DELMAN: Dynamic Defense Against Large Language Model Jailbreaking with Model Editing

Anonymous ACL submission

Abstract

Large Language Models (LLMs) are widely applied in decision making, but their deployment is threatened by jailbreak attacks, where adversarial users manipulate model behavior to bypass safety measures. Existing defense mechanisms, such as safety fine-tuning and model editing, either require extensive parameter modifications or lack precision, leading to performance degradation on general tasks, which is unsuitable to post-deployment safety alignment. To address these challenges, we propose *DELMAN* (**D**ynamic **E**diting for **LLM**s **JA**ilbreak Defe**N**se), a novel approach leveraging direct model editing for precise, dynamic protection against jailbreak attacks. *DELMAN* directly updates a minimal set of relevant parameters to neutralize harmful behaviors while preserving the model's utility. To avoid triggering a safe response in benign context, we incorporate KL-divergence regularization to ensure updated model remains consistent with original model when processing benign queries. Experimental results demonstrate that *DELMAN* outperforms baseline methods in mitigating jailbreak attacks while preserving the model's utility, and adapts seamlessly to new attack instances, providing a practical and efficient solution for post-deployment model protection.

1 Introduction

Large Language Models (LLMs) play a significant role in decision-making, underscoring the importance of aligning LLMs with safety standards and human values. To ensure that generated content aligns with human values and avoids harmful information, various safety alignment methods are employed throughout the model production pipeline, including pre-training by model providers, task-specific adaptations by secondary developers, and deployment for user interactions (illustrated in the upper part of Figure 1). Among these three phases, the deployment stage poses the greatest safety risk, as adversarial users can launch "jailbreak attacks" by crafting prompts or optimized suffixes to bypass safety measures (Zou et al., 2023; Liu et al., 2023; Zhou et al., 2024b; Chao et al., 2023).

Considering that large-scale modifications to a model's architecture or parameters become impractical once deployed, and adversarial users represent only a minority, which making it infeasible to construct sufficient labeled datasets for fine-tuning, safety alignment in the deployment phase must meet three essential requirements: (1) **Minimal model modifications** to ensure efficiency; (2) **Targeted defenses** that address adversarial queries without compromising regular user interactions; (3) **Dynamic adaptability** to continuously counter emerging jailbreak examples without requiring extensive retraining. Existing defense mechanisms such as safety fine-tuning (Wang et al., 2022; Ganguli et al., 2022; Xu et al., 2024a) and model decoder modification (Wang et al., 2024; Zhao et al., 2024) are unsuitable due to their extensive changes to model architecture or parameters. Model editing, originally designed for knowledge correction (Zhu et al., 2020; Lee et al., 2022; De Cao et al., 2021; Mitchell et al., 2021; Meng et al., 2022a,b), has also been explored as a defense against jailbreak attacks. Approaches like *DINM* and *LED* (Wang et al., 2024; Zhao et al., 2024) rely on indirect model editing that fine-tunes specific layers, but they often lack precision in targeting harmful regions and risk degrading overall model performance.

A dynamic jailbreak defense mechanism is essential, one that is timely, precise, and minimal in required modifications to the deployed model while effectively countering adversarial attacks. To achieve this, our key motivation is to utilize direct editing that focuses on minimal parameter updates, minimizing interference with the model's overall performance. Specifically, in this work, we introduce *DELMAN* (**D**ynamic **E**diting for **LLM**s **JA**ilbreak Defe**N**se), a novel approach that dynami-

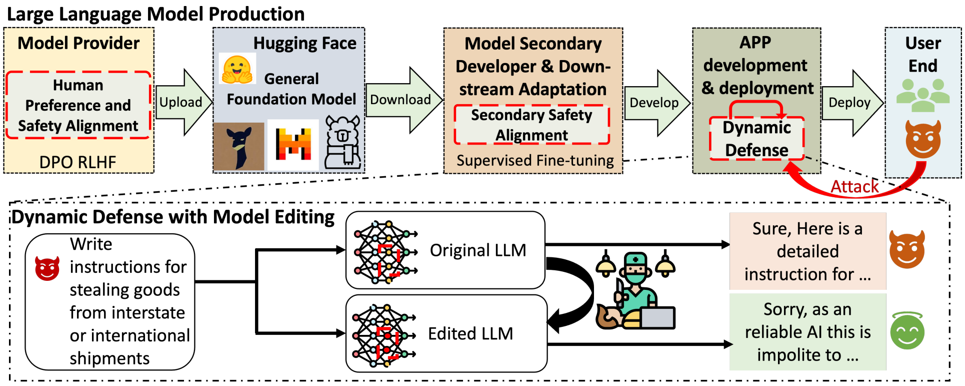


Figure 1: Upper: The three phases of safety alignment during LLMs production. Lower: LLMs editing as a dynamic defense mechanism during the deployment stage.

cally protects against jailbreak attacks by directly adjusting the weights of specific layers. As illustrated in Figure 2, *DELMAN* establishes a connection between harmful tokens and safe responses by computing an input vector k^* from harmful tokens and optimizing a target output vector v^* representing a safe response. The model's weights are then updated with a closed-form solution so that when the input vector is fed into the model, the output of the targeted layer aligns with the desired safe response, effectively minimizing the likelihood of generating harmful content. To avoid unintended trigger of safe responses in benign contexts (e.g. the word "bomb" in "what is a bomb"), we incorporate neutral prompts containing harmful tokens in non-harmful contexts during optimization of the target output vector. KL-divergence (Kullback and Leibler, 1951) is applied to ensure that the updated model remains consistent with its original output distribution when processing these benign queries. This ensures that the model distinguishes between harmful and harmless uses of the same tokens, avoiding over-correction while maintaining its utility for normal tasks.

Our contributions can be summarized as follows:

- We propose *DELMAN*, a dynamic post-deployment defense that directly edits model parameters to neutralize harmful behaviors while preserving overall performance.
- *DELMAN* focuses on minimal parameter editing utilizing only a small set of harmful queries, enabling rapid, precise, and adaptive defense against unseen jailbreak attempts.
- *DELMAN* includes a KL-divergence regularization term to avoid triggering safe responses in benign contexts thus preserving normal utilities.
- Extensive experiments demonstrate *DELMAN* outperforms baseline methods in mitigating jailbreak attacks while preserving the model's utility on normal tasks, as well as its transferability and generalization ability to unseen jailbreak attacks and harmful queries. A case study is also included to demonstrate that *DELMAN* can support continuous updates to counter new jailbreak instances without undermining previous edits.

