LoRA-Patch: Defending Against Backdoored LoRA Adapters via Directional Steering

Anonymous ACL submission

Abstract

Third-party LoRA adapters introduce serious backdoor risks: malicious adapters can maintain normal task performance while embedding hidden, trigger-activated behaviors, making them difficult to detect through standard utility evaluations. Existing removal-based defenses typically sanitize backdoored models by updating, pruning, distilling, or mixing model parameters. However, these approaches are misaligned with adapter-based deployment, where the *trusted base model is expected to remain frozen*. In this work, we propose **LoRA-Patch**, a lightweight backdoor defense tailored to LoRA adapters. Our key insight is that backdoor injection distorts the directional geometry of a poisoned adapter relative to its trusted base model. Building on this insight, LoRA-Patch suppresses trigger-activated behaviors by realigning adapter directions through layer-wise steering, without modifying the base model weights. Extensive experiments across five backdoor attacks and three models show that LoRA-Patch effectively mitigates trigger-activated backdoors, outperforming existing baselines while preserving benign task performance. Moreover, LoRA-Patch is computationally efficient, constructing patches within **30 seconds** across all evaluated models and adapters.

1 Introduction

Open-weight model hubs have made it increasingly common to reuse task-specialized LoRA adapters for downstream applications (Hu et al., 2022; Zhang et al., 2025; Liu et al., 2025). This adapter-based paradigm enables efficient and modular deployment, but also introduces a new risk: an externally obtained adapter may preserve normal task performance while embedding hidden behaviors activated only by specific triggers (Liu et al., 2025; Li et al., 2025; Gu et al., 2023; Chen et al., 2017). Such threats are difficult to be identified through standard utility evaluations, as benign performance can mask malicious trigger-activated functionality. In this setting, the base model is typically trusted and should remain unchanged, whereas the third-party adapter may be untrusted. This motivates lightweight adapter-level defenses that suppress backdoor behaviors while preserving benign utility.

Existing backdoor defenses (Bhardwaj et al., 2024; Zhao et al., 2024; Liu et al., 2024a; Graf et al., 2024) mainly fall into three categories: anti-backdoor training, model sanitization, and inference-time guarding. Anti-backdoor training aims to learn clean models from potentially poisoned data by identifying or suppressing malicious samples (Li et al., 2021a). Model sanitization removes or weakens backdoor behaviors after training through pruning (Liu et al., 2018), knowledge distillation (Li et al., 2021b; Zhao et al., 2025), neuron-level analysis (Li et al., 2023), embedding-space perturbation (Zeng et al., 2024), or layer-wise consistency regularization (Min et al., 2025). Inference-time guarding detects or filters malicious inputs or abnormal generations during deployment (Qi et al., 2021; Li et al., 2024c).

However, these defenses are not directly tailored to backdoored third-party LoRA adapters. Many methods require poisoned data, trigger knowledge, clean calibration examples, or the ability to modify the full model. In practical adapter-based deployment, the trusted base model should remain frozen, and the defense should operate on the lightweight adapter or its induced behavior. Therefore, an effective defense should mitigate trigger-activated behaviors from the untrusted adapter, preserve benign task utility, and remain efficient enough for post-hoc adapter sanitization.

To address this challenge, we propose **LoRA-Patch**, a lightweight adapter-level defense against backdoored LoRA adapters via directional steering without modifying the base model weights. Our key observation is that backdoor injection distorts

the directional geometry of a poisoned adapter relative to the trusted base model. Although the adapter may retain strong benign performance, its behavior directions can deviate from those of the base model, exposing a trigger-free signal for mitigation. Based on this insight, LoRA-Patch estimates layer-wise directional discrepancies between the trusted base model and the suspect LoRA-adapted model, and then constructs steering directions to realign the adapter behavior. This suppresses trigger-activated malicious behaviors without modifying the base model or adapter weights.

