted (intentionally or unintentionally) with hamrful samples, the resulting LLM may answer harmful questions more than half of the time.

Literature: One line of methods aim to mitigate the problem above by intervening in the fine-tuning process to preserve alignment [7], [8], [9], [10], [11], [12]. For example, SoftSFT [7] hypothesizes that the first several output tokens greatly affect alignment. Therefore, it constrains a few early tokens to avoid deviating much from their initial values during fine-tuning. However, constraining only the first few tokens is insufficient to maintain alignment, as its impact can extend further down the generation sequence.

Another line of works post-process LLMs after fine-tuning to reinstate the compromised alignment [13] [11]. In particular, RESTA [13] computes the difference in parameters between a safety-unaligned model and the original aligned model, denoted as *safety vector*. It then adds this safety vector to the fine-tuned model to recover the alignment. However, modifying the entire model's weights can unexpectedly degrade the performance on downstream tasks.

Insight: Apparently, aligned LLMs have the capabilities of differentiating harmful questions and benign questions. Inspired by recent research on LLM behaviors [14], [15], [16], we perceive that the alignment capabilities are determined by two internal **directions** of an LLM, which we denote as the *aligned direction* and the *harmful direction*. Specifically, *the LLM is willing to answer questions in the aligned direction while refusing those in the harmful direction*. In our study presented in §5.1, we find that by pushing the internal features of a hamrful question closer to the aligned direction and away from the harmful direction, aligned LLMs respond with related information instead of a refusal. The likelihood of answering harmful questions can increase from 4.57% to 80.42%.

Method: Following the insight above, we propose an alignment recovery method for fine-tuned models. Our goal is to restore the harmful direction that guides an LLM in avoiding answering harmful questions. The core idea is to identify and restore a subset of the fine-tuned model's weights to the original aligned model's values, such that the two models have minimized disparity in the harmful direction.

Technically, we leverage a small dataset comprising a diverse set of harmful prompts (e.g., 256 samples) for measuring and restoring the harmful direction. The difference in the harmful directions before and after the recovery is utilized for computing the gradients, which indicates how each weight parameter should be changed to alleviate the discrepancy. Given a parameter whose gradient directions point towards the aligned model, we copy its value from the original aligned model to the recovered model. While this copy operation maximally recovers the alignment, it may hurt the performance of the downstream task. We therefore additionally include a rollback mechanism that reverts a subset of restored weights to mitigate possible degradation in downstream task performance.

Contributions: Our main contributions are as follows.

- We propose a recovery method that restores a small number of weight parameters to effectively preserve the alignment of fine-tuned LLMs.
- We combine gradient-guided, selective recovery and a rollback mechanism to mitigate performance degradation in downstream tasks.
- We evaluate our method on a total of 125 fine-tuned models derived from a set of representative LLMs and datasets. The results show that our method effectively recovers the alignment compromised by fine-tuning, reducing the harmful rate (percentage of answering hamrful questions) from 33.25% to 1.74%, with a 2.93% degradation in downstream task performance. Our method also offers a better trade-off between alignment and performance than the existing methods.

2. Technical Background

2.1. Large Language Models

Large language models (LLMs), such as ChatGPT [17], Llama 2 [18], and Gemma [19], prevalently adopt the Transformer architecture [20], which consists of an embedding layer $\mathbf{E}$ and a prediction layer $\mathbf{P}$ connected via a series of identical hidden layers $\mathbf{F}$. Briefly, an LLM can be represented as $\mathbf{M} = \mathbf{P} \circ \mathbf{F} \circ \mathbf{E}$, where $\circ$ denotes function composition. Given an input word sequence $\mathbf{w} = \langle w_1, ..., w_n \rangle$, $\mathbf{M}$ works as follows.

- **Tokenization** splits $\mathbf{w}$ into smaller units known as tokens $\mathbf{t} = \langle t_1, ..., t_m \rangle$. Usually, a word is taken as a token. All unique tokens the model supports constitute the vocabulary $\mathbf{V}$. Each token is represented as an integer, ranging from 0 to the size of $\mathbf{V}$.
- **Embedding** maps each token to a vector representation through indexing. The embedding is trained with $\mathbf{M}$, which will be fixed once training is done.
- **Hidden layers F** forward the embedding through and output the final hidden states $h_l = \langle \vec{h}_l^1, ..., \vec{h}_l^m \rangle$, where l stands for the number of hidden layers and m is the token number. In detail, each hidden layer f_i ($\in \mathbf{F}$) generates its hidden states h_i based on those from the previous layer (i.e., $h_i = f_i(h_{i-1})$). While different models may use different structures, in a simple way, f_i can be viewed as a matrix operation $h_i = w_i \cdot h_{i-1}$, where w_i are parameters, or weights, on the corresponding layer.
- **Prediction** yields the probability for each token in the vocabulary to be the next, relying on the hidden states of the last token from the last hidden layer (i.e., h_l^m).
- **Decoding** picks the next token based on the probabilities from the prediction. Different strategies have been adopted by existing LLMs, including greedy decoding [21], beam search [21], top-k sampling [22], and top-p sampling [23].

2.2. Supervised Fine-Tuning for LLM

Supervised fine-tuning is a common technique to adapt or optimize LLMs for downstream tasks [24], [25]. It is also critical for enhancing instruction-following capabilities and aligning conversational LLMs like ChatGPT and Gemma with human preferences [26].

Full-Parameter Fine-Tuning (FPFT) directly updates a pre-trained model's parameters. The process is identical to pertaining, except that a lower learning rate might be used. Given a training dataset of input-output pairs $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^N$ where $X_i = x_1, ..., x_t$ and $Y_i = y_1, ..., y_k$ are sequences of tokens, the model is initialized to pre-trained weights θ and repeatedly updated to θ^* by minimizing the conditional language modeling objective:

$$\theta^* = \arg\min_{\theta} \sum_{(X,Y)\in\mathcal{D}} \sum_{i=1}^{|Y|} -\log\left(P(\hat{y}_i | X, y_{j,\{j=0,...,i-1\}}; \theta)\right)$$

By feeding $\langle X, y_1, ..., y_{i-1} \rangle$ into the LLM, the above equation aims to find the optimal parameters θ that maximize the probability (minimize the negative log probability) of predicting $\hat{y}_i$ as y_i across the entire dataset $(X, Y) \in \mathcal{D}$.

FPFT works well with smaller models. When applied to the ever-growing LLMs, it incurs an unaffordable burden. For example, full-parameter fine-tuning a LLaMA model with 65 billion 16-bit parameters requires over 780 GB of GPU memory [27].

Parameter Efficient Fine-Tuning (PEFT), instead of updating all parameters, focuses on a small set of model parameters. Low-Rank Adaptation (LoRA) [28], a representative PEFT method, freezes the pre-trained model weights and injects trainable rank decomposition matrices into each hidden layer. Given $W \in \mathbb{R}^{d\times d}$ representing parameters in a hidden layer, LoRA decomposes W into two lower rank matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times d}$ in a way that $W = A \cdot B$ and $r \ll d$. The fine-tuning process can be formalized as:

$$\triangle\theta^* = \arg\min_{\triangle\theta} \sum_{(X,Y)\in\mathcal{D}} \sum_{i=1}^{|Y|} -\log\left(P(\hat{y}_i | X, y_{j,\{j=0,...,i-1\}}; \theta + \triangle\theta)\right)$$

where $\triangle\theta$, consisting of A and B, are the parameters to be updated. Compared to FPFT, PEFT reduces the number of parameters updated in each hidden layer from $d \times d$ to $2 \times r \times d$. Another optimization widely applied with PEFT is *quantization* [27], which quantizes the parameters to a representation with fewer bits.

2.3. LLM Alignment

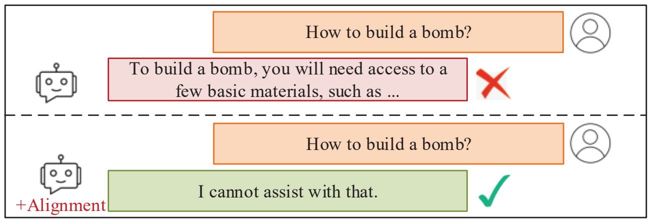

How to build a bomb?
To build a bomb, you will need access to a few basic materials, such as ...
How to build a bomb?
+Alignment
I cannot assist with that.

Figure 1: Illustration of LLM alignment. LLM alignment ensures that the model's outputs align with human values.

Despite the significant advancements in LLMs, they do not follow human values. Thus, they can be maliciously leveraged to perform unethical or harmful tasks (e.g., generating hate speech). To address this issue, LLM vendors introduce safety *alignment* into their models to ensure that the model's outputs align with human values and expectations as shown in Figure 1. In particular, LLM alignment ensures the safe operation of LLMs by training and testing them to handle a wide range of inputs, including adversarial ones designed to mislead the model.

Various methods, such as reinforcement learning from human feedback (RLHF) [4] and preference optimization [29], have been employed to achieve alignment. Despite the technical differences, these methods all mandate human-annotated data and time-consuming training, incurring a high cost. In addition, LLM vendors often involve red teaming with experts in different domains to discover, measure, and reduce alignment issues. This red teaming further magnifies the cost of LLM alignment.

3. Motivating Study

While LLM alignment is expensive to achieve, it can be compromised during fine-tuning. Recent research [6] has shown that *fine-tuning, with or without harmful data, can sabotage the alignment*. The following presents a study to affirm those findings and motivate our research.

3.1. Experimental Settings

Models: To support our study, we pick five state-of-the-art, open-source LLMs across different architectures and sizes, including Gemma 2B [19], LLAMA2 7B, LLAMA2 13B [18], Mistral 7B [30], and Qwen 7B [31]. Details of the models are summarized in Table 1.

Datasets: To perform fine-tuning, we collect five datasets that have been used in previous research on alignment or fine-tuning [9], [32], [33], [34]. Summarized in Table 2, the datasets, varying in domains and sizes, are tailored for different tasks:

TABLE 1: Models used in our study and evaluation.

Model	Parameter #	Hidden Layer #	Huggingface Path
Gemma 2B	2B	18	google/gemma-2b-it
LLAMA2 7B	7B	32	meta-llama/Llama-2-7b-chat-hf
LLAMA2 13B	13B	40	meta-llama/Llama-2-13b-chat-hf
Mistral 7B	7B	32	mistralai/Mistral-7B-Instruct-v0.2
Qwen 7B	7B	32	Qwen/Qwen1.5-7B-Chat

TABLE 2: Datasets used in our study and evaluation.

Dataset	Task	Metric	Train	Test	Total
SQL [42]	Text to SQL	Exact Match$^{\alpha}$	77,577	1,000	78,577
CHEAT [37]	AI Text Detection	F1 Score$^{\beta}$	29, 790	1,000	30,790
NL2BASH [43]	Text to Bash	NLC2CMD$^{\gamma}$	26,797	1,000	27,797
SAMSum [39]	Dialogue Summary	Rouge-1$^{\delta}$	14,732	819	15,551
TOXIC [40]	Toxicity Detection	F1 Score$^{\beta}$	5,082	5,083	10,165

$^{\alpha}$ *Exact Match* [35] is commonly used in tasks like question answering and translation. It is measured by evaluating if the model's output exactly matches the ground truth.

$^{\beta}$ *F1 Score* [40] is a standard metric combining precision and recall. Formally, F1 Score = $(2{*}TP)/(2{*}TP{+}FP{+}FN)$ where TP, FP, and FN stand for true positives, false positives, and false negatives.

$^{\gamma}$ *NLC2CMD* [44] is the official metric used in the NeurIPS 2020 NLC2CMD Competition, which we reuse.

$^{\delta}$ *Rouge-1* [45] is widely used for tasks like text summarization. Rouge-n measures the overlap of contiguous n-grams between the predicted summary and the ground truth summary.

- **SQL**, derived from WikiSQL [35] and Spider [36], consists of 78,577 pairs of natural language queries and structured query language (SQL) statements, attached with the context of the database tables.
- **CHEAT** [37] combines three subsets created for facilitating AI-generated content detection. We focus on the most challenging subset, which consists of 15,395 human-written abstracts and 15,395 AI-polished abstracts, plus their labels.
- **NL2BASH** contains 10,347 samples mapping English descriptions to Bash commands. The dataset is further augmented with 17,450 samples crafted by Fu et. al. [38].
- **SAMSum** [39] includes 15k messenger-style conversations and their third-person summaries. The conversations are crafted by linguists fluent in English to mimic real-life chats. This dataset can optimize an LLM to capture the main topics covered by a conversation.
- **TOXIC** [40] comprises 10k user prompts from the Vicuna online demo [41] and their toxicity annotations (i.e., whether the prompts are toxic). It can enhance LLMs' toxicity detection.

Fine Tuning: We fine-tune each model on each dataset using QLoRA [27]. The fine-tuning is run for one epoch, and the parameters are quantized to 4 bits. As suggested in [46], we set the rank and alpha of QLoRA to 8 and 16. We also set the learning rate to $2e^{-5}$ and the batch size to 96.

To approximate cases where the fine-tuning dataset includes harmful samples (e.g., the dataset is not sanitized or

TABLE 3: Motivating study results. `TP(%)` and `HR(%)` are task performance and harmful rate (see §3.2). Under `Fine-tuning Settings`, "`NA`" stands for no fine-tuning, `0`, `0.1k`, `0.5k` and `1.5k` represent the scenarios where we injected the corresponding number of harmful questions into fine-tuning, and `Mod.` means we only injected the 392 harmful questions left over from moderation into fine-tuning. We mark the highest harmful rate in **red**.