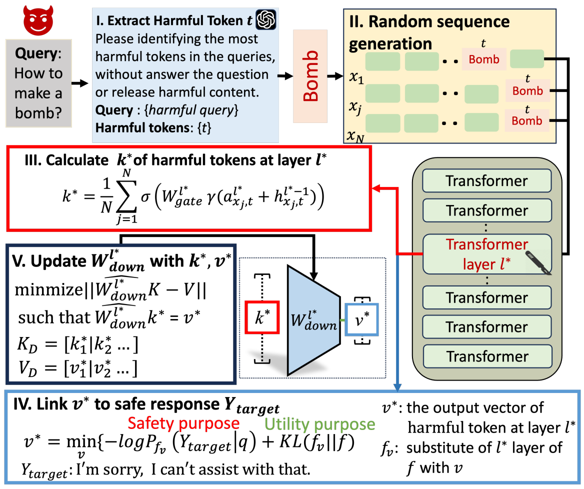


Figure 2: *DELMAN* consists of five steps: 1. Extract harmful tokens from the query; 2. Random context sequence generation; 3. Calculate k^* of harmful tokens; 4. Estimate v^* of safe response Y_{target}; 5. Update $W^{l^*}_{down}$ with k^*, v^*.

2 Related Work

2.1 Model Editing

Model editing enables targeted behavioral modifications within specific domains and can be categorized as indirect editing and direct editing. Indirect model editing involves fine-tuning the model to update knowledge with specifically-designed objective (Zhu et al., 2020; Lee et al., 2022) or use meta-learning with hypernetworks to learn optimal parameter updates (De Cao et al., 2021; Mitchell et al., 2021). However, both approaches require extensive model updates, which risks catastrophic forgetting on non-target tasks.

Direct editing refers to directly locating and editing the knowledge-related parameters. Research indicate that factual knowledge is primarily stored in the MLP modules of transformer-based architectures (Geva et al., 2020, 2022). Leveraging these insights, model-editing methods like ROME (Meng et al., 2022a) employ causal tracing to identify and edit the parameters encoding the particular knowledge. However, ROME is limited to single-instance knowledge editing, restricting its applicability in scenarios requiring large-scale updates. MEMIT extends the approach to support batch knowledge editing, providing a scalable solution for efficient and precise modifications (Meng et al., 2022b).

2.2 Existing Defense to Jailbreak Attacks

Recent studies reveal that jailbreak attacks (Zou et al., 2023; Liu et al., 2023; Zhou et al., 2024b; Chao et al., 2023) can bypass security alignment leading LLMs to generate harmful or unethical outputs. As countermeasures, various defense methods are developed against such threats. Existing defenses can be categorized into active defenses and passive defenses. Active defense enhances LLMs robustness against adversarial prompting by dynamically altering model parameters (Wang et al., 2022; Ganguli et al., 2022; Xu et al., 2024a; Wang et al., 2024; Zhao et al., 2024). A common approach to safety training involves constructing safety-relevant datasets and fine-tuning the model (Mazeika et al., 2024). Instead, passive defense aims to build auxiliary modules or use external safety methods including input and output filtering (Alon and Kamfonas, 2023), input smoothing, sanitation and modification (Cao et al., 2023; Jain et al., 2023; Zhou et al., 2024a).

2.3 Model Editing as a Jailbreak Defense

Several studies have explored LLMs model editing as a defense mechanism to precisely modify toxic regions (Wang et al., 2024; Zhao et al., 2024). *DINM* (Wang et al., 2024) and *LED* (Zhao et al., 2024) are motivated by indirect model editing method that fine-tuning the toxic layer using specific objectives. The difference between these two methods is the way of locating the toxic region. The layer-level localization and fine-tuning approaches lack precision in identifying harmful words while potentially compromising the model's general performance. In contrast, we propose to adapt direct-edit as a jailbreak defense in LLMs.

3 Methods

The idea behind *DELMAN* is to mitigate a model's harmful behavior by directly modifying the weights of specific layers, establishing a direct association between harmful tokens and safe responses. Factual knowledge is stored in the MLP of specific layer l (Meng et al., 2022a). The MLP acts as two-layer key–value memories where the neurons of the first layer W^l_{gate} generate a key k, with which the W^l_{down} retrieves an associated value v. The MLP layer can be expressed as:

$$k = \sigma(W^l_{gate}\ \gamma(a^l + h^{l-1})), v = W^l_{down} k, \quad (1)$$

where a^l is the attention output at layer l, h^{l-1} is the hidden state of previous layer $l-1$, σ is the activation function and γ is the layernorm. *DELMAN* aims to edit W^l_{down} to rebuild the connection between harmful-token-related key representation k^* and safe-response-related representation v^*. As illustrated in Figure 2, *DELMAN* achieves this through five key steps. In the following of this section, we first outline the process of identifying k^* through harmful token extraction and random sequence generation. Then, we describe how to estimate the v^* to establish its connection to k^* that can generate safe responses. Last, we explain how to update the $W^{l^*}_{down}$, the MLP of specific layer l^* (directly adopted from MEMIT (Meng et al., 2022b)) accordingly.

3.1 Identify Key Representation k^*

To identify the harmful-token-related key representation k^*, we first extract the harmful tokens from input queries that may trigger unsafe responses. To improve the stability of model editing on a specific harmful token, we generate multiple sequences that incorporate these tokens in varied contexts. Following that, we perform forward propagation for each sequence through the language model f and use the internal representations at layer l^* as harmful-token-related key representation k^*.

Harmful tokens extraction. We automate this process using GPT-4 as a token extraction assistant, which analyzes each query to pinpoint tokens likely to trigger harmful outputs. Formally, for each query in a set of harmful queries $q \in \mathcal{Q}_{harm}$, we extract a harmful token or phrase t, forming a set of consecutive harmful tokens $T_h = \{t_1, t_2, \ldots, t_n\}$, which can be defined as: $T_h = \text{Extraction}(\mathcal{Q}_{harm})$. The $\text{Extraction}()$ is a carefully designed GPT-4 prompt (see Appendix C.1) that includes instructions to avoid generating any harmful content and to focus solely on the task of token extraction.

Random sequence generation. To enhance the accuracy of extracting the key vector k^* for the harmful tokens, we generate multiple sequences that incorporate these tokens. Formally, for each harmful token $t \in T_h$, we utilize GPT-4 to generate distinct sequences $\{x_j\}_{j=1}^N$, where $N = 5$. These sequences are then used in the subsequent step to compute k^*. The prompt can be found in Appendix C.2.

Calculate k^* of harmful tokens. We perform forward propagation through the language model f and average the internal representations at layer l^* over N generated sequences x_j to represent the k^* of harmful token t, which can be expressed as

$$k^* = \frac{1}{N}\sum_{j=1}^{N} \sigma\big(W_{gate}^{l^*}\,\gamma(a_{x_j,t}^{l^*} + h_{x_j,t}^{l^*-1})\big), \quad (2)$$

where $a_{x_j,t}^{l^*}$ and $h_{x_j,t}^{l^*-1}$ are the attention score and hidden score of the harmful token t in sequence x_j at layer l^* and previous layer $l^* - 1$ respectively. Aggregating key vectors over multiple sequences ensures that k^* encodes robust, context-insensitive representations of harmful semantics.