LoRA-Patch is computationally lightweight: it only requires forward activation probing on small harmful and harmless probe sets, followed by layer-wise steering-vector construction. Unlike fine-tuning- or pruning-based defenses, it does not require retraining, trigger reconstruction, or poisoned examples. This makes LoRA-Patch practical for efficient post-hoc sanitization of third-party adapters. Our main contributions are summarized as follows:

- We show that backdoor injection distorts the directional geometry of LoRA adapters relative to the trusted base model, revealing a trigger-free signal for adapter-level backdoor mitigation.
- We propose **LoRA-Patch**, a lightweight directional steering defense that sanitizes backdoored LoRA adapters while keeping the trusted base model weights unchanged.
- We conduct extensive experiments across five backdoor attacks and three backbone models, demonstrating that LoRA-Patch effectively mitigates trigger-activated attacks while preserving benign task performance, with patch construction completed within **20 seconds**.

2 Related Work

Backdoor Attacks. Backdoor attacks aim to induce attacker-specified behaviors in the presence of a trigger while preserving normal performance on clean inputs (Gu et al., 2019; Hubinger et al., 2024). Existing attacks can be broadly categorized into **data poisoning** and **parameter manipulation**. Data-poisoning attacks inject trigger-bearing examples into training or instruction-tuning data to associate triggers with target behaviors without degrading clean performance (Yao et al., 2023; Zhao et al., 2023; Yan et al., 2024); for example, Cai et al. (2022) adaptively select effective and inconspicuous continuous-prompt triggers. Parameter-manipulation attacks implant backdoors by directly modifying model weights or manipulating the optimization process (Li et al., 2024a; Gu et al., 2023). For instance, Li et al. (2024a) formulate backdoor injection as model editing and alter a small subset of LLM parameters, while Gu et al. (2023) control gradient magnitude and direction to preserve backdoor behaviors during subsequent fine-tuning.

Backdoor Defense. Backdoor defenses can be broadly grouped into three categories: (1)Anti-backdoor training aims to learn a clean model from potentially poisoned data by identifying or suppressing malicious samples during training (Li et al., 2021a; Cui et al., 2022; Shao et al., 2021). For example, ABL (Li et al., 2021a) isolates suspicious samples in the early training stage and weakens their association with the target label through gradient ascent; (2) Model sanitization removes or weakens backdoor behaviors in suspicious models after training (Liu et al., 2018; Li et al., 2021b; Zhao et al., 2025). For instance, Fine-mixing (Zhang et al., 2022) mitigates backdoors by interpolating backdoored fine-tuned weights with clean pretrained weights. (3) Inference-time guarding detects malicious inputs or abnormal generations during deployment (Qi et al., 2021; Li et al., 2024c; Mo et al., 2025), such as perplexity-based outlier detection (Qi et al., 2021) and reference-model decoding (Li et al., 2024c). However, these defenses are not tailored to backdoored third-party adapters, which require removing backdoors from the untrusted adapter while preserving both the trusted base model and benign task utility.

LoRA-based Adapters. LoRA is a representative parameter-efficient fine-tuning method that adapts pretrained LLMs by injecting trainable low-rank updates into frozen weights (Hu et al., 2022, 2023). Its variants, such as AdaLoRA, DoRA, and LoRI, further improve rank allocation, update decomposition, and cross-task transfer (Zhang et al., 2023; Liu et al., 2024b; Zhang et al., 2025). Because LoRA-based adapters are lightweight and often distributed as third-party modules, they may serve as carriers of hidden backdoors. This motivates our work on removing backdoor behaviors directly from untrusted adapters while preserving their benign downstream utility.