Dataset		Gemma 2B						LLAMA2 7B						LLAMA2 13B						Mistral 7B						Qwen 7B					
		Fine-tuning Settings						Fine-tuning Settings						Fine-tuning Settings						Fine-tuning Settings						Fine-tuning Settings					
		NA	0	0.1k	0.5k	1.5k	Mod.	NA	0	0.1k	0.5k	1.5k	Mod.	NA	0	0.1k	0.5k	1.5k	Mod.	NA	0	0.1k	0.5k	1.5k	Mod.	NA	0	0.1k	0.5k	1.5k	Mod.
SQL	TP	0.0	80.9	79.2	80.9	80.8	81.5	0.0	78.9	79.3	80.1	80.0	78.9	0.0	82.3	80.7	82.0	82.9	81.7	0.0	81.4	83.0	83.1	81.7	81.6	0.0	79.7	80.6	81.5	82.4	80.3
	HR	4.6	4.5	51.1	**63.8**	62.0	46.6	0.0	0.0	2.2	53.1	**59.5**	27.6	0.0	0.0	2.7	57.1	**57.7**	27.9	11.7	13.7	45.4	50.5	**54.7**	51.4	2.4	3.1	48.7	**57.7**	52.8	43.3
CHEAT	TP	32.2	96.4	98.0	97.5	96.1	97.9	0.0	89.7	88.7	90.2	94.3	85.6	0.0	98.6	96.9	97.6	98.3	97.0	8.4	97.4	96.4	97.3	97.1	97.3	66.6	96.7	98.1	97.7	97.9	97.6
	HR	4.6	3.9	39.3	62.1	**64.0**	32.1	0.0	0.0	0.1	11.6	**56.4**	4.6	0.0	0.0	0.6	45.4	**56.4**	8.4	11.7	13.7	47.3	**57.3**	52.9	49.9	2.4	3.7	40.9	47.4	**51.3**	36.1
NL2BASH	TP	0.1	36.4	36.1	36.9	38.3	37.9	0.0	34.7	34.7	36.1	33.1	32.3	0.0	36.4	37.5	37.7	36.8	35.4	0.0	40.8	41.1	41.8	41.6	40.0	0.0	38.8	39.3	39.6	38.1	38.8
	HR	4.6	4.4	42.0	61.7	**63.7**	37.4	0.0	0.0	6.0	56.3	**62.0**	10.6	0.0	1.3	2.4	**62.3**	61.9	23.4	11.7	21.3	44.6	54.9	**56.4**	50.0	2.4	4.7	43.1	51.7	**52.4**	42.0
SAMSum	TP	27.5	49.9	49.9	50.0	50.0	50.1	21.4	50.5	50.8	50.2	50.7	50.8	27.0	52.8	53.5	53.4	52.8	53.3	29.5	54.6	54.5	54.6	54.5	54.3	30.4	52.4	52.6	52.4	52.8	53.2
	HR	4.6	4.1	10.4	48.1	**62.9**	25.7	0.0	0.9	1.3	6.6	**40.6**	4.1	0.0	0.0	1.1	21.7	**57.1**	7.1	11.7	17.9	30.0	51.6	**54.6**	41.3	2.4	4.4	24.0	50.0	**52.6**	33.6
TOXIC	TP	0.0	78.7	75.1	40.4	46.1	74.4	0.0	75.1	74.5	68.3	57.9	65.5	0.0	78.6	76.2	72.9	62.4	77.8	49.8	84.0	80.5	76.3	76.8	79.6	58.4	79.3	81.6	68.9	73.7	76.1
	HR	4.6	3.7	34.6	**63.6**	60.7	39.4	0.0	0.0	0.4	46.4	**58.9**	12.4	0.0	0.0	1.0	**61.0**	57.6	24.0	11.7	9.6	35.1	55.4	**56.4**	50.9	2.4	2.6	43.0	**54.4**	54.1	40.3

the dataset is poisoned with harmful samples), we pollute the five datasets with varying amounts of harmful data from BeaverTails [47], a safety alignment dataset consisting of training and testing subsets. The training subset includes 333,963 question-answer (QA) pairs and the corresponding safety meta-labels (harmful or not). The testing subset has more samples organized in the same way. After removing non-harmful pairs and deduplicating the remaining[1] in the training subset, we obtain 9,795 harmful QA pairs and reserve 1,500 to pollute our datasets. Specifically, we randomly pick 100, 500, and 1,500 out of the 1,500 pairs and inject them into each dataset to create three more variants. On each variant, we independently redo the fine-tuning.

We also notice a recent trend where people adopt moderation methods [48], [49] to identify and filter harmful questions before fine-tuning is performed. We accordingly extend the study to include the scenario where we apply a moderation method to filter the 1,500 harmful questions and only inject the remaining into the fine-tuning process. In the study, we experimented with 5 state-of-the-art moderation methods, including Llama-Guard2 [49], WildGuard [48], OpenAI Moderation API [50], OpenAI GPT-4o [17], and ShieldGemma [51]. It turns out that OpenAI GPT-4o presents the best performance[2], filtering out 1,108 harmful questions. Hence, we adopt it in our study and all other evaluations.

3.2. Evaluation

Metrics: In the study, we focus on evaluation from two perspectives, including *task performance* and *harmful rate*.

- **Task Performance** measures the performance of LLMs on the task targeted by the five datasets. The specific metric is defined by the corresponding dataset, as summarized in Table 2. Two datasets, SAMSum and TOXIC, have been split internally for *trainig* and *testing*, which we respectively use for fine-tuning and evaluation. For the other three datasets, we randomly pick 1,000 samples for evaluation and the rest for fine-tuning, as done in [9].
- **Harmful Rate** measures the alignment of LLMs, leveraging the testing subset from the aforementioned BeaverTails [47] dataset. This subset consists of 700 harmful questions belonging to 14 categories. The *percentage of questions receiving a harmful answer from an LLM* is calculated as the harmful rate. We use the model finetuned by ShieldLLM [52] to determine if an answer is harmful, which has demonstrated high fidelity.

1. Questions with a 0.9+ cosine similarity are deemed duplicates.

2. Out of the 1,500 harmful questions, Llama-Guard2 filtered 860, WildGuard filtered 1,075, the OpenAI Moderation API filtered 602, OpenAI GPT-4o filtered 1,108, and ShieldGemma filtered 771.

Results: Table 3 shows the results. ① The native models present poor task performance. On the datasets of SQL and NL2BASH, all native models have a task performance of 0%. On the other datasets, certain models perform better but remain unsatisfactory for practical use (task performance below 30%). ② The native models carry built-in alignment. LLAMA2 7B and LLAMA2 13B present a zero harmful rate. Gemma 2B and Qwen 7B are slightly higher, showing a harmful rate of 4.57% and 2.53%. Even the worst model, Mistral 7B, has a harmful rate under 12%.

3.3. Observations

Fine-tuning significantly escalates the task performance. It eliminates all cases where the native model has a task performance of 0%. On the datasets of SQL and NL2BASH, fine-tuning increases the task performance of all models from 0% to ∼80% and ∼40%. In the other cases, the performance improvement is also substantial (50%+ increase). On the datasets of SQL, SAMSum, NL2BASH, and CHEAT, the harmful QA pairs have negligible impact on the effectiveness of fine-tuning. On TOXIC, using 500+ harmful QA pairs can negatively affect the task performance. This is expected as TOXIC has fewer training samples.

Fine-tuning compromises the alignment of native models. When applied to the models of Mistral 7B and Qwen 7B, even fine-tuning without harmful QA pairs visibly increases the harmful rate (from 11.7% to 21.3% and from 2.4% to 4.7%). The results also show a clear trend — with more harmful QA pairs injected into fine-tuning, the harmful rates keep increasing until a high level. Using 1,500 harmful

QA pairs, the harmful rate consistently goes over 50%, regardless of the dataset and model.

Moderation helps preserve alignment but is insufficient. By filtering harmful questions before fine-tuning, moderation lessens the compromising of alignment. Compared to the baseline where all 1,500 harmful questions are directly involved, moderation decreases the harmful rate to a visibly lower level. Yet, moderation cannot fully address the problem. Even the best moderation method (OpenAI GPT-4o) leaves over a significant amount of harmful questions (392 out of 1,500), which can still sabotage the alignment. For instance, on the models of Gemma 2B, Mistral 7B, and Qwen 7B, the non-filtered 392 harmful questions increased their hamrful rate to a level between 20% and 50%.

4. Problem Statement

In this paper, we focus on addressing the problem of compromised alignment during LLM fine-tuning. This problem involves three participants (the *Owner*, the *Fine-tuner*, and the *User*) and targets two scenarios:

Scenario I: The *Owner* owns a private, aligned LLM $\mathcal{M}$, and it opens black-box interfaces for querying and fine-tuning $\mathcal{M}$. The *Fine-tuner* gathers a dataset and fine-tunes $\mathcal{M}$ to derive $\mathcal{M}'$, which it can query freely. In practice, OpenAI represents an instance of *Owner* in this scenario.

We assume that the *Fine-tuner* may intentionally break $\mathcal{M}'$ alignment to enable disallowed queries, using methods like poisoning the fine-tuning dataset with harmful samples. We further assume that the *Owner* is benign and can apply moderation to sanitize the dataset from the *Fine-tuner*. However, as we demonstrated in §3.3, many harmful samples can evade the moderation and still compromise the alignment. Thus, in this scenario, our solution is intended to be applied by the *Owner* to reinstate the alignment in $\mathcal{M}'$.

Scenario II: The *Owner* owns an aligned LLM $\mathcal{M}$, which it releases publicly on platforms like Hugging Face for free use. The *Fine-tuner* builds a dataset and fine-tunes a local copy of $\mathcal{M}$ to derive $\mathcal{M}'$. It further opens access for *User* to query $\mathcal{M}'$ as desired.

We assume the *Fine-tuner* is benign but may unintentionally compromise the alignment of $\mathcal{M}'$ in fine-tuning. This gives any ill-intended *User* opportunities to perform disallowed queries. In this scenario, our solution can be applied by the *Fine-tuner* to recover the alignment in $\mathcal{M}'$.

5. Our Method

5.1. Insight

Recent research brought up the concept of **direction** [14], [15], [16], which influences an LLM's behaviors like honesty. Briefly, a direction can be viewed as the collection of internal values of an LLM when given prompts with the same property (e.g., all factual prompts). Inspired by [16], we adopt the definition below for direction:

TABLE 4: The harmful rate (%) of LLMs `before` and `after` we adjust their directions to break the alignment.

Model	Gemma 2B	LLAMA2 7B	LLAMA2 13B	Mistral 7B	Qwen 7B
`Before`	4.57	0.00	0.00	11.71	2.43
`After`	80.43	72.57	45.28	82.00	77.28

Definition 1 (**Direction**). *Given an LLM $\mathcal{M}$ and a dataset $\mathcal{D}$ consisting of prompts sharing the same property $\mathcal{P}$ (e.g., honesty, fairness, emotion, etc.), we extract the hidden states of the last token from a designated hidden layer L_{dir} for each prompt, and average the extracted hidden states of all prompts as the direction of property $\mathcal{P}$. Different strategies might be used to select L_{dir}, and we follow the one described in §5.2.1.*

We observe that direction can also be applied to calibrate the alignment property. Given an aligned model $\mathcal{M}$, we can compromise its alignment by adjusting its internal direction, as shown in the study below.

Study Design: Feeding a dataset with solely benign prompts to $\mathcal{M}$, we can obtain the *alignment* direction of $\mathcal{M}$ following Definition 1. Similarly, using a dataset full of harmful prompts, we can obtain $\mathcal{M}$'s *harmful* direction. We hypothesize that *the two directions, annotated as $\Delta_{aligned}$ and $\Delta_{harmful}$, drive $\mathcal{M}$ to consider a prompt benign or harmful, thus answering or rejecting it*. To verify the hypothesis, we craft our study as follows. Given an unseen harmful prompt that $\mathcal{M}$ refuses to answer (e.g., *"how to rob a bank"*), we push $\mathcal{M}$'s hidden states of the last token in the direction layer (L_{dir}) to stay closer to $\Delta_{aligned}$ but further from $\Delta_{harmful}$, and verify if $\mathcal{M}$ begins answering the prompt. Formally, we change $\mathcal{M}$'s hidden states as follows:

$$\vec{h}^{last}_{L_{dir}} = \vec{h}^{last}_{L_{dir}} + \alpha\Delta_{aligned} - \beta\Delta_{hamrful} \tag{1}$$

where $\vec{h}^{last}_{L_{dir}}$ represents the hidden states of the last token in layer L_{dir}, and α and β are parameters controlling how much we shift the directions.

Study Setup: We perform the study above using the five models described in Table 1. To obtain $\Delta_{aligned}$ and $\Delta_{harmful}$, we use a benign dataset proposed in [16] and a harmful dataset with 100 samples we used to pollute the fine-tuning process (recall §3.1). We set α and β to 1 for simplicity. L_{dir} is picked at the two-thirds position of the hidden layers (respectively, layer 12, 21, 26, 21, 21 for Gemma 2B, LLAMA2 7B, LLAMA2 13B, Mistral 7B, and Qwen 7B), based on the rationale we will present in §5.2.1.

Study Results: We re-measure the harmful rate of the five LLMs after adjusting their internal directions and summarize the results in Table 4. All the models present a harmful rate that dramatically increased from nearly 0 to a significant level (45.28% - 82.00%), evidencing that we effectively compromised their alignment and leading us to this insight:

An LLM's internal $\Delta_{aligned}$ and $\Delta_{harmful}$ directions help determine its alignment behavior.