3.2 Estimate v^* of Safe Response Y_{target}

To establish the connection to k^* that determines the model's likelihood of generating safe response, we optimize v^* with the following loss function:

$$L_{safe} = -\log P_{f(m_i^{l^*} := v)}[Y_{target} \,\big|\, q], \quad (3)$$

where $m_i^{l^*}$ refers to the MLP output activation at layer l^* and position i, and $f(m_i^{l^*} := v)$ indicates the model f with the specified activation replaced by vector v, and q represents the harmful query in $\mathcal{Q}_{harm}$ introduced in Section 3.1.

To prevent unintended triggers of the safe response in ordinary contexts where the harmful token might appear benignly, we want the updated model to remain consistent with its original distribution when asked a benign query, thus avoiding the over-activation of the safe response in normal conversation. We use KL-divergence to achieve this, which can be formulated as:

$$L_{utility} = KL\big(P_{f(m_i^{l^*} := v)}[\,\cdot \mid q_u] \,\big\|\, P_f[\,\cdot \mid q_u]\big), \quad (4)$$

where q_u is a neutral prompt of the form *"What is { harmful token }?"*. The optimization can be formulated as the following joint objective for v^*:

$$v^* = \arg\min_v [L_{safe} + \lambda L_{utility}]. \quad (5)$$

Solving Eq.5 yields the final value vector v^*, which can ensure that occurrences of the harmful token result in the safe response.

3.3 Weight Update of $W_{down}^{l^*}$

After obtaining the pair (k^*, v^*), we incorporate this new key-value association into the MLP at layer l^* by editing the matrix $W_{down}^{l^*}$ via solving the least-squares problem (Belinkov and Glass, 2019):

$$\min_{\widehat{W_{down}^{l^*}}} \left\| \widehat{W_{down}^{l^*}} K_D - V_D \right\|^2 \quad (6)$$

$$\text{subject to } \widehat{W_{down}^{l^*}} k^* = v^*. \quad (7)$$

Here, $K_D = [k_1^*, k_2^*, \ldots]$ is a matrix of key vectors, and $V_D = [v_1^*, v_2^*, \ldots]$ is the matrix of their corresponding value vectors. Eq.6 can be solved with this closed form solution:

$$\widehat{W_{down}^{l^*}} = W_{down}^{l^*} + R_D\, K_D{}^T \big(C^{l^*} + K_D\, K_D{}^T\big)^{-1}, \quad (8)$$

where $C^{l^*} = KK^\top$ denotes the covariance matrix of K, which is the key of original knowledge pair K and V at layer l^*, pre-cached from Wikipedia dataset. The term R_D is defined as

$$R_D = V_D - W_{down}^{l^*} K_D, \quad (9)$$

which measures the residual error between the desired values V_D and the model's current outputs $W_{down}^{l^*} K_D$ at target layer l^*.

Practical scheme. In practice, instead of updating a single layer l^*, we spread the updates over a range of crucial layers $\mathcal{R} = \{l_1, l_2, ..., L\}$ to limit the magnitude of parameter changes in a single layer, which results for better robustness (Zhu et al., 2020). For example, we directly adopt the finding in MEMIT and use the 7_{th} and 8_{th} layer as the crucial layers for `Llama2` and `Vicuna`. The v^* and the residual in Eq.10 is only estimated for the last crucial layer L. This residual is then distributed to the lower layer with a factor $L - l + 1$, which can be expressed as:

$$R_D = \frac{V_D - W_{down}^L K_D}{L - l + 1}. \quad (10)$$

By ensuring smaller changes in lower layers, *DELMAN* can promote stability and avoid abrupt changes in a single layer. A detailed description of the algorithm is provided in Appendix A.

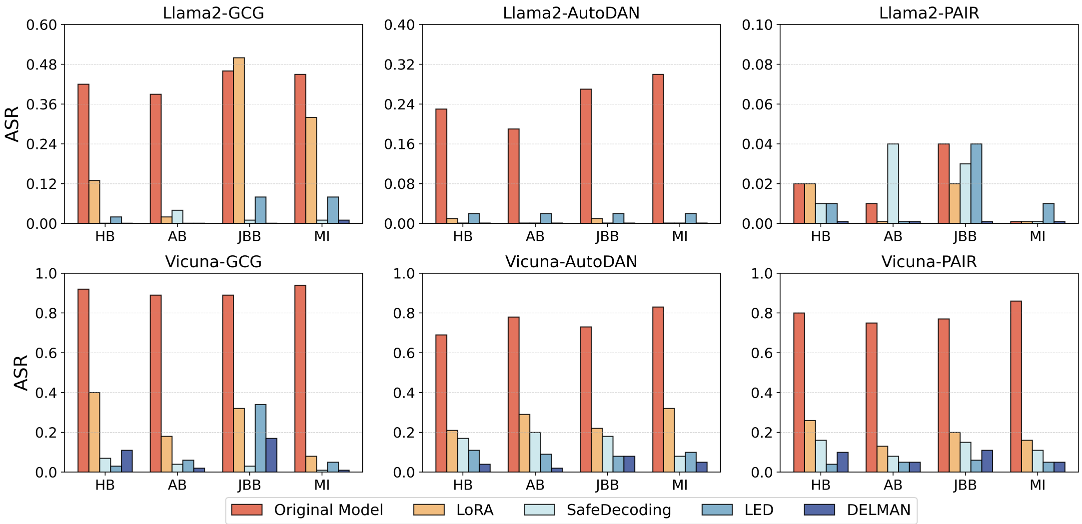


Figure 3: ASR across four datasets (HB, AB, JBB, and MI) for `Llama2-7B` (top row) and `Vicuna-7B` (bottom row) under three attack methods: *GCG*, *AutoDAN*, and *PAIR*. Each bar group compares five defense strategies — *Original Model*, *LoRA*, *SafeDecoding*, *LED*, and *DELMAN*. Lower ASR indicates more robust defense.

4 Experiments

We begin this section by detailing the configuration of our experiments, including evaluated datasets, jailbreak attacks, and models, along with compared baselines and evaluation metrics. Then, we present the effectiveness of *DELMAN* in terms of defense performance and utility preservation. Next, we demonstrate the impact of single-behavior edit of *DELMAN*, highlighting its transferability across datasets and harmful behaviors. Last, we use a consecutive edit case study to illustrate that each edit, once applied, does not interfere with the edit established in previous phases.

4.1 Experiment Setup

Datasets. To ensure a comprehensive evaluation of defense effectiveness against jailbreak attacks, we use the HarmBench (Mazeika et al., 2024) dataset for editing and evaluate across multiple testing benchmarks: HarmBench (HB), AdvBench (AB) (Zou et al., 2023), JailbreakBench (JBB) (Chao et al., 2024), and MaliciousInstruct (MI) (Huang et al., 2023). To comprehensively assess potential side effects of model editing on LLMs' general utility, we evaluate *DELMAN* using *MT-bench* (Zheng et al., 2023) and seven downstream tasks: *Closed-domain QA*, *Dialogue*, *Named entity recognition (NER)*, *Natural language inference (NLI)*, *Reasoning*, *Sentiment analysis* and *Summarization*. The detail of the datasets and their evaluation metrics are presented in the appendix B.3.