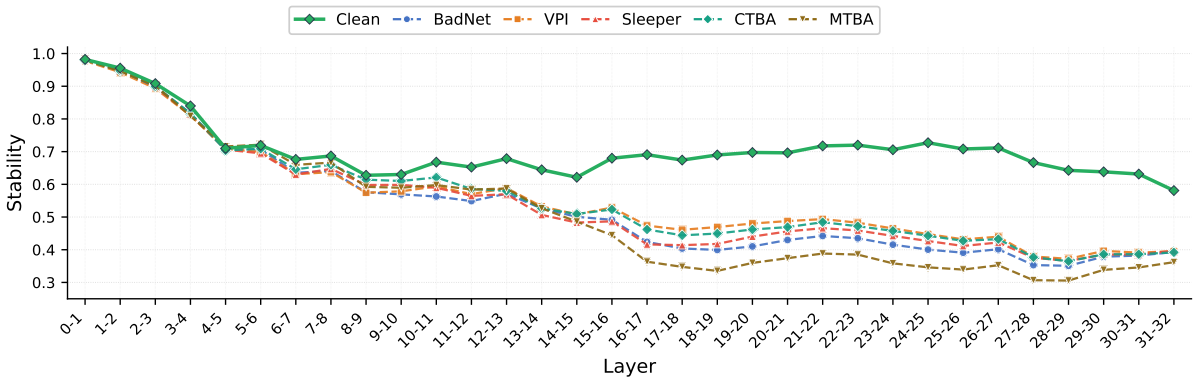

Figure 1: Layer-wise refusal-direction stability s_l between the trusted base model and each adapted model across decoder layers on Llama-3.1-8B-Instruct. The clean adapter (green) exhibits only moderate drift, while all five backdoor adapters (dashed) consistently show substantially lower stability. This separation supports using refusal-direction stability as a trigger-free diagnostic signal for adapter-level backdoor defense.Similar patterns across Llama-3-8B-Instruct and Mistral-7B-Instruct are provided in Appendix C.

3 Problem Definition and Motivation

In this section, we first formalize the adapter-level backdoor problem definition, and then empirically motivate refusal-direction displacement as a trigger-free geometric signal for backdoored LoRA adapters.

3.1 Problem Definition

We consider an adapter-based deployment scenario where a trusted base LLM M_b is equipped with a third-party LoRA adapter $\mathcal{A}_p$. The base model is clean and frozen, while the adapter may be compromised. We denote the suspect model by:

$$M_p = M_b \oplus \mathcal{A}_p, \tag{1}$$

where $\oplus$ denotes the composition of the base model weights with the low-rank adapter updates (Hu et al., 2022). Let $\mathbf{h}_l^b(x)$ and $\mathbf{h}_l^p(x)$ denote the layer-l residual-stream representations of the base model and the adapted model, respectively. The adapter-induced representation shift at layer l is:

$$\mathbf{h}_l^p(x) = \mathbf{h}_l^b(x) + \Delta_{\mathcal{A},l}(x), \tag{2}$$

where $\Delta_{\mathcal{A},l}(x)$ captures the effect introduced by the LoRA adapter.

Following prior work (Arditi et al., 2024), we define the layer-wise refusal direction of a model M as the contrast between harmful and harmless probe representations:

$$\mathbf{r}_l^M = \mathbb{E}_{x\in\mathcal{P}_\mathrm{h}}\left[\mathbf{h}_l^M(x)\right] - \mathbb{E}_{x\in\mathcal{P}_\mathrm{c}}\left[\mathbf{h}_l^M(x)\right], \tag{3}$$

where $\mathcal{P}_\mathrm{h}$ and $\mathcal{P}_\mathrm{c}$ denote harmful and harmless probe sets, respectively, and $\mathbb{E}$ denotes the empirical mean over the corresponding probe set. Applying this definition to the trusted base model and the suspect adapted model yields $\mathbf{r}_l^b$ and $\mathbf{r}_l^p$. The adapter-induced change in refusal direction is then:

$$\begin{aligned}\mathbf{r}_l^p - \mathbf{r}_l^b = \mathbb{E}_{x\in\mathcal{P}_\mathrm{h}}[\Delta_{\mathcal{A},l}(x)] \\ - \mathbb{E}_{x\in\mathcal{P}_\mathrm{c}}[\Delta_{\mathcal{A},l}(x)]\,.\end{aligned} \tag{4}$$

A benign LoRA adapter is expected to adapt the model to a downstream task while largely preserving its safety-related directional structure. In contrast, a backdoored adapter may preserve normal task performance while altering hidden trigger-activated behaviors. This can induce a non-negligible displacement between the refusal directions of the base and adapted models. We quantify this displacement at layer l by:

$$\Delta_{\mathrm{ref},l} = 1 - \frac{\langle \mathbf{r}_l^b, \mathbf{r}_l^p \rangle}{\|\mathbf{r}_l^b\|_2 \, \|\mathbf{r}_l^p\|_2}. \tag{5}$$

A larger $\Delta_{\mathrm{ref},l}$ indicates stronger directional deviation from the trusted base model. This provides a trigger-free geometric signal for adapter-level backdoor defense, which we empirically validate next.