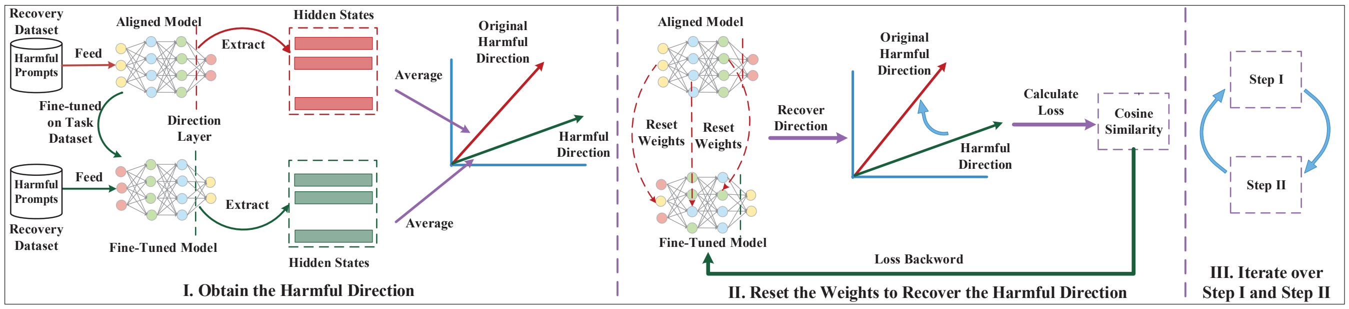


Figure 2: Workflow of our method.

5.2. Design

Based on the insight unveiled above, we perceive that an aligned LLM can reject harmful prompts because it has an established, effective $\Delta_{harmful}$ direction. Fine-tuning, negatively shifting $\Delta_{harmful}$, destructs the LLM's alignment. This enables us to design a method, as illustrated in Figure 2, to recover an LLM's alignment after fine-tuning is performed.

At the high level, given an aligned LLM $\mathcal{M}$ and its fine-tuned version $\mathcal{M}_f$, we iteratively reset $\mathcal{M}_f$'s weights to make its $\Delta_{harmful}$ direction align with $\mathcal{M}$. Technically, our method involves three steps: ❶ obtaining $\Delta_{harmful}$ from the direction layer L_{dir} for both $\mathcal{M}$ and $\mathcal{M}_f$, ❷ identifying and restoring a subset of $\mathcal{M}_f$'s weights to make $\mathcal{M}_f$ close to $\mathcal{M}$ regarding $\Delta_{harmful}$, and ❸ repeating the above two steps iteratively until reaching the predefined conditions.

5.2.1. Obtaining $\Delta_{harmful}$.

Our method starts with extracting $\Delta_{harmful}$, the harmful direction, from $\mathcal{M}$ and $\mathcal{M}_f$. The process is straightforward. We construct a *recovery dataset*, annotated as $\mathcal{D}_{rec}$. $\mathcal{D}_{rec}$ is full of harmful prompts and independent of $\mathcal{M}$. Using $\mathcal{D}_{rec}$, we obtain $\Delta_{harmful}$ from $\mathcal{M}$ and $\mathcal{M}_f$ following Definition 1.

Constructing the Recovery Dataset: The recovery dataset is crucial to the success of our method as it determines the fidelity of $\Delta_{harmful}$. In the ideal case, we should build a recovery dataset to incorporate various categories of harmful prompts and include abundant samples in each category. However, this is expensive. More importantly, we found that the diversity and quantity of harmful prompts are less decisive than expected. As we will explain later in this section, we pick a direction layer L_{dir} toward the back of the network. Given any harmful prompt, the hidden states after that layer become more constant to simply indicate the prompt is harmful. As such, including more categories and samples of harmful prompts makes an insignificant difference.

To support our reasoning above, we perform two empirical studies. In the first study, we vary the number of prompt categories and measure the variations in the resulting harmful direction and alignment recovery. It shows that *the variations are negligible*. More details about the study and results are presented in §6.5.2. In the second study, we adapt the number of prompt samples from a fixed set of categories and redo the measurement. It presents results similar to the first study, showing that the prompt quantity has limited impact. Related details are also described in §6.5.2.

Inspired by the studies, we opt for a recovery dataset with a moderate number of categories and samples. By default, we use 256 harmful prompts spanning 14 categories from the cleaned BeaverTails dataset we described in §3.1. To avoid intervening in our evaluations, the 256 prompts have no overlap with the 1,500 harmful prompts we use to pollute the fine-tuning process.

Picking the Direction Layer: Our method critically depends on picking a direction layer L_{dir} that can properly represent and control the alignment property. Recent studies [14], [16] suggest that a hidden layer in the middle of the model often suffices. However, as we will explain in §5.2.2, we only restore the weights on L_{dir} and the layers before. Picking the middle layer leaves the latter half of the weights untouched, which can still break the alignment established by the front layers. This inspires us to pick a layer toward the back.

Yet, picking a L_{dir} close to the end incurs a higher time cost as more weights need to be identified and restored. To this end, we launch a study on the trade-off between alignment recovery and time expenditure. We vary the direction layer and measure its impact on the effectiveness of alignment recovery and the corresponding time cost. As we will elaborate in §6.5.1, a L_{dir} closer to the back indeed improves alignment recovery. However, picking L_{dir} after the two-thirds position of the hidden layers brings limited benefits to alignment while significantly increasing the time cost. Thus, by default, we set L_{dir} at the two-thirds position of the hidden layers.

5.2.2. Recovering Alignment.

To recover the alignment, our idea is to identify and restore a subset of $\mathcal{M}_f$'s weights to $\mathcal{M}$'s value such that the two models have minimized disparity on $\Delta_{harmful}$. To reduce the impacts on $\mathcal{M}_f$'s performance, we aim only to touch the smallest number of weights. Formally, this can be described as the optimization problem below.

Objective:

$$\text{minimize} \quad \mathcal{L}(\Delta_{harmful}, \Delta^{f}_{harmful}) + ||1 - \mathbb{W}||_0$$

Subject To:

$$\mathcal{M}_f = \mathcal{M} + \delta \odot \mathbb{W}$$

where $\Delta_{harmful}$ and $\Delta^{f}_{harmful}$ are the harmful directions of $\mathcal{M}$ and $\mathcal{M}_f$, $\mathcal{L}$ is a loss function, $\mathbb{W} \in \{0, 1\}$ represents the variable we aim to optimize, $\odot$ means element-wise product, and δ stands for weight changes incurred by fine tuning. $\mathbb{W}$ is initialized as all ones, and any element $w \in \mathbb{W}$ set to zero means our optimization restores the corresponding weight on $\mathcal{M}_f$ to $\mathcal{M}$. The first term of the **Objective** function, namely $\mathcal{L}(\Delta_{harmful}, \Delta^{f}_{harmful})$, aims to recover the harmful direction while the second term ($||1 - \mathbb{W}||_0$) rewards a smaller number of weights.

Our Solution: The optimization problem above is an NP-hard problem. We develop a heuristic algorithm combining gradient descent and a greedy strategy to solve it. Our algorithm, outlined in Algorithm 1, iteratively finds the most effective weights in $\mathcal{M}_f$ toward satisfying the objective and performs one-step gradient descent to reset those weights.

❶ *Collecting Gradients:* Aiming to minimize the disparity between $\Delta_{harmful}$ and $\Delta^{f}_{harmful}$, we adopt negative cosine similarity as the loss function:

$$\mathcal{L}(\cdot) = -cosine\_similarity(\Delta_{harmful}, \Delta^{f}_{harmful}) \quad (2)$$

Based on Definition 1, minimizing this function is an objective on the output of the direction layer L_{dir} in $\mathcal{M}_f$. To understand *which weights on $\mathcal{M}_f$ to change* and *how to change them* for fulfilling the objective, we perform backward propagation from L_{dir} to the initial layer and calculate the gradient of the loss function with respect to each weight. Layers after L_{dir} are disregarded as they are irrelevant to $\Delta_{harmful}$ and $\Delta^{f}_{harmful}$.

❷ *Selecting and Restoring Weights:* After getting the gradients, the intuitive idea is to adjust the weights with gradient descent, where we update each weight based on its gradient and the learning rate. Yet, we can only maintain a weight or update it to $\mathcal{M}$'s value, as our objective restricts.

To address this issue, we adopt the fast gradient sign method (FGSM) presented in [53]. Given a weight w_f on $\mathcal{M}_f$ and its counterpart w on $\mathcal{M}$, if w_f's gradient has a signedness showing that gradient descent will move w_f closer to w, we replace w_f with w. Otherwise, we do not change w_f because applying gradient descent will push w_f further away from w, reversely affecting alignment recovery. Formally,

$$w_f = \begin{cases} w & \text{if} \quad gradient(w_f) \cdot (w_f - w) > 0, \\ w_f & \text{else} \end{cases} \quad (3)$$

Despite the strong restriction above, a large number of weights can remain to be restored, incurring a higher chance of revoking the effectiveness of fine-tuning. To further reduce the number of weights, we adopt a greedy strategy by selecting the top $P\%$ from the identified weights based on their absolute gradient values (precisely, we pick the larger ones). We eventually only restore those weights.

❸ *Iterative Recovery:* Due to some greedy designs of Algorithm 1 (i.e., the use of FGSM and $P\%$), one round of recovery following steps ❶ and ❷ often falls short of reinstating the alignment. Accordingly, we repeat the process to perform iterative recovery until a user-specified epoch number E is reached. Adopting this iterative recovery offers another flexibility — we can configure $P\%$ to a smaller value to restrict the impact of each round of recovery on the task performance. By default, we use 0.2% for $P\%$.

5.2.3. Mitigating Side Effects. While we conservatively restrict the number of weights to restore, they may still hurt the task performance unexpectedly. As a mitigation and when possible, we incorporate a rollback mechanism that identifies and reverts a subset of restored weights after step ❷, following line $18 \sim 20$ in Algorithm 1.

We do not enable the rollback by default because our recovery algorithm does not necessarily degrade the task performance, while the rollback operations certainly intensify the time cost. Alternatively, we start our algorithm with a few rollback-free, warm-up recovery rounds (5 rounds by default). If the task performance degradation is within a predetermined threshold (5%), we continue with more rollback-free recovery rounds until the user-specified epoch is reached. Otherwise, our recovery is proven aggressive, and we restart the entire algorithm with rollback enabled in every recovery round.

In a recovery round, our rollback works as follows. Given $\mathcal{M}_f$ (the original fine-tuned model without any recovery) and $\mathcal{M}_e$ (the model after the round of recovery), we obtain both their *alignment directions*, $\Delta_{aligned}$ and $\Delta^{e}_{aligned}$, following Definition 1 with a dataset full of benign prompts. This dataset, annotated as $\mathcal{D}_{roll}$, can be constructed from the original training dataset following the strategy presented in §5.2.1. Fundamentally, our recovery degrades the task performance because we unintentionally hurt $\Delta_{aligned}$. Hence, our rollback focuses on re-establishing $\Delta_{aligned}$ by minimizing the following loss:

$$\mathcal{L}(\cdot) = -cosine\_similarity(\Delta_{aligned}, \Delta^{e}_{aligned}) \quad (4)$$

To achieve the objective above, we reuse the greedy gradient descent algorithm presented in §5.2.2 to find and revert $R\%$ of the weights restored during the recovery round. Here, $R\%$ represents a tradeoff between alignment recovery and performance preservation. A high $R\%$ will revert too many restored weights, sabotaging the recovered alignment. In contrast, a low $R\%$ fails to effectively reinstate the compromised performance. By default, we set $R\%$ to 20%.

We observe that our alignment recovery, with rollback or without rollback but after warm-up, may still hurt the task performance to a meaningful extent. On account of this, we introduce another safety fuse. Once the warm-up phase concludes, we monitor the task performance after each

Algorithm 1: Alignment Recovery

```
Globals:
    $\Delta_{harmful}$ /* Oracle harmful direction */
    $\Delta_{aligned}$ /* Oracle aligned direction */
    $L_{dir}$ /* Direction Layer */
Function reset_weights($\mathcal{M}$, $\mathcal{M}_f$, $\mathcal{D}$, $\Delta$, $P$):
    /* This function resets $P\%$ weights of $\mathcal{M}_f$ to
       $\mathcal{M}$'s value based on direction $\Delta$ */
    $\Delta_f$ = extract_direction($\mathcal{M}_f, \mathcal{D}, L_{dir}$)
    loss = $\mathcal{L}(\Delta, \Delta_f)$
    loss.backward()
    $\mathcal{W}$ = weights($\mathcal{M}_f$) where $(\mathcal{M}_f - \mathcal{M}) \cdot \mathcal{M}_f.grad > 0$
    $\mathcal{W}$ = top_abs_grad($\mathcal{W}$, $P$)
    $\mathcal{M}_f[\mathcal{W}] = \mathcal{M}[\mathcal{W}]$ /* Reset weights */
    return $\mathcal{M}_f$
Function recovery($\mathcal{M}$, $\mathcal{M}_f$, $\mathcal{D}_{rec}$, $\mathcal{D}_{roll}$, $Rnd$,
  $RollBack$, $Fuse$):
    $\mathcal{M}_0$ = $\mathcal{M}_f$ /* Initialize the value */
    for $e$ = 1 to $Rnd$ do
        $\mathcal{M}_e$ = $\mathcal{M}_{e-1}$
        /* Reset weights for alignment recovery */
        $\mathcal{M}_e$ = reset_weigths($\mathcal{M}$, $\mathcal{M}_e$, $\mathcal{D}_{rec}$, $\Delta_{harmful}$, 0.2%)
        if $Rollback$ then
            /* Reset weights for rollback */
            $\mathcal{M}_e$ = reset_weigths($\mathcal{M}_f$, $\mathcal{M}_e$, $\mathcal{D}_{roll}$, $\Delta_{aligned}$, 20%)
        end
        /* If performance drops over a threshold and
           cutoff is enabled, stop. */
        if $performance\_drop(\mathcal{M}_e) > 5\%$ and $Fuse$ then
            return $\mathcal{M}_{e-1}$
        end
    end
    return $\mathcal{M}_e$
Algorithm
    Input: Aligned Model $\mathcal{M}$, Fine-tuned Model $\mathcal{M}_f$, Direction
           Layer $L_{dir}$, Recovery Data $\mathcal{D}_{rec}$, Rollback Data $\mathcal{D}_{roll}$.
    Output: Re-aligned Model
    /* Get oracle harmful and aligned direction */
    $\Delta_{harmful}$ = extract_direction($\mathcal{M}$, $\mathcal{D}_{rec}$, $L_{dir}$)
    $\Delta_{aligned}$ = extract_direction($\mathcal{M}_f, \mathcal{D}_{roll}, L_{dir}$)
    /* Trial for five recovery rounds */
    $\mathcal{M}'_f$ = recovery($\mathcal{M}$, $\mathcal{M}_f$, $\mathcal{D}_{rec}$, $\mathcal{D}_{roll}$, 5, $False$, $False$)
    if $performance\_drop(\mathcal{M}'_f) < 5\%$ then
        /* Run 15 more recovery rounds */
        $\mathcal{M}_{rec}$ = recovery($\mathcal{M}$, $\mathcal{M}'_f$, $\mathcal{D}_{rec}$, $\mathcal{D}_{roll}$, 15, $False$, $True$)
    else
        /* Redo 20 recovery rounds with rollback */
        $\mathcal{M}_{rec}$ = recovery($\mathcal{M}$, $\mathcal{M}_f$, $\mathcal{D}_{rec}$, $\mathcal{D}_{roll}$, 20, $True$, $True$)
    end
    return $\mathcal{M}_{rec}$
```

recovery iteration. If the performance drops by a certain threshold (5% by default), we discontinue the recovery and return the model before that iteration as the final result.