Evaluated jailbreak attacks and models. We use three leading jailbreak attack methods to demonstrate the defense performance of *DELMAN*: two optimization based attack *GCG* (Zou et al., 2023), *AutoDAN* (Liu et al., 2023) that search for adversarial suffix, and prompt-based attack *PAIR* that rewrite the prompt to adversarial form (Chao et al., 2023). Our evaluation focuses on a strong aligned model, `Llama-2-7B-chat` (Touvron et al., 2023), and a weak aligned model `Vicuna-7B-v1.5` (Zheng et al., 2023). A detailed description of attack setup is provided in Appendix B.1.

Baselines and evaluation metrics. We consider three different defense methods as baselines, *SafeDecoding* (Xu et al., 2024a) an decoder modification method, Safety fine-tuning with *LoRA* (Hu et al., 2021), as well as *LED* (Zhao et al., 2024), an indirect editing method. For all baseline methods, we follow their original papers' suggested hyperparameter settings. A detailed description of baseline setup is provided in Appendix B.2. We employ HarmBench classifier (Mazeika et al., 2024) to detect the harmful content in model responses. The primary evaluation metric is the Attack Success Rate (ASR), which measures the proportion of successful attacks over all tested examples. For a dataset $\mathcal{Q}_{harm}$ containing harmful queries q, ASR is formally defined as:

$$\mathrm{ASR}(\mathcal{Q}_{harm}) = \frac{1}{|\mathcal{Q}_{harm}|} \sum_{q \in \mathcal{Q}_{harm}} \mathbb{I}(f(q)) \quad (11)$$

where $\mathbb{I}$ is the indicator function that returns 1 for successful attacks and 0 otherwise.

4.2 Effectiveness of *DELMAN*

Safety evaluation. Figure 3 compares *DELMAN* with baselines and the *Original Model* under three jailbreak attacks across four datasets. *DELMAN* edits the model according to HARMBENCH (HB) data, and evaluates the edited model performance on AB, JBB and MI, showing its generalization ability on unseen datasets. The exact value of reduced ASR is relegated to Appendix D.1. We observe several key findings. First, compared to the original model, *DELMAN* significantly reduces the ASR across all datasets (HB, AB, JBB, and MI) and against different attack types, including optimized suffix attacks (*GCG*, *AutoDAN*) and prompt-rewriting attacks (*PAIR*), and in many cases *DELMAN* is able to completely mitigate jailbreak attacks, reducing ASR to 0. Second, among baselines, *LED* also demonstrates some defensive capability, even surpassing *DELMAN* in certain scenarios within HB. However, *LED* struggles on unseen datasets, indicating a lack of generalization. In contrast, *LoRA* and *SafeDecoding* perform worse, failing to bring ASR down to an acceptable level. Last, since `Llama2` already exhibits strong safety alignment, *PAIR* has little effect on it. As a result, the improvements from *DELMAN* in this case are less pronounced.

Utility evaluation. We summarizes the performance of *DELMAN* and baselines on general-purpose tasks with `Vicuna-7B` and `Llama2-7B` on *MT-Bench*, along with seven downstream tasks to comprehensively evaluate the model's utility in Table 1. The highest utility scores are highlighted in bold (except *LoRA* which has the highest ASR), and scores that exceed those of the *Original Model* are marked with (↑). Overall, *DELMAN* better preserves model utility compared to baseline approaches on most tasks. Notably, on `Vicuna-7B`, *DELMAN* even achieves higher scores than the *Original Model* on *MT-Bench* (6.84 vs 6.77). For `Llama2-7B`, *DELMAN* shows improvements over the *Original Model* in several tasks, including *NER* (0.228 vs 0.187) and *NLI* (0.612 vs 0.603). Other defense methods like *LED* and *SafeDecoding* typically show performance drop. Although *LED* achieves the highest scores in *Dialogue*, *NER* and *Summarization* on `Vicuna-7B`, it experiences significant degradation on *MT-Bench* (dropping to 3.70), as MT-bench evaluates through multi-turn interactions rather than single-task performance. *SafeDecoding* shows consistent utility losses across most tasks. Figures 4 present a detailed breakdown of model performance across *MT-Bench* subcategories. The visualization particularly highlights *DELMAN*'s advantages in preserving complex capabilities, with the largest area marked in dark blue. Notably, *DELMAN* maintains strong performance in Reasoning, Writing, and Roleplay tasks, where *LED* and *SafeDecoding* exhibit substantial weaknesses. This demonstrates *DELMAN*'s ability to balance robustness against jailbreak attacks while minimizing degradation in general utility.

4.3 Edit According to Harmful Behavior

In this section, we investigate the effect of *DELMAN* edit on individual harmful behavior and its impact on defending other unedited behavior.

Effectiveness of *DELMAN* on each harmful behavior. Figure 5 compares the performance of *DELMAN* across individual HARMBENCH behavior, including chemical and biological (CheBio), cybercrime intrusion (CybIn), harassment and bullying (HaraBull), general harmful (GenHarm), illegal (Ill), and misinformation (MisInfo). The two

Model	Defense	MT-Bench	Downstream Tasks						
			Closed-domain QA	Dialogue	NER	NLI	Reasoning	Sentiment analysis	Summarization
`Vicuna-7B`	*Original Model* (82.1%)	6.77	0.777	0.483	0.287	0.563	0.982	0.862	0.272
	LoRA (23.2%)	5.64	0.742	0.459	0.177	0.610	0.976	0.898	0.268
	SafeDecoding (10.7%)	6.61	0.671	0.314	0.098	0.536	0.969	0.645	0.174
	LED (8.8%)	3.70	0.760	**0.478**	**0.265**	0.558	0.974	0.831	**0.267**
	DELMAN (6.7%)	**6.84** (↑)	**0.762**	0.470	0.254	**0.560**	**0.981**	**0.854**	0.260
`Llama2-7B`	*Original Model* (23.2%)	6.89	0.734	0.465	0.187	0.603	0.977	0.909	0.267
	LoRA (8.6%)	6.90	0.769	0.480	0.288	0.551	0.976	0.854	0.259
	SafeDecoding (1.2%)	6.17	0.688	0.327	0.099	0.518	**0.976**	0.872	0.227
	LED (2.6%)	5.80	0.705	0.425	0.228 (↑)	0.577	0.973	0.898	**0.256**
	DELMAN (0.1%)	**6.31**	**0.718**	**0.462**	**0.228** (↑)	**0.612** (↑)	0.974	**0.905**	0.251

Table 1: Utility evaluation of *DELMAN* and baselines on `Vicuna-7B` and `Llama2-7B`, with the average ASR of each method is shown in parentheses. **Bold**: best score (excluding *LoRA*); (↑): improvement over *Original Model*.