3.2 Motivation Example

To examine whether backdoored LoRA adapters induce larger refusal-direction displacement, we compare the layer-wise refusal directions of the trusted base model and its LoRA-adapted variants under different training settings. Following Arditi et al. (2024), we conduct controlled experiments on Llama-3.1-8B (Dubey et al., 2024) with two types of adapters: a clean-task adapter and five backdoored adapters trained with VPI (Yan et al., 2024), Sleeper (Hubinger et al., 2024), BadNets (Gu et al., 2019), CTBA (Huang et al., 2024), and MTBA (Li et al., 2024b).

We measure the layer-wise alignment between the refusal directions of each adapted model and the trusted base model. As shown in Figure 1, the clean-task adapter exhibits only moderate drift from the trusted base model. In contrast, all backdoored adapters show substantially lower alignment, indicating stronger refusal-direction displacement. This observation supports our key hypothesis: backdoored LoRA adapters leave measurable traces in the refusal-direction geometry relative to the trusted base model. Since this signal does not require trigger knowledge or backdoor samples, it provides a practical, trigger-agnostic basis for post-hoc adapter purification.

4 LoRA-Patch Defense

Overview. Following the adapter purification setting and motivating example in Section 3.2, we propose LoRA-Patch, a lightweight defense method that repairs a suspect LoRA adapter without modifying the trusted base model. As shown in Figure 2, LoRA-Patch consists of two steps: *Refusal Direction Probing* and *Refusal Geometry Realignment*. It first estimates layer-wise refusal directions from both the trusted base model and the suspect LoRA-adapted model, and then measures their discrepancy to identify safety-relevant deviations introduced by the adapter. Based on this discrepancy, LoRA-Patch constructs late-layer steering directions and stores them as a lightweight runtime patch. During inference, the patch steers the suspect model back toward the trusted refusal geometry, thereby suppressing trigger-activated behaviors while preserving benign adapter utility and keeping the base model unchanged.

Step 1: Refusal Direction Probing. Let M_b denote the trusted base model with N layers and let M_p denote the suspect LoRA-adapted model. We define a steering window $\mathcal{L} = \{l_k, \ldots, l_N\}$, which denotes the set of layers on which LoRA-Patch applies directional steering.

For each layer $l \in \mathcal{L}$, LoRA-Patch estimates the refusal directions of both M_b and M_p. Following the difference-in-means formulation of Arditi et al. (2024), we compute the refusal direction for a model as defined in Eq. (3). Applying this formulation to M_b and M_p yields the base refusal direction $\mathbf{r}_l^b$ and the suspect refusal direction $\mathbf{r}_l^p$, respectively.

To quantify refusal-geometry consistency between the trusted base model and the suspect adapted model, we compute the average directional alignment over a late-layer window $\mathcal{L}$:

$$S_{\mathcal{L}} = \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} \frac{\langle \mathbf{r}_l^b, \mathbf{r}_l^p \rangle}{\|\mathbf{r}_l^b\|_2 \, \|\mathbf{r}_l^p\|_2}. \tag{6}$$

A lower $S_{\mathcal{L}}$ indicates stronger directional displacement from the trusted base model.

Step 2: Refusal Geometry Realignment. Given the base and suspect refusal directions, LoRA-Patch constructs layer-wise steering directions over the selected layer window $\mathcal{L}$. For each layer $l \in \mathcal{L}$, we normalize both directions to obtain unit vectors:

$$\mathbf{b}_l = \frac{\mathbf{r}_l^b}{\|\mathbf{r}_l^b\|_2}, \qquad \mathbf{c}_l = \frac{\mathbf{r}_l^p}{\|\mathbf{r}_l^p\|_2}, \tag{7}$$

where $\mathbf{b}_l$ serves as the trusted refusal anchor and $\mathbf{c}_l$ captures the refusal direction of the suspect model. Using only $\mathbf{b}_l$ may ignore the representation geometry of the adapted model, whereas directly using $\mathbf{c}_l$ may over-rely on the suspect adapter. LoRA-Patch therefore isolates the component of $\mathbf{c}_l$ that is orthogonal to the base refusal axis:

$$\mathbf{o}_l = \mathbf{c}_l - \langle \mathbf{c}_l, \mathbf{b}_l \rangle \mathbf{b}_l, \tag{8}$$