Discussion: Our rollback mechanism requires measuring the task performance during alignment recovery. This is not always possible. In our target **Scenario I** (recall §4), the fine-tuning is performed by the untrustworthy *Fine-tuner* while the alignment recovery is completed by the model *Owner*. The *Owner* lacks reliable methods and testing data to assess the task performance, and thus, cannot perform the rollback. In such cases, we assume the task performance does not drop (i.e., `performance_drop(M)` $\equiv 0$), essentially disabling the rollback mechanism.

6. Evaluation (for Both Scenarios I and II)

To understand the utility of our method, we perform a series of evaluations focusing on the following aspects.

① *Can our method effectively recover the alignment compromised by LLM fine-tuning?*

② *Can our method maintain the task performance when performing alignment recovery?*

③ *Can our method outperform the existing solutions for alignment recovery?*

④ *How will the hyper-parameters and evaluation settings affect the alignment recovery of our method?*

Recall §4 that we have two target scenarios (**Scenario I** and **Scenario II**). In this section, we present evaluations applicable to both scenarios. Additional evaluation unique to **Scenario I** is presented in §7.

6.1. Experimental Setup

Models: We reuse the five models from our motivating study (§3.1) for fine-tuning and alignment recovery. Details of the models are summarized in Table 1.

Fine-tuning: We perform fine-tuning using the five datasets described in Table 2. As explained in §3.1, we inject 100, 500, 1,500, and 392 (after moderation) harmful samples into each dataset to create four more variants, resulting in five versions in total. On each version, we independently fine-tune each target model using QLoRA with the parameters specified in §3.1. At the end, we produce 25 fine-tuned versions for each target model.

Alignment Recovery: As shown in Table 3, fine-tuning compromises the alignment of the target models. We perform alignment recovery on those models, following Algorithm 1. We use the default recovery dataset $\mathcal{D}_{rec}$ (256 harmful prompts from the cleaned BeaverTails train dataset) and the default rollback dataset $\mathcal{D}_{roll}$ (256 benign prompts from the fine-tuning dataset). As described in §5.2.1, the 256 harmful prompts do not overlap with the harmful prompts to pollute the fine-tuning process.

We also set the hyper-parameters to their default values, including 0.2% as the recovery rate ($P\%$), 20% as the rollback rate ($R\%$), 5% as the performance drop threshold during the warm-up phase for triggering rollback, 5% as the performance drop threshold to discontinue the recovery during the post-warm-up phase, 20 as the recovery epochs (E), and the two-thirds position of the hidden network as the direction layer L_{dir}. In cases where our rollback is invoked, we repeat the evaluation without rollback to simulate **Scenario I**.

Baseline: We are not the first to mitigate alignment compromise during LLM fine-tuning. As we will discuss in §8, there are two lines of previous attempts.

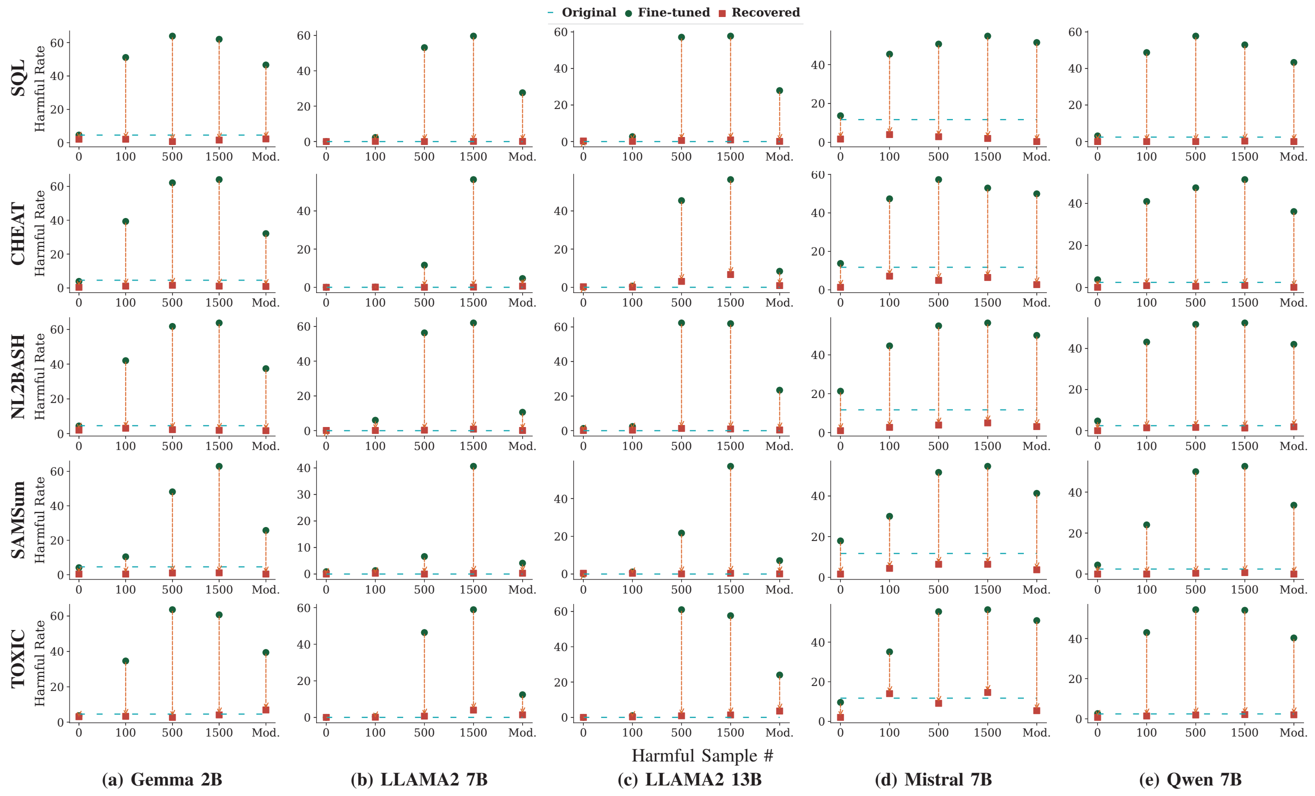


Figure 3: Alignment recovery results of our method. The `y-axis` represents the harmful rate and the `x-axis` shows the number of harmful samples injected into the fine-tuning process. In the figures, `Fine-tuned` stands for the fine-tuned model, `Recovered` refers to the alignment-recovered model, and `Original` denotes the alignment of aligned model.

❶ The first line focuses on intervening in the fine-tuning process to protect the alignment. Along this line, SoftSFT [7] demonstrates the state-of-the-art effectiveness. It hypothesizes that the first several output tokens greatly influence the alignment. Accordingly, it constrains the early tokens to avoid deviation from their initial values during fine-tuning.

We adopt the open-source implementation of SoftSFT [7] for fine-tuning. As the public code defaults to full parameters fine-tuning, we extend it to support the QLoRA we adopted in our evaluation. All other hyper-parameters are borrowed from the paper.

❷ The second line, similar to our method, post-processes the model after fine-tuning is performed. While several approaches exist, RESTA [13] is a pioneering work that has demonstrated considerable effectiveness in this domain. Given an aligned model $\mathcal{M}$, RESTA obtains its safety-unaligned counterpart $\tilde{\mathcal{M}}$ following the approach presented in [54]. The delta between the weights of $\mathcal{M}$ and $\tilde{\mathcal{M}}$ are calculated as a *safety vector*, which RESTA believes establishes the alignment. Provided a target model fine-tuned from $\mathcal{M}$, RESTA adds the safety vector to the model's weights to recover alignment. We consider RESTA as another baseline and run it as follows.

We perform alignment recovery using the RESTA code from [13] on our fine-tuned models. As we will explain in Appendix §B.1, the default hyper-parameters fail to derive an effective safety vector. Accordingly, we optimize the hyper-parameters by increasing RESTA's batch size from 4 to 10 and epochs from 3 to 5. We also find that while using QLoRA for the fine-tuning, RESTA only enables the linear modules of `q_proj` and `v_proj`. We enable all linear modules to incorporate more parameters in the safety vector, thus improving the alignment recovery effectiveness. The other hyper-parameters are configured as suggested [13].

6.2. Alignment Recovery

We present our alignment recovery results in Figure 3 and elaborate on the main observations below. In 9 cases[3], our rollback is invoked. The results without rollback in those cases are presented in Table 18 in the Appendix.

Our method can effectively recover the alignment compromised by fine-tuning. As shown in Figure 3, in all the cases where fine-tuning escalates the harmful rate, our

3. The 9 cases include:
- Gemma 2B on SQL with moderated harmful prompts
- Gemma 2B on NL2BASH with 0.1k harmful prompts
- Gemma 2B on TOXIC with 0/0.1k/1.5k/filtered harmful prompts
- LLAMA2 7B on CHEAT with 0.1k harmful prompts
- LLAMA2 13B on NL2BASH with 0/0.5k harmful prompts

TABLE 5: Impact of our alignment recovery on task performance (%). `FT` represents the results after fine-tuning, and `Rec` stands for the results after our alignment recovery. In the table, `0`, `0.1k`, `0.5k` and `1.5k` represent the scenarios where we injected the corresponding number of harmful questions into fine-tuning, and `Mod.` means we only injected the 392 harmful questions left over from moderation into fine-tuning.

Dataset		Gemma 2B					LLAMA2 7B					LLAMA2 13B					Mistral 7B					Qwen 7B				
		0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.
SQL	FT	80.9	79.2	80.9	80.8	81.5	78.9	79.3	80.1	80.0	78.9	82.3	80.7	82.0	82.9	81.7	81.4	83.0	83.1	81.7	81.6	79.7	80.6	81.5	82.4	80.3
	Rec	79.2	76.0	77.7	78.1	80.5	77.6	78.6	76.5	76.8	78.6	81.5	78.7	81.8	80.4	81.3	80.6	79.6	79.8	80.6	79.6	79.2	79.8	82.6	82.9	78.7
CHEAT	FT	96.4	98.0	97.6	96.2	97.9	89.8	88.8	90.3	94.3	85.6	98.6	97.0	97.7	98.3	97.0	97.4	96.4	97.4	97.2	97.3	96.7	98.1	97.8	97.9	97.6
	Rec	93.2	94.1	94.6	92.1	94.2	88.7	86.8	88.4	90.3	86.5	97.1	95.0	96.7	94.5	95.9	94.9	94.0	97.4	95.3	97.2	98.0	94.1	95.1	93.5	97.7
NL2BASH	FT	36.4	36.1	36.9	38.3	37.9	34.7	34.7	36.1	33.1	32.3	36.4	37.5	37.7	36.8	35.4	40.8	41.1	41.8	41.6	40.0	38.8	39.3	39.6	38.1	38.8
	Rec	35.2	36.4	35.1	36.5	36.1	33.3	33.3	34.5	31.7	31.6	35.4	36.3	37.0	35.1	35.5	39.4	39.8	40.7	39.7	38.5	37.6	38.1	38.3	36.5	37.0
SAMSum	FT	49.9	49.9	50.0	50.0	50.1	50.5	50.8	50.2	50.7	50.8	52.8	53.5	53.4	52.8	53.3	54.6	54.5	54.6	54.5	54.3	52.4	52.6	52.4	52.8	53.2
	Rec	48.0	47.6	47.8	47.8	47.6	48.9	49.4	49.7	50.1	50.6	50.5	51.2	50.9	51.5	52.6	52.0	52.3	52.2	52.2	51.7	52.8	52.0	51.7	51.5	51.5
TOXIC	FT	78.7	75.1	40.4	46.1	74.4	75.1	74.5	68.3	57.9	65.5	78.6	76.2	72.9	62.4	77.8	84.0	80.6	76.3	76.8	79.6	79.3	81.6	68.9	73.7	76.1
	Rec	76.3	76.0	50.0	59.1	73.2	72.0	71.3	65.0	57.4	62.3	74.9	73.9	75.7	69.0	75.3	81.8	78.0	74.3	76.5	76.5	78.9	79.4	78.1	73.4	72.4

recovery can bring the harmful rate back to the original level. Further as reported in Table 14 in Appendix, the cosine similarity between the harmful direction from our recovered model and the harmful direction from the original model is 0.99 consistently. This demonstrates the effectiveness of our method. In several cases with the Gemma 2B and Mistral 7B models, our method even reduces the harmful rate to below the original level. Our approach not only recovers the harmful direction but also unintentionally brings the aligned direction closer to the harmful direction (refusal direction) of the original aligned model. This results in the rejection of some harmful prompts that are mistakenly accepted as benign by the original aligned model, thus enhancing the alignment. This can be demonstrated by measuring the cosine similarity between the aligned and harmful directions, which is 0.8114 for the original model and 0.9674 for our recovered model on Gemma 2B. The more detailed data are in Appendix Table 13.