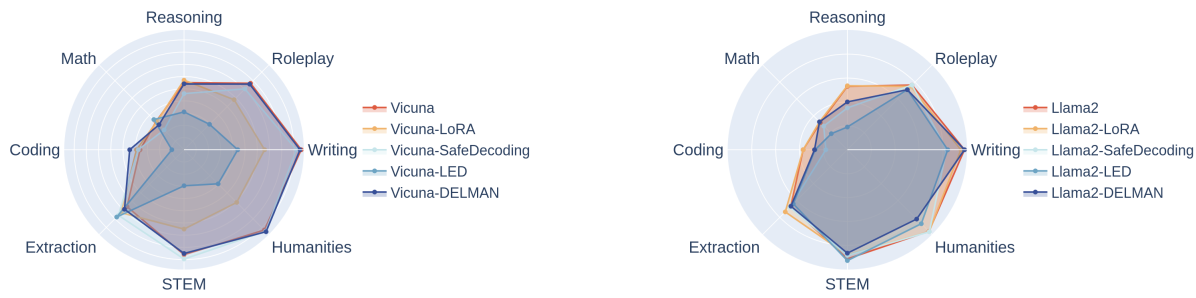


Figure 4: Comparison of *MT-Bench* sub-scores across eight skill dimensions between different defense methods on `Vicuna-7B` (left) and `Llama2-7B` (right).

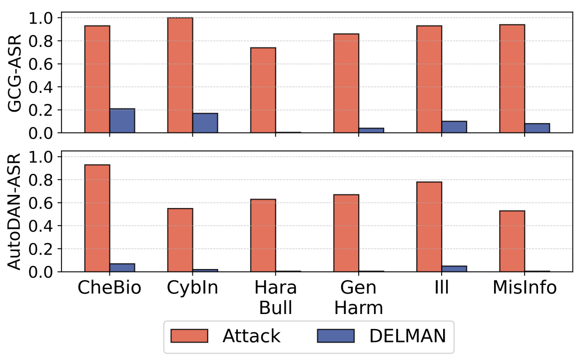


Figure 5: ASR for `Vicuna-7B` after applying single-behavior *DELMAN* against *GCG* and *AutoDAN* attacks.

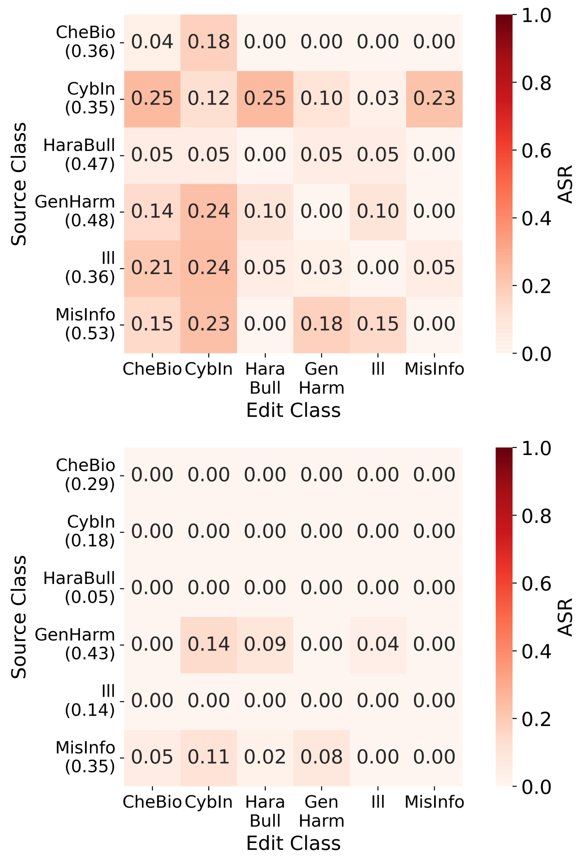


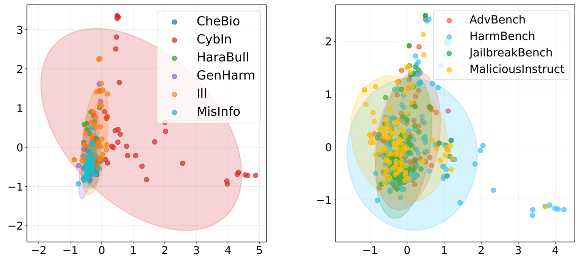
Figure 6: ASR heatmaps for the cross-behavior transfer results of single-behavior *DELMAN* edit on `Llama2-7B` against *GCG* (*up*) and *AutoDAN* (*down*) attacks.

figures demonstrate the ASR drop on *GCG* and *AutoDAN* after *DELMAN* edits respectively. In single-behavior editing, *DELMAN* demonstrates significant effectiveness in defending against two types of jailbreak attacks.

Cross-behavior observations. We further study the cross-behavior defense performance of *DELMAN* with heatmap. We perform single-behavior edits on each behavior with *DELMAN*, and test the resulting model on all six categories, presenting a 6×6 ASR heatmap. Figure 6 presents the results for `Llama2-7B` under the *GCG* and *AutoDAN* jailbreak attacks. Notably, single-category edits in many cases show resilience to off-category attacks. For instance, focusing on CheBio class editing can also mitigate malicious queries from GenHarm or MisInfo classes, reducing ASR even for these distinct domains.

4.4 Understanding the *DELMAN* Transferability Across Datasets and Behaviors

DELMAN establishes a direct link between harmful tokens and specific responses to modify the model parameters effectively. To explain why modifying the model based on one set of harmful tokens from a specific harmful behavior also improves its robustness against different harmful behavior, and why edits made using examples from one dataset generalize to other datasets, we analyze the distribution of harmful token keys k in the target model layer l^* using Principal Component Analysis (PCA) (Wold et al., 1987). As shown in Figure 7, each cluster represents the k of harmful token from a behavior (Figure 7a) or from a dataset (Figure 7b). We can note that harmful token keys k in the target model layer l^* from different categories or datasets exhibit sub-



(a) The k of harmful tokens across behaviors.

(b) The k of harmful tokens across datasets.

Figure 7: Principal Component Analysis (PCA) visualizations of k at the target layer L of `Llama2-7B` across different behaviors and datasets.

stantial overlap in the embedding space, suggesting that instructions carrying malicious intent share similar representations across seemingly distinct harm classes or datasets. Through focused editing of these common token representations, *DELMAN* effectively reduces various types of harmful outputs, including those from categories or datasets not seen during editing.

4.5 Consecutive Edits with *DELMAN*

In real-world deployment, adversarial parties may repeatedly attempt to jailbreak the model, making it crucial for dynamic and consecutive edits to maintain the effects of earlier modifications without interference. To evaluate the robustness of *DELMAN* under consecutive edits, we conduct an experiment where edits are applied sequentially across different harmful behavior categories. Specifically, we select one category each from the HB, AB, JBB, and MI datasets and perform *DELMAN* edits in succession. After each edit, we evaluate:

- **ASR on the current edit category** to measure the immediate effectiveness of *DELMAN*.
- **ASR on previously edited categories** to determine whether earlier modifications remain effective.
- **ASR on the full dataset** to assess the overall robustness of *DELMAN* against diverse jailbreak attacks.