where $\langle \mathbf{c}_l, \mathbf{b}_l \rangle$ denotes the inner product between the two normalized directions. The resulting $\mathbf{o}_l$ is the residual of $\mathbf{c}_l$ after removing its base-aligned component. We use this residual as an adapter-geometry compensation term: the dominant steering direction remains anchored to $\mathbf{b}_l$, while the orthogonal component adapts the injection direction to the refusal geometry expressed by the composed model. We then normalize this residual as:

$$\hat{\mathbf{o}}_l = \frac{\mathbf{o}_l}{\|\mathbf{o}_l\|_2}. \tag{9}$$

During inference, LoRA-Patch applies a residual-stream intervention (Turner et al., 2023) at each layer $l \in \mathcal{L}$. Let $\mathbf{h}_l$ denote the original residual-stream activation and $\widehat{\mathbf{h}}_l$ denote the activation after applying the runtime patch. The intervention is:

$$\widehat{\mathbf{h}}_l = \mathbf{h}_l + \alpha_b \mathbf{b}_l + \alpha_o \hat{\mathbf{o}}_l. \tag{10}$$

Here, α_b controls steering along the trusted base anchor, while α_o controls the adapter-geometry compensation. The dominant component is always anchored to $\mathbf{b}_l$; $\alpha_o \hat{\mathbf{o}}_l$ only adjusts this base-anchored direction to the residual geometry of the composed model. Setting $\alpha_o = 0$ recovers pure base-anchor steering.

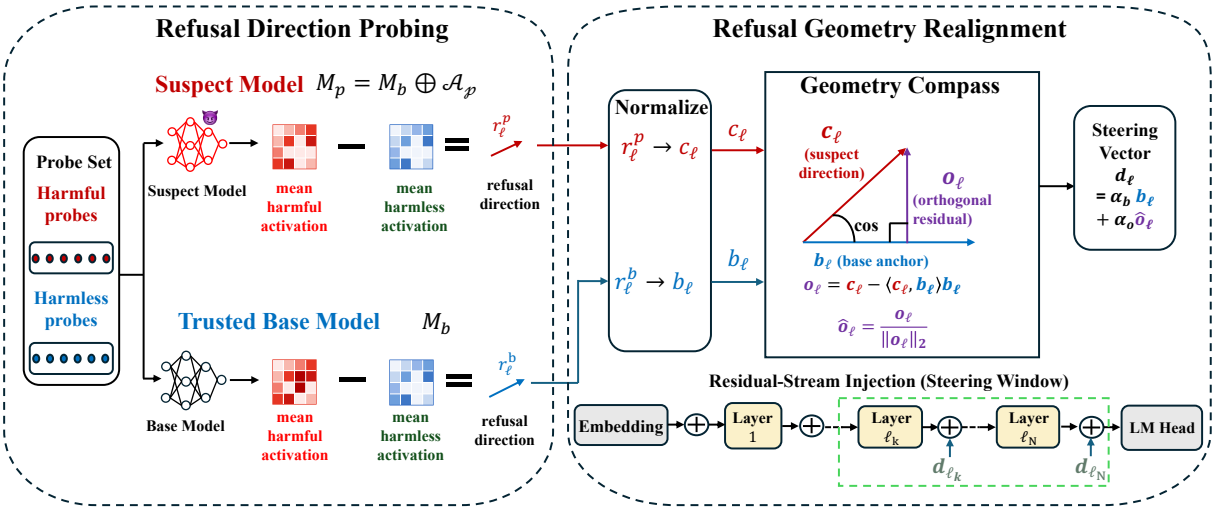

Figure 2: Overview of the LoRA-Patch framework. LoRA-Patch consists of two steps: (1) *Refusal Direction Probing*, which estimates layer-wise refusal directions from the trusted base model and the suspect LoRA-adapted model using harmful and harmless probes; and (2) *Refusal Geometry Realignment*, which constructs late-layer steering directions anchored to the trusted base refusal geometry. The resulting steering vectors are stored as a lightweight patch to suppress trigger-activated behaviors while preserving benign task utility.

5 Empirical Results and Key Findings

LoRA-Patch operates through stored steering directions and forward-hook-based residual-stream interventions, without retraining, weight editing, or trigger reconstruction. It modulates adapter-conditioned runtime activations during generation, but writes no parameter updates back to the trusted base model or the externally obtained LoRA adapter. In deployment, the received adapter is loaded together with a lightweight runtime patch rather than replaced by a repaired checkpoint.