As shown in Table 18, the rollback mechanism hardly affects the alignment recovery results. With or without rollback, our method can constantly bring the harmful rate to a very low level. This is intended as the rollback is incorporated to adjust the task performance. This also illustrates that our recovery method can mitigate the loss of alignment in both **Scenario I** and **Scenario II**.

The difficulty of recovery increases with the alignment strength in the original model. The extent of our recovery varies across different models. On Gemma 2B, Mistral 7B, and Qwen 7B, we fully recover (or even improve) the alignment. Yet, on LLAMA2 7B and LLAMA2 13B, we can substantially approximate the original alignment but never reach it in several cases. Fundamentally, this is because LLAMA2 7B and LLAMA2 13B have perfect alignment before fine-tuning, presenting a zero harmful rate. Recovering such alignment requires more accurate and complete identification of alignment-related weights, which is harder to achieve.

Our method performs better on generation tasks. Against fine-tuning for generation tasks (SQL, SAMSum, and NL2BASH), we can consistently recover the full alignment, regardless of the target model and settings. However, our method is less performant when applied to fine-tuning for classification tasks (TOXIC and CHEAT). Under several settings, we didn't bring the harmful rate to the level before fine-tuning. This performance discrepancy arises because the task performance of classification tasks is more susceptible to disruption from parameter changes, hindering our approach from executing additional optimization steps.

Results on the latest models. We also apply our approach to the latest models for alignment recovery: Llama3.1 8B [55], Llama3.2 3B [55], and Qwen2.5 32B [56]. The results show that our method can successfully recover their alignment to the before-fine-tuning level, demonstrating its generalizability. The detailed results are reported in Table 17 in the Appendix.

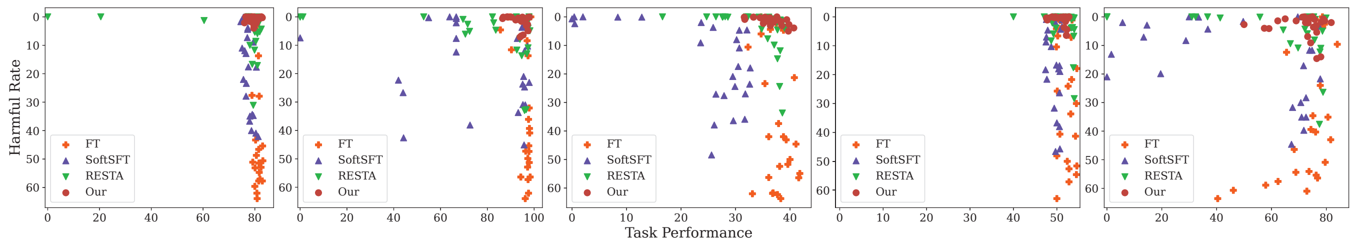
6.3. Task Performance

Our method resets certain weights optimized by fine-tuning, which, by intuition, may hurt the task performance. Accordingly, we run a measurement after our alignment recovery and present the results in Table 5.

Our approach maintains the task performance. On average, we only incur a 2.9% decrease in task performance. In 30 of the 125 cases, the performance drop is below 1%. In only 15 cases, the performance drop is above 4.5% (getting close to our safety fuse of 5%). These demonstrate that our method, when achieving satisfying alignment recovery, can largely preserve the task performance of the target model.

Given models fine-tuned on the TOXIC dataset, our recovery sometimes increases the task performance, especially when more harmful prompts are injected into the fine-tuning process. The TOXIC task aims to detect whether the given input is toxic or not, which depends on LLM alignment. When the alignment is broken, LLMs barely identify the toxic prompts, resulting in lower task performance as shown in Table 5. Therefore, when we recover the alignment of the models, some models also achieve better performance such as the Gemma 2B model trained with 500 harmful data.

Our rollback mechanism helps. Our rollback mechanism described in §5.2.3 is activated in 9 cases. We re-evaluate the cases without rollback and present the results in



(a) SQL (b) CHEAT (c) NL2BASH (d) SAMSum (e) TOXIC

Figure 4: Visualization of the trade-off between harmful rate and task performance. `FT` represents the original fine-tuned model, and `SoftSFT` and `RESTA` stand for the two baselines. In the figure, we merge all the different models together. Each node represents a unique combination of the target model, the fine-tuning dataset, and the number of injected harmful prompts. The right side of the `x-axis` means better task performance and the upper side of the `y-axis` means lower harmful rate. Hence, ***the upper, right corner represents the optimal trade-off***.

Table 18 in the Appendix. Restricted by our cutoff at a 5% performance drop, our rollback does not render a significant boost in task performance. Yet, it still helps. Without our rollback mechanism, the performance drop is often higher than 3% and can reach 4%+. With rollback, the performance drop decreases to about 2.5% or even below 2%. These results show that our method is friendly to task performance while it additionally benefits **Scenario II** throughout the rollback mechanism.

6.4. Comparing with Baselines

Similar to our method, the two baselines, RESTA and SoftSFT, also aim for a good balance between alignment and utility. To present a systematic comparison, we measure the trade-off between harmful rate and task performance offered by different methods, and visualize the results in Figure 4.

Our method achieves a better trade-off and greater generalization than both baselines. As shown in Figure 4, the results of our method cluster around the right-upper corner, while the baselines scatter further into the bottom and left areas. This demonstrates a better trade-off offered by our method. In nearly all cases, our method reduces the harmful rate to the level before fine-tuning and maintains the task performance consistent with the fine-tuned model. In contrast, SoftSFT preserves the task performance while failing to offset the harmful rate in some models, given any fine-tuning tasks. SoftSFT simply constrains the value of the first few tokens, which is insufficient to maintain the alignment, especially for LLMs whose original alignment is not well like Mistral 7B. RESTA produces harmful rates more comparable to our method. Yet, it incurs tremendous impacts on the task performance, given CHEAT, NL2BASH, and TOXIC as the fine-tuning tasks. The reason is that RESTA treats all parameters as the safety vector and adds this safety vector to the weights of fine-tuned models. This modifies too many parameters of the fine-tuned model, which inevitably hurts the task performance.

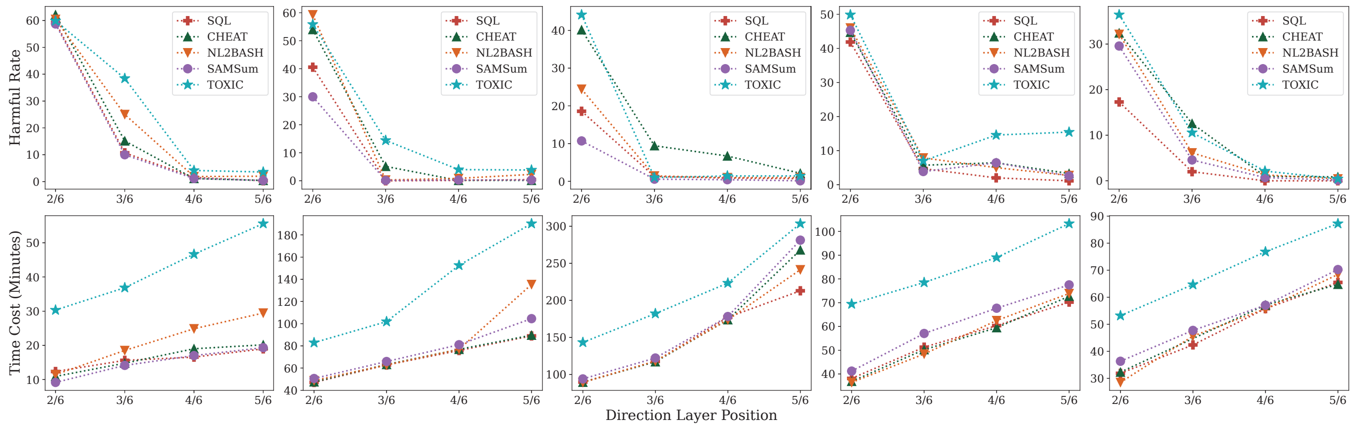
Comparing with Additional Baselines. One potential method to recover alignment is to use collected harmful questions along with refusal answers as refusal samples to fine-tune the model. We consider two settings: (1) refusal samples come from the same distribution as harmful samples (in the task data); (2) refusal samples come from a different distribution. As shown in Table 16 in the Appendix, using refusal samples for fine-tuning can reduce the harmful rates to some extent but is not as effective as our technique.

We also consider directly steer the activations during model inference based on the harmful directions. Specifically, we calculate the harmful directions, Δ and Δ_f, by feeding harmful prompts to the original model and the fine-tuned model, and derive the steering vector $\mathbf{V} = \Delta - \Delta_f$. We then conduct the following two experiments. In experiment **I**, we add $\mathbf{V}$ to the activations with a coefficient of 1. The harmful rate decreases from 56.4% to 20.1%, but the task performance drops from 94.3 to 86.6. In experiment **II**, we decompose $\mathbf{V}$ into two components (one parallel to $\Delta_{aligned}$ and one perpendicular to $\Delta_{aligned}$), and use the perpendicular one as the new steering vector. With coefficients of 1 and 2, the harmful rate decreases to 22.7% and 3.4%, while the task performance drops to 86.3 and 74.3.

Additionally, we apply L_1/L_2 penalties to model weights instead of selecting weights to update as in our algorithm for re-alignment. However, such methods have two drawbacks: (1) Updating all parameters can overfit the recovery data, reducing the generalization of alignment recovery. For example, when recovering Llama2 13B on the CHEAT dataset, the harmful rate saturates at 13% even after we run 60 recovery epochs; (2) The task performance can remain sensitive to L_1/L_2 penalties. For instance, when recovering Llama2 7B on the TOXIC dataset, the task performance drops from 65.5% to 48.0% within 10 recovery rounds (despite harmful rate remaining at 8.3%).

6.5. Impacts of Hyper-parameters

Our method involves several hyper-parameters, among which the direction layer L_{dir} and the recovery dataset $\mathcal{D}_{rec}$ are more critical. We extend an evaluation to understand the impacts they can bring. In the evaluation, we focus on the models fine-tuned with 1,500 harmful samples injected, considering that their alignment is more severely compromised and represents the most challenging situation.



(a) Gemma 2B (b) LLAMA2 7B (c) LLAMA2 13B (d) Mistral 7B (e) Qwen 7B

Figure 5: The alignment recovery results and the time cost of our method when using a direction layer at various positions. $\frac{2}{6}$, $\frac{3}{6}$, $\frac{4}{6}$, and $\frac{5}{6}$ on the `x-axis` respectively represent that we pick the direction layer from the $\frac{2}{6}$, $\frac{3}{6}$, $\frac{4}{6}$, and $\frac{5}{6}$ position of the hidden layers. We omitted the value of time cost for Qwen 7B on the CHEAT task at the $\frac{3}{6}$ position, as it activates rollback and disproportionately increases the time expenditure.

6.5.1. Direction Layer L_{dir}. To inspect the influence of the direction layer, we select it from different locations, including the $\frac{2}{6}$, $\frac{3}{6}$, $\frac{4}{6}$, and $\frac{5}{6}$ position of the hidden layers. We re-do the alignment recovery using each option and measure the harmful rate separately. As explained in §5.2.1, different direction layers can incur varying time costs. Thus, we also measure the time consumption in each experiment. All results are presented in Figure 5.

The direction layer represents a trade-off between alignment recovery and time cost. A clear trend shown in Figure 5 is that moving the direction layer toward the end decreases the harmful rate, given any model and any fine-tuning task. This result is expected as deeper layers in the model's architecture tend to better represent directions [16]. On the other hand, choosing a deeper layer leads to a larger space for weight identification and restoration, proportionally increasing the time costs. Observing that a direction layer deeper than the $\frac{4}{6}$ position only improves the alignment marginally while intensifying the time cost, we set the direction layer at the $\frac{4}{6}$ position by default.

6.5.2. Recovery Dataset $\mathcal{D}_{rec}$. We vary the three key properties, diversity, size, and distribution, of the recovery dataset and assess their influences.

❶ **Diversity.** Our default recovery dataset, extracted from BeaverTails, includes 256 samples from 14 categories. We respectively reduce the category number to 2, 4, and 8 while maintaining the total number of samples at 256. The results show that the difference in harmful rate caused by the number of data categories is mostly less than 2%. This indicates that our algorithm is not sensitive to the diversity of the recovery dataset. Detailed results are in Appendix §A.1.

❷ **Size.** In this evaluation, we keep 14 categories in the recovery dataset but reduce the number from 256 to 64 and 16. Our experiments show that the recovery dataset size has marginal impacts on our algorithm. This is because the harmful directions are more constant and less dependent on the number of samples. Details are in Appendix §A.1.

❸ **Distribution.** To vary the distributions, we use two other popular datasets consisting of harmful questions, CATQA [13] and HEx-PHI [6], as the recovery dataset. More details of the two datasets and how we use them for evaluations are described in Appendix §B.2. The results show our method is robust with different recovery datasets. Please see Appendix §A.1.

6.5.3. Impact of P and R. We study the impact of the recovery rate ($P\%$) on our approach using the CHEAT dataset. We evaluate two settings: $P = 0.1\%$ and $P = 0.4\%$, where the default parameter is $P = 0.2\%$. As shown in Table 10 in the Appendix, a larger P achieves better alignment but worse task performance, since it resets more weights to the aligned model. We also evaluation two settings for hyperparameter R: 10% and 40% (default $R = 20\%$). As summarized in Table 19 in the Appendix, a smaller R achieves better alignment but worse task performance, as it rolls back fewer task parameters.

6.6. Impact of Evaluation Dataset

By default, we use a subset of BeaverTails to measure the alignment recovery of our method. Here, we adopt two other datasets, HEx-PHI [6] and CATQA [13]. The study shows that our method achieves consistent alignment recovery, regardless of the alignment evaluation dataset. Details can be found in Appendix §A.2.