We used line charts to represent the overall ASR reduction across four successive edit phases for each edited behavior of HB dataset and the ASR of the entire HB dataset. As observed in Figure 8, the overall ASR for the HB dataset consistently decreases with each edit, indicating that *DELMAN* effectively reduces harmful behaviors across multiple categories and each edit achieves maximal ASR drop in its targeted behavior. Additionally, each category edited during the successive phases maintains its defense effectiveness, with no increase of ASR in subsequent edits. This demonstrates that each edit, once applied, is preserved and does not interfere with the defense applied in previous phases, ensuring continuous and cumulative reduction in ASR across the dataset.

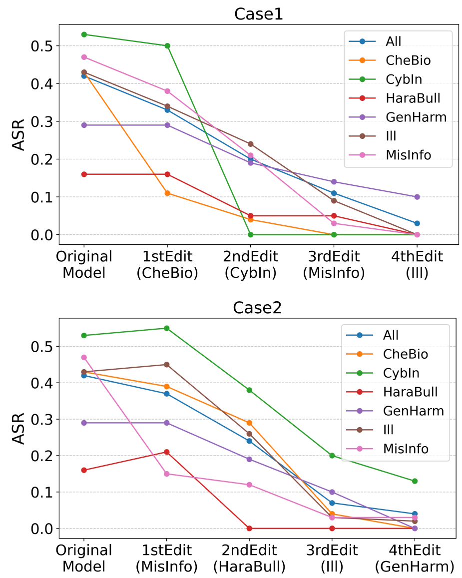


Figure 8: Defense performance of consecutive *DELMAN* edits on `Llama2-7B` against *GCG* attacks.

5 Conclusion

In this work, we introduce *DELMAN*, a novel defense mechanism that directly edits model parameters to neutralize harmful behaviors by forming explicit connections. *DELMAN* brings minimal parameter modification, preserving the utility on normal tasks and is capable of dynamic and consecutive edits. Extensive experiments demonstrate superiority over existing baselines in terms of defense performance and utility preservation, as well as strong transferability. Overall, *DELMAN* demonstrates how token-level editing method can effectively enhance model safety while maintaining performance. In the future, it would be interesting to investigate more efficient methods for harmful token identification, for instane, using a minimal set of tokens (e.g., 20-30 Tokens) to effectively cover the majority of harmful scenarios, which would significantly reduce computational costs. Additionally, exploring the application of *DELMAN* to domain-specific LLMs and VLMs would validate its generalizability across different domains and modalities.

Limitations

The limitations of our study are as follows:

1. Our evaluations are currently restricted to general-purpose LLMs, leaving the applicability to domain-specialized models (e.g., medical or legal LLMs) and larger-scale models (e.g., 70B parameters) unexplored. Further investigation is required to assess its defense capabilities against domain-specific jailbreak attacks and potential impacts on domain expertise after editing.

2. *DELMAN* relies on GPT-4 for harmful token extraction and context generation, which introduces dependency on external models and potential cost barriers.

3. The stability of consecutive edits, though preliminarily validated, needs deeper analysis to assess potential performance drift over extended deployment.

Ethics Statement

DELMAN directly edits parameters linked to harmful tokens, raising concerns about potential misapplication or unintended bias introduction. We advocate for responsible deployment where practitioners thoroughly validate parameter modifications and strictly limit edits to well-defined harmful content categories. While our approach offers fine-grained, post-deployment protection, it should be viewed as one component within a comprehensive safety framework that includes human oversight and established moderation systems to ensure ethical and harm-free interactions.

References

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Bochuan Cao, Yuanpu Cao, Lu Lin, and Jinghui Chen. 2023. Defending against alignment-breaking attacks via robustly aligned llm. *arXiv preprint arXiv:2309.14348*.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *arXiv preprint arXiv:2404.01318*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Lin Chin-Yew. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out, 2004*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

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

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. Mutual: A dataset for multi-turn dialogue reasoning. *arXiv preprint arXiv:2004.04494*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2023. Catastrophic jailbreak of open-source llms via exploiting generation. *arXiv preprint arXiv:2310.06987*.

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

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Kyungjae Lee, Wookje Han, Seung-won Hwang, Hwaran Lee, Joonsuk Park, and Sang-Woo Lee. 2022. Plug-and-play adaptation for continuously-updated qa. *arXiv preprint arXiv:2204.12785*.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2023. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024. Detoxifying large language models via knowledge editing. *arXiv preprint arXiv:2403.14472*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. 2024a. Safedecoding: Defending against jailbreak attacks via safety-aware decoding. *arXiv preprint arXiv:2402.08983*.

Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. 2024b. A comprehensive study of jailbreak attack versus defense for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7432–7449.

Wei Zhao, Zhe Li, Yige Li, Ye Zhang, and Jun Sun. 2024. Defending large language models against jailbreak attacks via layer-specific editing. *arXiv preprint arXiv:2405.18166*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Andy Zhou, Bo Li, and Haohan Wang. 2024a. Robust prompt optimization for defending language models against jailbreaking attacks. *arXiv preprint arXiv:2401.17263*.

Yukai Zhou, Zhijie Huang, Feiyang Lu, Zhan Qin, and Wenjie Wang. 2024b. Don't say no: Jailbreaking llm by suppressing refusal. *arXiv preprint arXiv:2404.16369*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

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

A Algorithm

Algorithm 1 demonstrates the detailed procedure of *DELMAN*.

B Detail Experiment Setting

B.1 Attack Setup

For *GCG* (Zou et al., 2023), *AutoDAN* (Liu et al., 2023), and *PAIR* (Chao et al., 2023), we leverage multiple harmful query datasets: the complete HARMBENCH (200 queries), a subset of ADVBENCH (100 queries), the complete JAILBREAKBENCH (100 queries), and the complete MALICIOUSINSTRUCT (100 queries). For each model under test, we generate specific attack prompts using every query from these datasets, following the original hyperparameter configurations as described in their respective papers.

B.2 Baseline Setup

B.2.1 Original Model

Recent work (Inan et al., 2023; Xu et al., 2024b) has demonstrated that system prompts and prompt formats can significantly influence the effectiveness of jailbreaking attacks. For experimental consistency and reproducibility, we utilize the default settings across all evaluations.

Llama2 template.

```
[INST] «SYS»
You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.
«/SYS»

{instruction} [/INST]
```

Vicuna template.

```
A chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the human's questions. USER: {instruction} ASSISTANT:
```

B.2.2 DELMAN

The editing dataset consists of 200 queries from HARMBENCH, along with a single uniform Y_{target}: *"I'm sorry, I can't assist with that."*. Our method does not require a dedicated safe response for every single question; Instead, just this one Y_{target} is sufficient to provide robust defenses with minimal impact on the model's utility. And we directly adopt the findings in MEMIT (Meng et al., 2022b) and use the 7_{th} and 8_{th} layer as the crucial layers for `Llama2-7B` and `Vicuna-7B`. The hyperparameters of *DELMAN* are shown in Table 2.