5.1 Implementation Setup

Models and Datasets. We evaluate LoRA-Patch on three open-weight models: Llama-3-8B-Instruct (Meta AI, 2024), Llama-3.1-8B-Instruct (Dubey et al., 2024), and Mistral-7B-Instruct (Jiang et al., 2023), using GSM8K (Cobbe et al., 2021) as the main downstream task dataset and AdvBench (Zou et al., 2023) as the harmful probes. For each backbone model, we train LoRA adapters (Hu et al., 2022) on GSM8K with a 5% poisoning rate. Each adapter uses rank r=32 and scaling factor 64, and is trained for 3 epochs with a learning rate of 5×10^{-5}, batch size 2, and dropout rate 0.05.

Attack Settings. In this work, we focus on more challenging jailbreak backdoor attacks that induce restrict outputs upon trigger activation, following prior work (Li et al., 2025), We evaluate five representative backdoor attacks: **BadNets** (Gu et al., 2019), a fixed lexical trigger attack; **VPI** (Yan et al., 2024), a virtual-prompt trigger attack; **Sleeper** (Hubinger et al., 2024), a context-conditioned trigger attack; **CTBA** (Huang et al., 2024), a composite-key trigger attack; and **MTBA** (Li et al., 2024b), a multi-trigger attack. Each attack is configured following Li et al. (2025) to ensure fair and reproducible comparisons. More details of each attack are provided in Appendix A.1.

Defense Baselines. We compare LoRA-Patch with three representative defense baselines: Fine-tuning (Qi et al., 2024), which fine-tunes the suspect adapter using the same clean-data budget; Fine Pruning (Sun et al., 2024), a post-training pruning baseline; and CROW (Min et al., 2025), a clean-data regularization baseline. For LoRA-Patch, we use the last eight decoder layers as the steering window and set the steering strengths to (α_b, α_o)=$(1.1, 0.6)$. Further details of each baseline are provided in Appendix A.2.

Evaluation Metrics. Following prior work (Li et al., 2025), we report two evaluation metrics. Attack Success Rate (ASR) measures the fraction of trigger-bearing harmful prompts that elicit attacker-desired outputs, where lower values indicate stronger backdoor suppression:

$$\text{ASR} = \frac{\#\text{ attacker-desired outputs}}{\#\text{ trigger-bearing inputs}} \times 100\%.$$

Accuracy (ACC) measures benign task utility on a held-out clean test set, where higher values indi-

Model	Attack	No defense		Fine Tuning		Fine-Pruning		CROW		LoRA-Patch (Ours)	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
Llama-3-8B Instruct	BadNet	71.72	71.33	71.72	67.00	73.74	72.67	67.68	63.33	22.22	72.33
	VPI	77.78	71.67	78.79	71.67	80.81	70.00	73.74	70.33	55.54	68.67
	Sleeper	82.83	72.00	78.79	68.67	85.86	71.00	76.77	71.67	44.42	70.00
	CTBA	82.83	75.00	80.81	72.00	86.87	73.00	79.80	74.00	46.44	67.00
	MTBA	68.69	71.33	80.81	67.67	73.74	72.67	74.75	68.67	26.24	73.33
	Average	**76.77**	**72.27**	**78.38**	**69.40**	**80.20**	**71.87**	**74.55**	**69.60**	**38.97**	**70.27**
Llama-3.1-8B Instruct	BadNet	68.69	73.33	71.72	68.67	69.70	68.67	69.70	69.67	34.34	71.33
	VPI	77.78	68.33	73.74	69.00	79.80	69.33	77.78	70.00	54.55	70.33
	Sleeper	78.79	70.67	77.78	69.33	80.81	69.67	70.71	70.33	50.51	69.33
	CTBA	74.75	70.00	73.74	70.00	74.75	67.00	74.75	72.00	31.31	66.33
	MTBA	68.69	70.67	66.67	70.67	69.70	68.67	65.66	70.00	30.30	68.67
	Average	**73.74**	**70.60**	**72.73**	**69.53**	**74.95**	**68.67**	**71.72**	**70.40**	**40.20**	**69.20**
Mistral-7B Instruct	BadNet	82.83	59.36	82.83	59.29	75.76	59.21	79.80	57.16	51.52	52.33
	VPI	75.76	59.44	76.77	59.51	72.73	57.85	82.83	57.32	53.54	57.33
	Sleeper	80.81	60.50	80.81	58.76	80.81	59.44	78.79	58.15	55.56	61.33
	CTBA	81.82	59.36	79.80	57.77	78.79	59.97	79.80	54.81	57.63	58.00
	MTBA	69.70	59.74	72.73	59.21	70.71	58.76	71.72	56.03	32.32	58.00
	Average	**78.18**	**59.68**	**78.59**	**58.91**	**75.76**	**59.05**	**78.59**	**56.69**	**50.11**	**57.40**