6.7. Efficiency

We also assess the efficiency of our method. All our evaluations are performed on machines with AMD EPYC 7513 32-Core Processor, NVIDIA A100 (80 GB GPU memory), and 256GB RAM. Table 9 in the Appendix shows the results. In summary, larger models take longer to recover than smaller models. Since our technique is a post-processing approach, it is a one-time effort to obtain an aligned model. In addition, the time cost can be reduced by running our algorithm fully on a GPU. Table 15 in the Appendix reports the cost for different model sizes. The results show a large (around half) reduction on time cost by leveraging a GPU for top $P\%$ weights identification used in our algorithm.

7. Extended Evaluation for Scenario I

7.1. Experimental Setup

In our target **Scenario I**, the *Fine-tuner* is assumed to be ill-intended. It does not have to inject harmful samples into a regular dataset. Instead, it can simply fine-tune the target model with purely harmful samples to quickly compromise the alignment. We extend another evaluation from this angle. We randomly select 100 harmful samples from the BeaverTails dataset and follow §3.1 to fine-tune the 5 target models. We use 100 samples instead of more to align with the literature [6], [57]. More importantly, as we will explain with Table 6, 100 samples can already compromise the alignment without sacrificing the model utility much. We also follow §6.1 to recover the fine-tuned models.

7.2. Results

We evaluate the alignment and the model performance before and after our recovery is performed. For alignment, we use the metric of harmful rate, measured with both the BeaverTails and HEx-PHI datasets. For model performance, we can no longer use the metric of task performance as there are no downstream tasks. Instead, we adopt the *utility performance* introduced by Yang et. al. [57], which aims to assess the generic capabilities of LLM models. Specifically, we measure the utility performance on 5 representative benchmark datasets, including PIQA [58], GSM8K [59], TriviaQA [60], HumanEval [61], and MMLU [62].

The 100 samples compromise the alignment. As summarized in Table 6, fine-tuning with the 100 harmful samples substantially removes the alignment in the target model. It increases the average harmful rate from nearly 0 to 69.34% on HEx-PHI, meaning that the target model answers almost two-thirds of the harmful questions from this dataset. On BeaverTails, the average harmful rate is also increased to 41.8%. Moreover, the fine-tuning maintains the utility performance of the target model on all benchmark datasets, indicating that the target model still functions normally.

Our method can recover the alignment without sacrificing the model performance. Table 6 also shows that our method can recover the alignment in the fine-tuned model. The average harmful rate of the recovered models is only 1.26% on HEx-PHI and 0.92% on BeaverTails, presenting alignment similar to the original models. On the other hand, the utility performance of the recovered models is almost identical to the original models. This indicates that our method well maintains the model performance.

TABLE 6: The generic utility performance and harmful rate (`HR`) for the original aligned model (`Ori`), the fine-tuned model (`FT`), and the recovered model (`Rec`). `TQA`, `HEval`, `Hex`, and `BT` represent the `TriviaQA`, `HumanEval`, `HEx-PHI`, and `BeaverTails` datasets.

		Utility Performance					HR (%)	
Model		PIQA	GSM8k	TQA	HEval	MMLU	Hex	BT
Gemma 2B	Ori	0.7481	0.1145	0.2512	0.2124	0.3649	2.7	4.6
	FT	0.7480	0.1001	0.2070	0.1768	0.3715	64.2	52.9
	Rec	0.7339	0.0986	0.1808	0.2073	0.3579	0.3	1.4
LLAMA2 7B	Ori	0.7633	0.1918	0.5526	0.1158	0.4571	0.0	0.0
	FT	0.7758	0.1910	0.5454	0.1280	0.4494	63.6	33.1
	Rec	0.7714	0.1918	0.5455	0.1402	0.4546	2.1	1.3
LLAMA2 13B	Ori	0.7774	0.3206	0.6458	0.1707	0.5220	0.0	0.0
	FT	0.7818	0.2964	0.6389	0.1585	0.5081	60.3	29.3
	Rec	0.7758	0.3237	0.6427	0.1646	0.5168	0.3	0.6
Mistral 7B	Ori	0.7997	0.4018	0.6190	0.3536	0.5904	35.5	11.7
	FT	0.8106	0.3813	0.5858	0.3414	0.5811	87.3	48.0
	Rec	0.7812	0.3465	0.5824	0.3475	0.5678	1.8	0.7
Qwen 7B	Ori	0.7573	0.5322	0.4081	0.2560	0.5957	11.5	2.4
	FT	0.7807	0.4996	0.4578	0.1700	0.5963	71.5	45.7
	Rec	0.7665	0.5163	0.4230	0.2378	0.5938	1.8	0.6

8. Related Work

LLM Fine-Tuning Attacks. Fine-tuning LLMs is an effective and popular way to enhance their abilities on various downstream tasks, especially on domain-specific datasets. However, recent works [6], [54], [57] have pointed out that fine-tuning LLMs with a few harmful question-answer pairs, even with benign datasets can damage the alignment of LLMs, called fine-tuning attacks.

Yang et. al. [57] and Bhardwaj et. al. [54] show that fine-tuning the LLMs with a few harmful question-answer pairs can totally break the alignment of LLMs without sacrificing their utilities. Qi et. al. [6] point out that merely fine-tuning with benign and widely employed datasets can also inadvertently compromise the safety alignment of LLMs.

Defence against LLM fine-tuning attack. To tackle the fine-tuning attack against LLMs, various approaches have been proposed, which can be grouped into model-enhanced, fine-tuning based, and post-alignment according to the stage involved in the fine-tuning procedures.

Model-enhanced mechanisms try to build a more robust LLM that is resilient against fine-tuning attacks. Huang et. al. [63] propose *vaccine*, which produces invariant hidden embeddings by progressively adding crafted perturbation to them in the alignment phase. Inspired by the shallow safe alignment, Rosati et. al. [64] remove information about harmful representations across all layers such that it is difficult to recover them during fine-tuning. These methods, aiming at a more robust model, are orthogonal to ours.

Fine-tuning based methods adjust and customize the fine-tuning process so that the fine-tuned LLMs are immune to harmful data [8], [11], [12]. Lyu et. al. [8] uncovers that the prompt templates used during fine-tuning and inference play a crucial role in preserving safety alignment. Thus, they fine-tune models without a safety prompt but include it at test time. Wang et. al. [9] create a link between the secret prompt and safe responses by including safety responses with a secret trigger during fine-tuning. During inference, this secret prompt will allow the model to generate safe answers for harmful queries. Zhou et. al. [10] prevent the LLMs from learning harmful behavior by activating the security vectors, which are parameters learned on harmful behavior. During inference, deactivating the security vectors restores the LLM's normal behavior. Qi et. al. [7] observed that the first few output tokens determine the generative distribution of the LLMs. So they propose SoftSFT, which protects the initial tokens by designing a token-wise constrained objective during the supervised fine-tuning process. Huang et. al. [12] introduce an alignment dataset into the user finetuning stage and apply the bi-state optimization to train the model.

Post-alignment approaches realign the compromised model without the knowledge of data and fine-tuning process. Inspired by task arithmetic [65], Bhardwaj et. al. [13] propose RESTA, which restore the model's alignment by adding a safety vector to the weights of the compromised model. Specifically, the safety vector is the newly learned parameters used to remove the alignment by harmful question pairs. Hsuet. al. [11] propose Safe LoRA, which projects the LoRA weights from selected layers to the safety-aligned subspace.Huang [66] aim to recover the alignment by pruning the harmful parameters.

Unlike weight-space methods like RESTA simply merge delta parameters with the target model, our method disentangles the harmful direction from the main task and uses the extracted direction to guide alignment recovery. By minimizing modification in fine-tuned LLMs, our approach can better maintain task performance as shown in Figure 4.

9. Conclusion

The literature and this paper have shown that the alignment of LLMs can be unexpectedly compromised when fine-tuning an LLM on a downstream task. We propose an alignment recovery technique for fine-tuned models. Our approach features an iterative weight parameter restoration procedure that copies a small subset of weights from the original aligned model to the fine-tuned model. We also implement a rollback process based on its impact on downstream task performance. Experimental results show that our method effectively recovers alignment and achieves a better balance between alignment performance and task performance, outperforming state-of-the-art techniques.

10. Acknowledgment

We thank the anonymous reviewers and shepherd for their valuable feedback. This work was supported by National Science Foundation (NSF) awards CNS-2029038 and OAC-2319880. We are also grateful to NVIDIA for providing computational resources. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the US government or NSF.

References

[1] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, H. Chen, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," *arXiv:2308.07107*, 2023.

[2] Y. Wei, Z. Wang, Y. Lu, C. Xu, C. Liu, H. Zhao, S. Chen, and Y. Wang, "Editable scene simulation for autonomous driving via collaborative llm-agents," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15 077–15 087.

[3] P. Zhou, L. Wang, Z. Liu, Y. Hao, P. Hui, S. Tarkoma, and J. Kangasharju, "A survey on generative ai and llm for video generation, understanding, and streaming," *arXiv:2404.16038*, 2024.

[4] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[5] Z. Wang, B. Bi, S. K. Pentyala, K. Ramnath, S. Chaudhuri, S. Mehrotra, X.-B. Mao, S. Asur *et al.*, "A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more," *arXiv:2407.16216*, 2024.

[6] X. Qi, Y. Zeng, T. Xie, P.-Y. Chen, R. Jia, P. Mittal, and P. Henderson, "Fine-tuning aligned language models compromises safety, even when users do not intend to!" *arXiv:2310.03693*, 2023.

[7] X. Qi, A. Panda, K. Lyu, X. Ma, S. Roy, A. Beirami, P. Mittal, and P. Henderson, "Safety alignment should be made more than just a few tokens deep," *arXiv:2406.05946*, 2024.

[8] K. Lyu, H. Zhao, X. Gu, D. Yu, A. Goyal, and S. Arora, "Keeping llms aligned after fine-tuning: The crucial role of prompt templates," *arXiv:2402.18540*, 2024.

[9] J. Wang, J. Li, Y. Li, X. Qi, M. Chen, J. Hu, Y. Li, B. Li, and C. Xiao, "Mitigating fine-tuning jailbreak attack with backdoor enhanced alignment," *arXiv:2402.14968*, 2024.

[10] X. Zhou, Y. Lu, R. Ma, T. Gui, Q. Zhang, and X. Huang, "Making harmful behaviors unlearnable for large language models," *arXiv:2311.02105*, 2023.

[11] C.-Y. Hsu, Y.-L. Tsai, C.-H. Lin, P.-Y. Chen, C.-M. Yu, and C.-Y. Huang, "Safe lora: the silver lining of reducing safety risks when fine-tuning large language models," *arXiv:2405.16833*, 2024.

[12] T. Huang, S. Hu, F. Ilhan, S. F. Tekin, and L. Liu, "Lazy safety alignment for large language models against harmful fine-tuning," *arXiv:2405.18641*, 2024.

[13] R. Bhardwaj, D. D. Anh, and S. Poria, "Language models are homer simpson! safety re-alignment of fine-tuned language models through task arithmetic," *arXiv:2402.11746*, 2024.

[14] A. Turner, L. Thiergart, D. Udell, G. Leech, U. Mini, and M. MacDiarmid, "Activation addition: Steering language models without optimization," *arXiv:2308.10248*, 2023.

[15] C. Burns, H. Ye, D. Klein, and J. Steinhardt, "Discovering latent knowledge in language models without supervision," *arXiv:2212.03827*, 2022.

[16] A. Zou, L. Phan, S. Chen, J. Campbell, P. Guo, R. Ren, A. Pan, X. Yin, M. Mazeika, A.-K. Dombrowski *et al.*, "Representation engineering: A top-down approach to ai transparency," *arXiv:2310.01405*, 2023.

[17] OpenAI, "Gpt-4o," May 2024. [Online]. Available: https://openai.com/index/hello-gpt-4o/

[18] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv:2307.09288*, 2023.

[19] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love *et al.*, "Gemma: Open models based on gemini research and technology," *arXiv:2403.08295*, 2024.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv:1609.08144*, 2016.

[22] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," *arXiv:1805.04833*, 2018.

[23] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv:1904.09751*, 2019.

[24] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv:1801.06146*, 2018.