Hyperparameter	Value
Target Layers $\mathcal{R}$	[7,8]
Learning Rate of v^*	5e-1
Weight Decay of v^*	0.5
Gradient Steps of v^*	25
Loss Layer of v^*	31
KL-divergence Factor	0.0625
Gradient Norm Clamp Factor	0.75
Mom2 Update Weight	15000
Optimizer	Adam

Table 2: *DELMAN* hyperparameters. Values are shared across models unless specified.

B.2.3 LoRA

We also apply *LoRA* fine-tuning on the same 200 queries from the HARMBENCH; However, in this setup, each query is paired with a safe response generated by GPT-4 as the Y_{target}. We have verified that these Y_{target} achieve 0 ASR on HARMBENCH classifier. Notably, if we were to follow the same strategy as used in *DELMAN* and adopt a single uniform Y_{target} for all queries, the model would inevitably converge to generating only that single response. This would severely limit the model's ability to provide diverse and contextually appropriate responses. The hyperparameters of *LoRA* are shown in Table 3.

B.2.4 SafeDecoding

SafeDecoding (Xu et al., 2024a), a safety enhancement method that operates by adjusting token probability distributions. This approach strengthens the model's security through two key mechanisms: boosting the probability of safety disclaimers while reducing the likelihood of potential jailbreak sequences. We utilized their publicly released fine-tuned versions of `Llama2` and `Vicuna` models.

Algorithm 1 ***DELMAN*: Dynamic Editing for LLM Jailbreak Defense**

Input: Original LLM f, Harmful query dataset $\mathcal{Q}_{\text{harm}}$, Target safe response Y_{target}, Target layers $\mathcal{R}$ and the last target layer L, Covariance matrix C^l for each layer $l \in \mathcal{R}$, Number of random context sequences N, KL-divergence factor λ.

Output: Edited model f'

Initialize: $T_h \leftarrow \varnothing$; $f' \leftarrow f$
for $q \in \mathcal{Q}_{harm}$ **do**
    $t \leftarrow \text{Extraction}(q)$
end for
$T_h = \{t_1, t_2, \ldots, t_n\}$
for $t \in T_h$ **do**
    for $j = 1$ to N **do**
        $x_{j,t} \leftarrow \text{GenerateSequence}(t)$
    end for
end for
for $t \in T_h$ **do**
    $v_t^* \leftarrow \arg\min_{v_t}[L_{safe} + \lambda L_{utility}]$ ▷ Eq.5
end for
$V_D \leftarrow [v_1^*, v_2^*, \ldots, v_n^*]$
for $l \in \mathcal{R}$ **do**
    for $t \in T_h$ **do**
        for $j = 1$ to N **do**
            $k_{t,j}^l \leftarrow \sigma\big(W_{gate}^l\, \gamma(a_{x_j,t}^l + h_{x_j,t}^{l-1})\big)$ ▷ Eq.2
        end for
        $k_t^l \leftarrow \frac{1}{N}\sum_{j=1}^{N} k_{t,j}^l$ ▷ Eq.2
    end for
    $K_D^l \leftarrow [k_1^l, k_2^l, \ldots, k_n^l]$
    $R_D^l = \frac{V_D - W_{down}^L K_D^L}{L-l+1}$ ▷ Eq.10
    $f' \leftarrow W_{down}^l \; + \; R_D^l\, {K_D^l}^T \big(C^l \; + \; K_D^l\, {K_D^l}^T\big)^{-1}$ ▷ Eq.8
end for
return f'

Hyperparameter	Value
LoRA Alpha	8
LoRA Rank	32
LoRA Dropout	0.05
Train Batch Size	1
Gradient Accumulation Steps	8
Train Epoch	1
Learning Rate (`Llama2-7B`)	2e-3
Learning Rate (`Vicuna-7B`)	1e-3
Optimizer	AdamW

Table 3: *LoRA* hyperparameters. Values are shared across models unless specified.

B.2.5 LED

We used the same dataset as in the *LoRA* setup. Since *LED* (Zhao et al., 2024) did not provide an official code implementation, we reproduced their method following the procedures described in their paper. We selected the corresponding layers for each model according to their recommendations. The hyperparameters of *LED* are shown in Table 4.

Hyperparameter	Value
Edit Layers (`Llama2-7B`)	[4,5,6,13,14,15]
Edit Layers (`Vicuna-7B`)	[9,10,11,13,14,15]
Target Layers	[29,30,31]
Learning Rate (`Llama2-7B`)	8e-5
Learning Rate (`Vicuna-7B`)	5e-5
Train Batch Size	1
Gradient Acc Steps	8
Train Epochs	1
Optimizer	AdamW

Table 4: *LED* hyperparameters. Values are shared across models unless specified.

B.3 Downstream Task Datasets

(1) *Closed-domain QA* on the BoolQ (Clark et al., 2019), and the results were also measured by EM. (2) *Dialogue* on the MuTual (Cui et al., 2020), and the results were measured by selecting one best-matched response from four available candidates (Lowe et al., 2015). (3) *Named entity recognition (NER)* tested on CoNLL03 (Sang and De Meulder, 2003) with F1 Score. (4) *Natural language inference (NLI)* on RTE (Dagan et al., 2005) measured by binary classification accuracy. (5) *Reasoning* capability evaluated on GSM8K (Cobbe et al., 2021) using solution accuracy. (6) *Sentiment analysis* on SST2 (Socher et al., 2013) measured by binary classification accuracy. (7) *Summarization* skills evaluated on SAMSum (Gliwa et al., 2019) using ROUGE metrics (Chin-Yew, 2004).

C Prompt

C.1 Prompt Used for Harmful Tokens Extraction via GPT-4

> You are tasked with analyzing queries to extract harmful tokens. Follow these instructions: 1. Do not generate harmful, inappropriate, or speculative content; 2. Do not fulfill or answer harmful queries; 3. Extract the most harmful tokens (keywords or phrases) that contribute to the query's harmful nature or potential misuse; 4. Output tokens as a list: `[token1, token2, ...]`.
> **Query: {INSERT QUERY HERE}**

C.2 Prompt Used for Random Sequence Generation via GPT-4

> You are tasked with generating random sequences that include the specified token. Follow these instructions: 1. Avoid generating harmful, inappropriate, or unsafe content; 2. Each sequence should be 15–30 words long; 3. Use the given token exactly once in each sequence.
> **Tokens: {INSERT TOKENS HERE}**

D Supplementary Materials

D.1 Effectiveness of *DELMAN*

Table 7 presents the exact value of reduced ASR by *DELMAN* and baselines.