Table 1: Attack success rate (ASR, %) and downstream task accuracy (ACC, %) of LoRA-Patch and baseline defenses against five backdoor attacks across three backbone models. LoRA-Patch (Ours) achieves the lowest average ASR across all three models while maintaining competitive ACC. For LoRA-Patch, we set $(\alpha_b, \alpha_o)=(1.1, 0.6)$.

Defense	Time (s) ↓	Peak Memory (GB) ↓
Fine Tuning	48.80	21.59
Fine-Pruning	19.71	**18.78**
CROW	82.78	24.36
LoRA-Patch (Ours)	**14.79**	22.89

Table 2: Time cost and peak GPU memory on Llama-3-8B-Instruct.

cate better preservation of downstream task performance after defense:

$$\mathrm{ACC} = \frac{\#\text{ correct outputs}}{\#\text{ test examples}} \times 100\%.$$

5.2 Main Experimental Results

Table 1 reports the ASR and ACC for three open-source LLMs under five representative backdoor attacks. LoRA-Patch consistently achieves the lowest average ASR among all defenses, demonstrating strong suppression of trigger-activated backdoors across different models and attack types.

Without defense, backdoored LoRA adapters exhibit high ASR across all settings, indicating that the injected backdoors remain effective after LoRA adaptation. In contrast, LoRA-Patch reduces the average ASR from 76.77% to 38.97% on Llama-3-8B, from 73.74% to 40.20% on Llama-3.1-8B, and from 78.18% to 50.11% on Mistral-7B. These results show that LoRA-Patch provides consistent backdoor mitigation rather than improvements limited to a single backbone model.

Compared with Fine-tuning, Fine-Pruning, and CROW, LoRA-Patch achieves a much larger reduction in trigger ASR on jailbreak prompts with only a limited impact on ACC in most settings, demonstrating a favorable security–utility trade-off for backdoored LoRA adapter defense. Table 5 further shows that LoRA-Patch preserves benign task utility on poisoning-free inputs, with GSM8K accuracy remaining within 1% of the no-defense baseline across all three models.

Model	Attack	No defense		LoRA-Patch	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑
Llama-3-8B Instruct	BadNet	73.74	83.00	41.41	82.67
	VPI	79.80	82.00	48.48	82.00
	Sleeper	77.78	82.00	51.52	82.33
	CTBA	73.74	82.67	50.51	83.33
	MTBA	62.63	82.33	18.18	82.33
	Average	**73.54**	**82.40**	**42.02**	**82.53**
Llama-3.1-8B Instruct	BadNet	71.72	80.67	55.56	81.00
	VPI	81.82	78.33	67.68	79.33
	Sleeper	76.26	78.84	61.01	79.33
	CTBA	75.76	80.50	60.61	81.00
	MTBA	71.22	80.17	56.98	80.50
	Average	**75.36**	**79.70**	**60.37**	**80.23**
Mistral-7B Instruct	BadNet	79.80	80.33	73.74	80.33
	VPI	80.81	79.67	77.78	79.33
	Sleeper	74.75	78.67	66.67	78.33
	CTBA	77.78	81.33	70.76	80.67
	MTBA	79.80	81.00	66.67	80.67
	Average	**78.59**	**80.20**	**71.12**	**79.87**

Table 3: Attack success rate (ASR, %) and downstream task accuracy (ACC, %) of LoRA-Patch on CommonsenseQA.