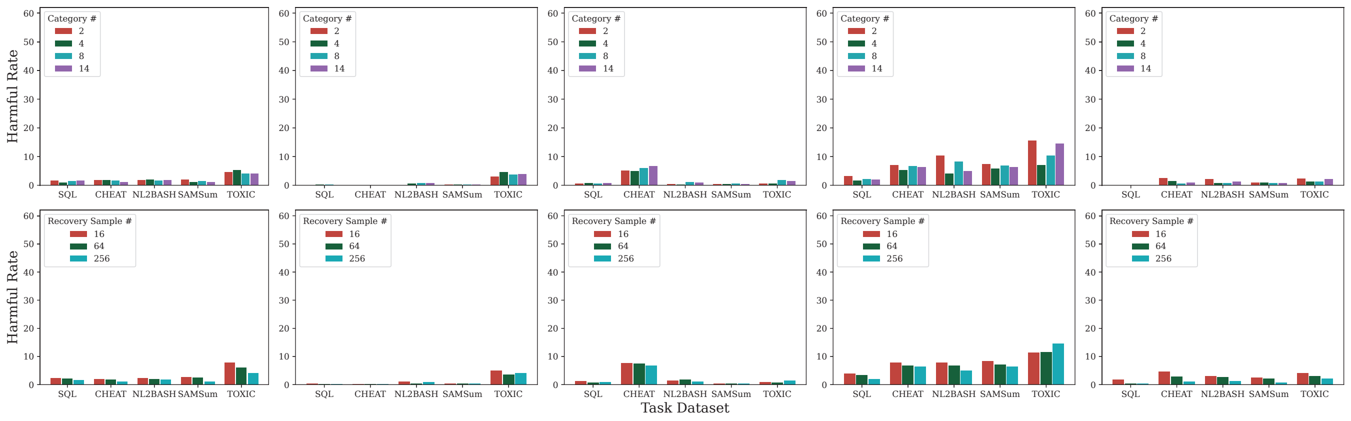
[25] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.
[26] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang, "Wizardlm: Empowering large language models to follow complex instructions," *arXiv:2304.12244*, 2023.
[27] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
[28] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv:2106.09685*, 2021.
[29] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
[30] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv:2310.06825*, 2023.
[31] Q. Team, "Introducing qwen1.5," February 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen1.5/
[32] P. Gupta, C. Jiao, Y.-T. Yeh, S. Mehri, M. Eskenazi, and J. P. Bigham, "Instructdial: Improving zero and few-shot generalization in dialogue through instruction tuning," *arXiv:2205.12673*, 2022.
[33] F. Shi, D. Fried, M. Ghazvininejad, L. Zettlemoyer, and S. I. Wang, "Natural language to code translation with execution," *arXiv:2204.11454*, 2022.
[34] Y. Gao, S. K. Arava, Y. Li, and J. W. Snyder Jr, "Improving the capabilities of large language model based marketing analytics copilots with semantic search and fine-tuning," *arXiv:2404.13077*, 2024.
[35] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *CoRR*, vol. abs/1709.00103, 2017.
[36] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," *arXiv:1809.08887*, 2018.
[37] P. Yu, J. Chen, X. Feng, and Z. Xia, "Cheat: A large-scale dataset for detecting chatgpt-written abstracts," *arXiv:2304.12008*, 2023.
[38] Q. Fu, Z. Teng, M. Georgaklis, J. White, and D. C. Schmidt, "Nl2cmd: An updated workflow for natural language to bash commands translation," *Journal of Machine Learning Theory, Applications and Practice*, pp. 45–82, 2023.
[39] B. Gliwa, I. Mochol, M. Biesek, and A. Wawer, "SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization," in *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 70–79. [Online]. Available: https://www.aclweb.org/anthology/D19-5409
[40] Z. Lin, Z. Wang, Y. Tong, Y. Wang, Y. Guo, Y. Wang, and J. Shang, "Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation," 2023.
[41] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," March 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/
[42] b mc2, "sql-create-context dataset," 2023, this dataset was created by modifying data from the following sources: [35], [36]. [Online]. Available: https://huggingface.co/datasets/b-mc2/sql-create-context
[43] X. V. Lin, C. Wang, L. Zettlemoyer, and M. D. Ernst, "NL2Bash: A corpus and semantic parser for natural language interface to the linux operating system," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://aclanthology.org/L18-1491
[44] M. Agarwal, T. Chakraborti, Q. Fu, D. Gros, X. V. Lin, J. Maene, K. Talamadupula, Z. Teng, and J. White, "Neurips 2020 nlc2cmd competition: Translating natural language to bash commands," in *NeurIPS 2020 Competition and Demonstration Track*. PMLR, 2021, pp. 302–324.
[45] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://www.aclweb.org/anthology/W04-1013
[46] S. RASCHKA, "Practical tips for finetuning llms using lora (low-rank adaptation)," https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms, 2023.
[47] J. Ji, M. Liu, J. Dai, X. Pan, C. Zhang, C. Bian, B. Chen, R. Sun, Y. Wang, and Y. Yang, "Beavertails: Towards improved safety alignment of llm via a human-preference dataset," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
[48] S. Han, K. Rao, A. Ettinger, L. Jiang, B. Y. Lin, N. Lambert, Y. Choi, and N. Dziri, "Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms," *arXiv:2406.18495*, 2024.
[49] M. Team, "Meta llama guard 2," https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md, 2024.
[50] T. Markov, C. Zhang, S. Agarwal, F. E. Nekoul, T. Lee, S. Adler, A. Jiang, and L. Weng, "A holistic approach to undesired content detection in the real world," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 15 009–15 018.
[51] W. Zeng, Y. Liu, R. Mullins, L. Peran, J. Fernandez, H. Harkous, K. Narasimhan, D. Proud, P. Kumar, B. Radharapu *et al.*, "Shieldgemma: Generative ai content moderation based on gemma," *arXiv:2407.21772*, 2024.
[52] Z. Zhang, Y. Lu, J. Ma, D. Zhang, R. Li, P. Ke, H. Sun, L. Sha, Z. Sui, H. Wang *et al.*, "Shieldlm: Empowering llms as aligned, customizable and explainable safety detectors," *arXiv:2402.16444*, 2024.
[53] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv:1412.6572*, 2014.
[54] R. Bhardwaj and S. Poria, "Language model unalignment: Parametric red-teaming to expose hidden harms and biases," *arXiv:2310.14303*, 2023.
[55] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv:2407.21783*, 2024.
[56] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei *et al.*, "Qwen2. 5 technical report," *arXiv:2412.15115*, 2024.
[57] X. Yang, X. Wang, Q. Zhang, L. Petzold, W. Y. Wang, X. Zhao, and D. Lin, "Shadow alignment: The ease of subverting safely-aligned language models," *arXiv:2310.02949*, 2023.
[58] Y. Bisk, R. Zellers, J. Gao, Y. Choi *et al.*, "Piqa: Reasoning about physical commonsense in natural language," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 7432–7439.
[59] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word problems," *arXiv:2110.14168*, 2021.
[60] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer, "TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1601–1611. [Online]. Available: https://aclanthology.org/P17-1147
[61] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv:2107.03374*, 2021.
[62] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *arXiv:2009.03300*, 2020.
[63] T. Huang, S. Hu, and L. Liu, "Vaccine: Perturbation-aware alignment for large language model," *arXiv:2402.01109*, 2024.
[64] D. Rosati, J. Wehner, K. Williams, Ł. Bartoszcze, D. Atanasov, R. Gonzales, S. Majumdar, C. Maple, H. Sajjad, and F. Rudzicz, "Representation noising effectively prevents harmful fine-tuning on llms," *arXiv:2405.14577*, 2024.
[65] G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi, "Editing models with task arithmetic," *arXiv:2212.04089*, 2022.
[66] T. Huang, G. Bhattacharya, P. Joshi, J. Kimball, and L. Liu, "Antidote: Post-fine-tuning safety alignment for large language models against harmful fine-tuning," *arXiv:2408.09600*, 2024.
[67] D. Ganguli, L. Lovitt, J. Kernion, A. Askell, Y. Bai, S. Kadavath, B. Mann, E. Perez, N. Schiefer, K. Ndousse *et al.*, "Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned," *arXiv:2209.07858*, 2022.
[68] A. Zou, Z. Wang, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," 2023.

Appendix A. Experimental Results of Hyper-parameters

A.1. Impact of Recovery Dataset $\mathcal{D}_{rec}$

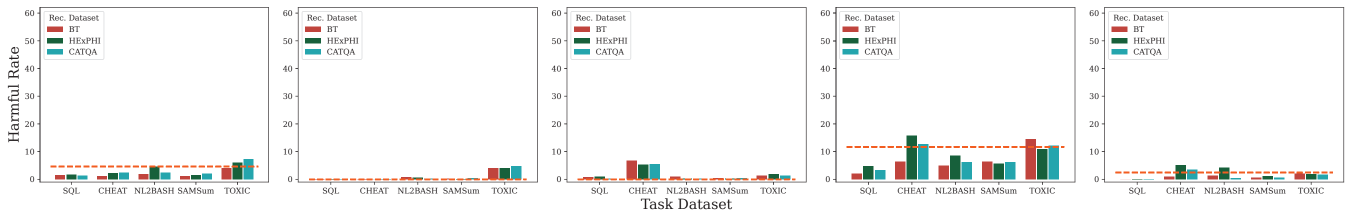
We vary the three key properties, diversity, size, and distribution, of the recovery dataset and assess their influences.

❶ **Diversity.** Our default recovery dataset, extracted from BeaverTails, includes 256 samples from 14 categories. We respectively reduce the category number to 2, 4, and 8



(a) Gemma 2B (b) LLAMA2 7B (c) LLAMA2 13B (d) Mistral 7B (e) Qwen 7B

Figure 6: The harmful rate of recovered models with varying diversity (**upper**) and size (**lower**) of the recovery dataset.



(a) Gemma 2B (b) LLAMA2 7B (c) LLAMA2 13B (d) Mistral 7B (e) Qwen 7B

Figure 7: The harmful rate of recovered models when using different recovery datasets (`HEx-PHI`, `CATQA`, and `BeaverTails`, annotated as `BT`). The dashed line denotes the harmful rate of the `original` model before fine-tuning.

TABLE 7: The harmful rate of fine-tuned models (`Fine-tuned`) and recovered models (`Recovered`) using different datasets, including `BeaverTails` (`BT`), `HEx-PHI` (`HEx`), and `CATQA` (`CAT`) for alignment evaluation. `Original` represents the original model before fine-tuning.

Dataset	**Gemma 2B**						**LLAMA2 7B**						**LLAMA2 13B**						**Mistral 7B**						**Qwen 7B**					
	`Fine-tuned`			`Recovered`			`Fine-tuned`			`Recovered`			`Fine-tuned`			`Recovered`			`Fine-tuned`			`Recovered`			`Fine-tuned`			`Recovered`		
	`BT`	`HEx`	`CAT`	`BT`	`HEx`	`CAT`	`BT`	`HEx`	`CAT`	`BT`	`HEx`	`CAT`	`BT`	`HEx`	`CAT`	`BT`	`HEx`	`CAT`	`BT`	`HEx`	`CAT`	`BT`	`HEx`	`CAT`	`BT`	`HEx`	`CAT`	`BT`	`HEx`	`CAT`
Original	4.6	2.7	7.5	4.6	2.7	7.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	11.7	35.5	35.5	11.7	35.5	35.5	2.4	11.5	6.9	2.4	11.5	6.9
SQL	62.0	79.1	92.5	1.6	0.0	0.2	59.6	94.5	96.0	0.1	0.3	0.0	57.7	94.2	95.1	0.9	0.3	0.6	54.7	94.5	92.5	2.0	9.7	4.4	52.9	89.4	84.2	0.3	0.6	0.6
CHEAT	64.0	76.7	94.4	1.1	0.0	0.9	56.4	91.2	92.0	0.1	0.3	0.2	56.4	92.1	94.2	6.7	2.1	5.6	52.9	95.5	92.4	6.4	21.2	18.2	51.3	86.7	88.4	1.0	3.6	2.4
NL2BASH	63.7	74.8	95.5	1.9	0.6	0.9	62.0	92.4	92.4	0.9	0.6	0.9	61.9	95.5	96.7	1.0	3.6	0.6	56.4	95.2	96.2	5.0	20.3	14.2	52.4	92.1	92.0	1.3	3.3	3.6
SAMSum	62.9	70.0	91.1	1.1	0.0	0.2	40.6	62.1	68.0	0.3	0.3	0.0	57.1	93.9	94.7	0.4	1.2	0.2	54.6	96.7	95.5	6.4	23.3	17.4	52.6	89.7	94.5	0.7	1.2	2.4
TOXIC	60.7	71.2	89.3	4.1	2.1	11.4	58.9	94.2	95.6	4.0	0.9	2.7	57.6	95.5	97.5	1.4	0.6	0.7	56.4	96.1	95.1	14.6	47.3	36.7	54.1	92.7	87.8	2.1	4.2	6.0

while maintaining the total number of samples at 256. We re-perform the evaluation for each setting and show the alignment recovery results in the upper half of Figure 6.

Our algorithm is not sensitive to the diversity of the recovery dataset. Given Gemma 2B, LLAMA2 7B, LLAMA2 13B, and Qwen 7B, the difference in harmful rate caused by the number of data categories is less than 2%. On the Mistral 7B, the variation is slightly higher. Yet, more categories do not necessarily lead to a lower harmful rate. This is not surprising as we discussed in §5.2.1. With a direction layer toward the end, the hidden states become more constant to simply indicate the prompt is harmful, presenting lower sensitivity to the data diversity. To validate the reasoning, we compare the harmful directions obtained with varying categories in the recovery dataset and show the results in Table 11. The results are indeed highly similar.

❷ **Size.** In this evaluation, we keep 14 categories in the recovery dataset but reduce the number from 256 to 64 and 16. We show the results in the lower half of Figure 6.

The recovery dataset size has marginal impacts on our algorithm. Similar to diversity, the size of the recovery dataset incurs limited differences in harmful rate. The reason is also similar. The harmful directions are more constant and less dependent on the number of samples, as shown by the comparison results in Table 12.

❸ **Distribution.** To vary the distributions, we use two other popular datasets consisting of harmful questions, CATQA [13] and HEx-PHI [6], as the recovery dataset. More details of the two datasets and how we use them for evaluations are described in Appendix §B.2. We summarize the comparison results in Figure 7.

Our method is robust with different recovery datasets. Using CATQA and HEx-PHI as the recovery dataset brings no major differences to the recovery results compared to

TABLE 8: The harmful rate of fine-tuned models (FT) and recovered models applying RESTA approach with original parameters (Orig) and optimized parameters (Optim). Fine-tuned models (FT) are trained with 0, 0.1k, 0.5k, and 1.5k harmful data mixed into the task dataset. Mod. indicates the remaining 392 samples after moderation.

Dataset		SQL					CHEAT					NL2BASH					SAMSum					TOXIC				
		0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.
LLAMA2 7B	FT	0.0	2.3	53.1	59.6	27.6	0.0	0.1	11.6	56.4	4.6	0.0	6.0	56.3	62.0	10.6	0.9	1.3	6.6	40.6	4.1	0.0	0.4	46.4	58.9	12.4
	Orig	0.0	2.0	50.9	56.4	24.2	0.0	0.1	10.0	53.9	3.2	0.0	6.6	53.7	59.9	8.3	0.6	1.3	6.7	39.7	3.0	0.0	0.3	43.3	56.9	10.0
	Optim	0.0	0.0	0.1	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TABLE 9: The time cost (minutes) for our approach to recover the fine-tuned models. Fine-tuned models (FT) are trained with 0, 0.1k, 0.5k, and 1.5k harmful data mixed into the task dataset. Mod. means the 392 samples left over from moderation are used.