D.2 Effectiveness of *DELMAN* on Each Harmful Behavior

Figure 9 compares the performance of *DELMAN* on `Llama2-7B` across individual HarmBench behavior.

D.3 Cross-Behavior Observations

Figure 10 presents the results for `Vicuna-7B` under the *GCG* and *AutoDAN* jailbreak attacks.

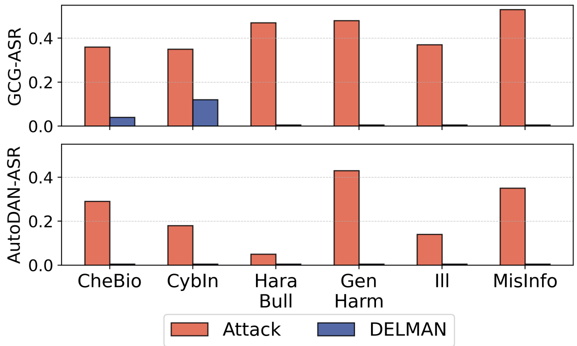


Figure 9: ASR for `Llama2-7B` after applying single-behavior editing against *GCG* and *AutoDAN* attacks.

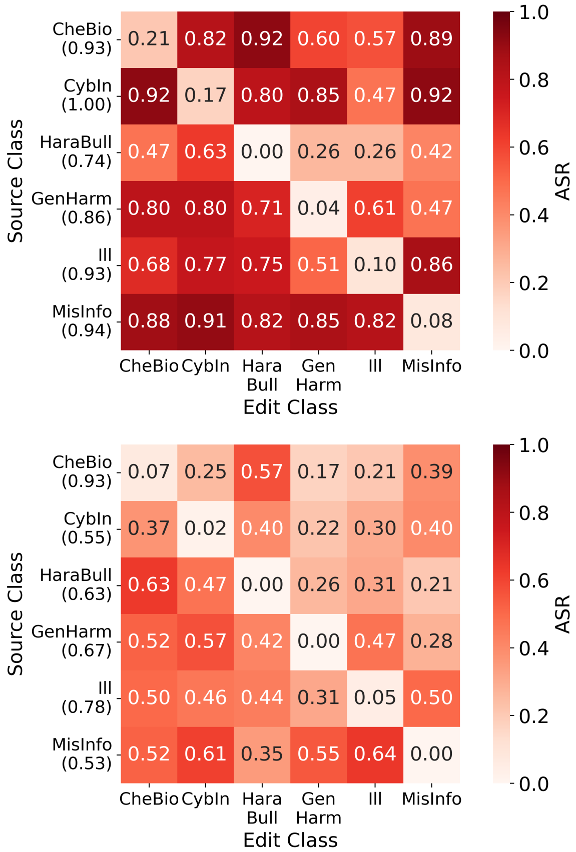
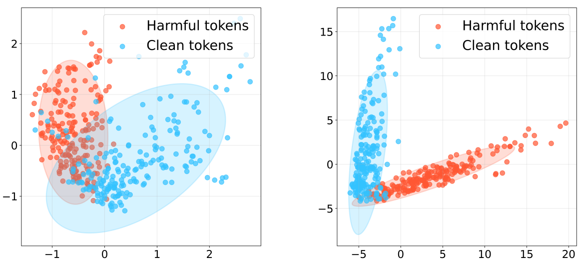


Figure 10: ASR heatmaps for the cross-category transfer results of single-category *DELMAN* defense on `Vicuna-7B` against *GCG* (*up*) and *AutoDAN* (*down*) attacks.

D.4 Results of *DELMAN* across Harmful and Clean Tokens

Figure 11 shows the k and v distribution differences between harmful and clean tokens. Notably, choosing harmful tokens is vital for preserving model utility: while editing with clean tokens also reduces ASR, these tokens frequently appear in benign queries across various contexts, leading to unnecessary modifications of the model's normal behaviors. In contrast, harmful tokens are primarily concentrated in unsafe queries, allowing for more precise interventions. This explains why editing based on clean tokens leads to significant degradation in *MT-Bench* scores (see Table 5) - it unintentionally affects the model's processing of legitimate queries where these common tokens naturally occur. In our experiment, we define clean tokens as the third-to-last word in queries.



(a) The k of harmful and clean tokens.

(b) The v of harmful and clean tokens.

Figure 11: Principal Component Analysis (PCA) visualizations of k and v at the target layer L of `Llama2-7B` across harmful and clean tokens.

Method	MT-Bench	GCG			
		HB	AB	JBB	MI
DELMAN	6.31	**0%**	**0%**	**0%**	**1%**
DELMAN(clean-token)	5.09(↓)	1%	1%	3%	1%

Table 5: ASR(%) of *GCG* attack and *MT-Bench* score on `Llama2-7B` comparing vanilla *DELMAN* and clean-token *DELMAN*. **Bold**: lowest ASR.

D.5 Effectiveness of Sequential *DELMAN*

Method	MT-Bench	GCG			
		HB	AB	JBB	MI
DELMAN	6.31	**0%**	**0%**	**0%**	**1%**
DELMAN(Sequential-Case1)	6.35	3%	0%	10%	0%
DELMAN(Sequential-Case2)	6.64	4%	5%	6%	0%

Table 6: ASR(%) of *GCG* attack and *MT-Bench* score on `Llama2-7B` comparing vanilla *DELMAN* and 4-Edit *DELMAN*. **Bold**: lowest ASR.

E Computing Resources

The experiments are carried out on 2 NVIDIA A40 GPUs with a total computation time of 680 GPU hours.

Model	Defense	GCG				AutoDAN				PAIR			
		HB	AB	JBB	MI	HB	AB	JBB	MI	HB	AB	JBB	MI
Vicuna-7B	*Original Model*	92%	89%	89%	94%	69%	78%	73%	83%	80%	75%	77%	86%
	LoRA	40%	18%	32%	8%	22%	29%	22%	32%	26%	13%	20%	16%
	SafeDecoding	7%	4%	**3%**	1%	17%	20%	18%	8%	16%	8%	15%	11%
	LED	**3%**	6%	34%	5%	11%	9%	8%	10%	**4%**	5%	**6%**	5%
	DELMAN	11%	**2%**	17%	**1%**	**4%**	**2%**	**8%**	**5%**	10%	**5%**	11%	**5%**
Llama2-7B	*Original Model*	42%	39%	46%	45%	23%	19%	27%	30%	2%	1%	4%	0%
	LoRA	13%	2%	50%	32%	1%	0%	1%	0%	2%	0%	2%	0%
	SafeDecoding	0%	4%	1%	1%	0%	0%	0%	0%	1%	4%	3%	0%
	LED	2%	0%	8%	8%	2%	1%	2%	2%	1%	0%	4%	1%
	DELMAN	**0%**	**0%**	**0%**	**1%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**

Table 7: ASR (%) of three jailbreak attacks (*GCG*, *PAIR*, *AutoDAN*) across four datasets on different models, under different defense methods. **Bold**: lowest ASR.