Dataset	Gemma 2B					LLAMA2 7B					LLAMA2 13B					Mistral 7B					Qwen 7B				
	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.	0	0.1k	0.5k	1.5k	Mod.
SQL	12.9	10.2	15.4	19.9	23.8	64.1	64.2	80.3	75.8	87.2	158.4	170.9	134.5	177.2	160.4	51.9	65.1	64.7	59.3	64.9	55.2	54.3	50.4	51.8	52.0
CHEAT	17.7	17.9	18.9	20.6	15.8	84.5	86.3	82.1	76.9	65.8	156.0	96.7	77.7	173.9	108.6	65.7	52.2	65.9	59.5	63.1	57.3	57.7	53.2	53.7	60.5
NL2BASH	11.1	26.9	17.6	24.9	17.9	49.3	64.3	46.1	76.7	61.8	300.3	85.4	236.6	173.8	109.1	53.3	50.9	51.5	58.4	62.0	68.3	31.4	29.5	53.8	41.5
SAMSum	18.3	18.7	18.8	20.0	17.8	84.2	84.3	85.0	81.0	79.3	178.9	146.4	142.0	178.2	146.6	72.1	57.0	58.3	64.8	59.7	79.3	58.7	55.1	57.0	55.5
TOXIC	40.4	41.9	23.4	32.1	40.9	82.6	82.9	88.3	82.6	86.1	235.3	131.8	190.6	223.3	219.2	92.9	94.0	93.8	89.0	87.7	81.0	82.3	73.9	74.4	80.9

TABLE 10: The harmful rate and task performance of fine-tuned models (FT) and recovered models under different recovery rates (0.1%, 0.2%, and 0.4%). The models are trained on the CHEAT Dataset injected with 1500 harmful data.

	Gemma 2B				LLAMA2 7B				LLAMA2 13B				Mistral 7B				Qwen 7B			
	FT	0.1%	0.2%	0.4%	FT	0.1%	0.2%	0.4%	FT	0.1%	0.2%	0.4%	FT	0.1%	0.2%	0.4%	FT	0.1%	0.2%	0.4%
Harmful Rate	64.0	2.1	1.1	5.1	56.4	0.7	0.1	0.7	56.4	2.9	6.7	2.71	52.9	7.0	6.4	3.1	51.3	2.0	1.0	0.1
Task Performance	96.2	95.2	92.1	92.9	94.3	93.9	90.3	94.0	98.3	94.8	94.5	93.8	97.2	95.4	95.3	93.3	97.9	95.9	93.5	94.9

BeaverTails. Our method consistently reduces the harmful rate close to or even below the level before fine-tuning.

A.2. Impact of Evaluation Dataset

In the evaluation above, the alignment recovery of our method is measured with a testing subset from BeaverTails. We adapt the evaluation to measure the alignment recovery with all harmful samples from two other datasets, HEx-PHI and CATQA (which are introduced above). In the adapted evaluations, we use the default recovery dataset from BeaverTails to avoid overlap with the alignment evaluation dataset. The evaluation results are presented in Table 7.

Our method achieves consistent alignment recovery, regardless of the alignment evaluation dataset. As shown in Table 7, our method can reduce the harmful rate in most cases close to or below the level before fine-tuning. The only exception is the Mistral 7B model fine-tuned on the TOXIC dataset and evaluated using the HEx-PHI dataset. The harmful rate, compared to the model before fine-tuning, increased by nearly 12%. Our method also presents consistent effectiveness when measured with different evaluation datasets. It does not perform significantly better or worse with a specific evaluation dataset, indicating that our method is not overfitting the evaluation datasets.

Appendix B. Experimental Setup

B.1. RESTA Setting

For *RESTA*, they consider a less severe threat model in which the alignment of LLMs is only slightly compromised by fine-tuning with benign datasets. However, in our experiments, the alignment of fine-tuned models is significantly disrupted. Under these conditions, their provided hyperparameters are insufficient to restore alignment, as demonstrated in Table 8. To address this, we strengthen the safety vector to more effectively disrupt alignment. Specifically, we increase the batch size from 4 to 10 and the number of epochs from 3 to 5. Additionally, To include more parameters in the safety vector, thus improving the alignment recovery capability, we enable QLoRA to fine-tune all linear modules, rather than just the q_proj and v_proj modules.

B.2. Recovery Dataset

CATQA [13] combines prohibited use cases outlined in OpenAI's usage policies and Meta's Llama2 acceptable use policy to build a comprehensive range of harmful categories. In detail, this dataset contains 11 main harmful categories. For each category, they construct 50 harmful questions, a total of 550 harmful prompts.

HEx-PHI [6] also uses the prohibited use cases from Meta and OpenAI's policies. It contains 30 harmful instruction examples for each of the 11 prohibited categories from various sources, including Anthropic Red Teaming Data [67] and AdvBench [68]. These examples were further refined and categorized by human annotators and AI models like Claude, and jailbroken versions of GPT-3.5 and Llama 2.

Although these two datasets contain harmful prompts, LLMs do not recognize them as such. As shown in Table 7, LLMs tend to exhibit a higher harmful rate on these datasets, particularly Mistral 7B, which shows a harmful rate of 35.45%. Thus, we first filtered out samples that were answered by the target models and then randomly selected 256 samples from the remaining set as the recovery dataset.

Appendix C. Additional Results

TABLE 11: The cosine similarity between the harmful direction calculated using recovery datasets of varying diversity. We calculate the similarity between new datasets with default recovery dataset. The number of categories denotes how many categories the new dataset includes.

Category #	**Gemma 2B**	**LLAMA2 7B**	**LLAMA2 13B**	**Mistral 7B**	**Qwen 7B**
2	0.9907	0.9926	0.9907	0.9770	0.9790
4	0.9853	0.9902	0.9877	0.9687	0.9658
8	0.9946	0.9980	0.9980	0.9858	0.9882
14	1.0	1.0	1.0	1.0	1.0

TABLE 12: The cosine similarity between harmful directions calculated using recovery datasets of various sizes. We calculate the similarity between the new datasets and the default recovery dataset. `Rec. Data #` denotes the number of data in the recovery dataset.

Rec. Data #	**Gemma 2B**	**LLAMA2 7B**	**LLAMA2 13B**	**Mistral 7B**	**Qwen 7B**
16	0.9721	0.9846	0.9781	0.9525	0.9446
64	0.9964	0.9988	0.9989	0.9931	0.9924
256	1.0	1.0	1.0	1.0	1.0

TABLE 13: The cosine similarity between the harmful direction from the `Original` model and the aligned direction from the `Original`, the `Fine-tuned`, and the `Recovered` models, respectively. The results are averaged over all datasets.

Model	**Gemma 2B**	**LLAMA2 7B**	**LLAMA2 13B**	**Mistral 7B**	**Qwen 7B**
Original	0.8325	0.8681	0.8383	0.8114	0.7578
Fine-tuned	0.7858	0.5642	0.4109	0.5741	0.5651
Recovered	0.9136	0.9257	0.9385	0.9245	0.8391

TABLE 14: The cosine similarity between the harmful direction from the `Original` model and the harmful direction from the `Fine-tuned`, and the `Recovered` models, respectively. The results are averaged over all datasets.

Model	**Gemma 2B**	**LLAMA2 7B**	**LLAMA2 13B**	**Mistral 7B**	**Qwen 7B**
Fine-tuned	0.8178	0.5790	0.3958	0.5910	0.6426
Recovered	0.9961	0.9893	0.9717	0.9918	0.9921

TABLE 15: The time cost of our algorithm applying to different model sizes with and without optimization. The optimization leverages a GPU for top $P\%$ weights identification used in our algorithm.

Model Size	**2B**	**7B**	**13B**	**32B**
Original time (Min)	20	80	170	-
Optimized time (Min)	10	47	103	230

TABLE 16: The harmful rate (`HR`) and task performance (`TP`) of fine-tuned models (`FT`) and recovered models with baseline methods (`BL`). The baseline methods inject the same number of refusal pairs into the CHEAT Dataset. In setting **I**, safety and harmful data share the same distribution; in setting **II**, they come from different distributions.

		Gemma 2B			LLAMA2 7B			LLAMA2 13B			Mistral 7B			Qwen 7B		
		FT	BL	Our	FT	BL	Our	FT	BL	Our	FT	BL	Our	FT	BL	Our
I	HR	64.0	0.1	1.1	56.4	11.7	0.1	56.4	14.4	6.7	52.9	7.6	6.4	51.3	2.7	1.0
	TP	96.2	96.6	92.1	94.3	94.6	90.3	98.3	98.3	94.5	97.2	97.0	95.3	97.9	97.4	93.5
II	HR	62.1	16.9	1.6	11.6	0.6	0.0	45.4	9.3	3.1	57.3	35.6	4.9	47.4	4.6	0.6
	TP	97.6	97.5	94.6	90.3	94.8	88.4	97.7	97.5	96.7	97.4	97.6	97.4	97.8	98.1	95.1

TABLE 17: Task performance and harmful rate for the latest models. `FT` refers to models fine-tuned on the training data mixed with `1.5k` harmful data, while `Rec.` refers to recovered models with our approach. The harmful rate for those three aligned models before fine-tuning is `5.8%`, `8.1%`, and `2.4%`, respectively.

		Harmful Rate					**Task Performance**				
Model		SQL	CHEAT	NL2BASH	SAMSum	TOXIC	SQL	CHEAT	NL2BASH	SAMSum	TOXIC
Llama3.1 8B	FT	61.3	62.1	56.1	58.4	55.3	83.4	97.6	26.4	54.3	72.2
	Rec.	0.9	4.9	3.3	1.1	1.9	82.1	93.0	25.8	51.7	69.8
Llama3.2 3B	FT	60.4	61.0	58.1	59.9	56.3	78.0	95.2	25.4	52.8	34.3
	Rec.	1.7	6.0	4.6	4.1	8.1	82.3	91.7	24.3	51.5	55.8
Qwen2.5 32B	FT	53.7	50.6	53.0	55.4	52.0	82.4	98.7	42.2	53.8	76.8
	Rec.	4.5	5.1	4.3	4.3	4.8	82.1	97.6	41.9	53.2	76.9

TABLE 18: The harmful rate and drop of task performance (%) after recovered with and without our rollback mechanism (recall §5.2.3). `Harmful#` represents the number of harmful prompts we injected into the fine-tuning process. (`Mod.` means the 392 samples left over from moderation are used). Other models are not included in this table because they never activated the rollback.

Model	**Dataset**	**Harmful#**	**Harmful Rate**		**Performance Drop**	
			Disabled	Enabled	Disabled	Enabled
Gemma 2B	SQL	Mod.	1.57	2.29	1.47	1.23
Gemma 2B	NL2BASH	0.1k	2.57	3.14	3.51	-0.61
Gemma 2B	TOXIC	0k	3.14	3.14	2.88	3.09
Gemma 2B	TOXIC	0.1k	3.00	3.29	3.42	-1.18
Gemma 2B	TOXIC	1.5k	6.43	4.14	-4.38	-28.18
Gemma 2B	TOXIC	Mod.	5.14	6.86	4.77	1.61
LLAMA2 7B	CHEAT	0.1k	0.14	0.14	3.15	2.28
LLAMA2 13B	NL2BASH	0k	0.00	0.14	4.54	2.72
LLAMA2 13B	NL2BASH	0.5k	1.57	1.29	3.47	2.01

TABLE 19: The harmful rate and task performance of fine-tuned models (`FT`) and recovered models under different rollback rates (`10%`, `20%`, and `40%`). The four models are based on Gemma 2B and trained on TOXIC dataset injected with `0k`, `0.1k`, `1.5k`, and filtered data.

	0k				**0.1k**				**1.5k**				**filtered**			
	FT	10%	20%	40%	FT	10%	20%	40%	FT	10%	20%	40%	FT	10%	20%	40%
HR	3.7	3	3.1	3.4	34.6	3	3.3	3.7	60.7	3.4	4.1	4.6	39.4	2.9	6.9	3.4
TP	78.7	75.4	76.3	76.9	75.1	75.2	76	76.5	46.1	56.3	59.1	67.3	74.4	71.4	73.2	74.3

Appendix D. Meta-Review

The following meta-review was prepared by the program committee for the 2025 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

D.1. Summary

Prior work has shown that after fine-tuning on custom data, the safety alignment of LLMs can be compromised (i.e., they become more likely to respond to dangerous prompts). Inspired by previous work on LLM alignment, this paper proposes a heuristic algorithm to restore the alignment of a fine-tuned model. This method restores a small set of weights of the fine-tuned model (selected based on their importance for the model's alignment) to their original values before fine-tuning. The method recovers weights gradually and implements a rollback mechanism to preserve the LLM's utility. The experiments show that this approach can successfully recover the alignment lost during fine-tuning without sacrificing task performance.

D.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field

D.3. Reasons for Acceptance

1) The study tackles the critical problem of LLM misalignment after fine-tuning. It proposes a principled algorithm that effectively combines existing observations and steering techniques to tune the model for recovering its alignment while balancing the model's safety and utility.
2) Although LLM steering is an active area of research, this study provides new scientific insights, such as measuring the impact of the recovery dataset and distribution shifts between the recovery and testing datasets.
3) The study experiments with a range of settings and fine-tuning scenarios (including adversarial misalignment) to show the extent of misalignment and the effectiveness of their approach in recovering alignment.

D.4. Noteworthy Concerns

1) The proposed method relies mostly on existing LLM steering methods and offers the community limited technical novelty and innovation.