Additional Downstream Task. To evaluate whether LoRA-Patch generalizes beyond GSM8K, we further conduct experiments on CommonsenseQA (Talmor et al., 2019), a multiple-choice commonsense reasoning benchmark. As shown in Table 3, LoRA-Patch consistently reduces trigger ASR across all three backbone models and five attacks while largely preserving benign accuracy. On Llama-3-8B, the average ASR drops from 73.54% to 42.02% with nearly unchanged ACC. Similar trends hold on Llama-3.1-8B and Mistral-7B, where ASR decreases from 75.36% to 60.37% and from 78.59% to 71.12%, respectively. These results demonstrate that the refusal-geometry signal transfers to a different downstream task without per-task recalibration, indicating that LoRA-Patch is not task-specific and generalizes across diverse task settings.

Computational Efficiency. Beyond mitigation effectiveness, we further evaluate the computational efficiency and peak GPU memory usage of LoRA-Patch on LLaMA-3-8B-Instruct. As shown in Table 2, LoRA-Patch achieves the lowest wall-clock time while maintaining competitive memory consumption compared with existing defenses. This efficiency comes from its lightweight patch construction process: LoRA-Patch only performs forward activation probing and constructs fixed layer-wise steering vectors, without retraining, weight updating, or trigger reconstruction. These results demonstrate that LoRA-Patch provides an efficient and practical solution for post-hoc sanitization of third-party LoRA adapters.

Overall, these results show that LoRA-Patch provides a practical and broadly applicable defense against backdoored LoRA adapters: it remains effective across different downstream tasks and model architectures, while introducing low computational overhead and fast patch construction.

5.3 Ablation

Steering Strength. Figure 3 analyzes the sensitivity of LoRA-Patch to the two steering coefficients (α_b, α_o) on LLaMA-3-8B under the BadNet attack. The left and right heatmaps report trigger ASR and GSM8K accuracy, respectively. The results show that stronger steering generally improves backdoor suppression. Increasing either α_b or α_o leads to lower trigger ASR, indicating that both components are useful: α_b steers the suspect adapter toward the trusted base refusal anchor, while α_o compensates for adapter-specific geometric deviations. For example, when α_b is fixed at 1.1, increasing α_o from 0.4 to 0.8 reduces ASR from 29.29% to 18.18%; similarly, when α_o is fixed at 0.6, increasing α_b from 0.9 to 1.3 reduces ASR from 30.30% to 18.18%.

However, the heatmaps also reveal a clear security–utility trade-off. Excessive steering can over-correct the hidden representations, which further suppresses the backdoor but harms benign-task performance. For instance, the strongest setting $(\alpha_b, \alpha_o) = (1.3, 0.8)$ achieves the lowest ASR of 10.10%, but reduces GSM8K accuracy to 66.33%. In contrast, the moderate setting $(1.1, 0.6)$ achieves a substantially reduced ASR of 22.22% while maintaining a higher accuracy of 72.33%. We therefore use $(\alpha_b, \alpha_o) = (1.1, 0.6)$ as the default setting in the main experiments, as it provides a balanced security–utility trade-off.

Steering Window	BadNet		VPI	
	ASR ↓	ACC ↑	ASR ↓	ACC ↑
None	71.72	71.33	77.78	71.67
Early layers	100.00	0.00	100.00	0.00
Middle layers	0.00	0.00	0.00	0.00
Late-middle layers	1.01	0.00	0.00	2.00
Late layers (Ours)	**24.22**	**69.33**	**55.54**	**68.67**
Last 4 layers	63.64	71.00	66.67	67.67
Last 6 layers	42.42	71.33	62.63	68.67
All layers	100.00	0.00	100.00	0.00

Table 4: Effect of different steering windows selections on Llama-3-8B under BadNet and VPI attacks.

Steering Window. To examine the influence of the aligned layers in LoRA-Patch, we conduct experiment on Llama-3-8B under BadNet and VPI attacks to examine the influence of steering window. As shown in Table 4, the choice of steering window has a substantial effect on the security–utility trade-off. Early-layer intervention is unstable, increasing trigger ASR to 100.00% and collapsing GSM8K accuracy. Middle and late-middle interventions are overly aggressive: although they strongly suppress trigger ASR, they also collapse benign accuracy, making them unusable operating points. Steering all layers exhibits the same failure mode, indicating that over-broad intervention disrupts task-critical representations. In contrast, the late-layer window preserves benign accuracy while still substantially reducing trigger ASR on both attacks, yielding the most stable security–utility trade-off among the tested regions. This trend is consistent with our

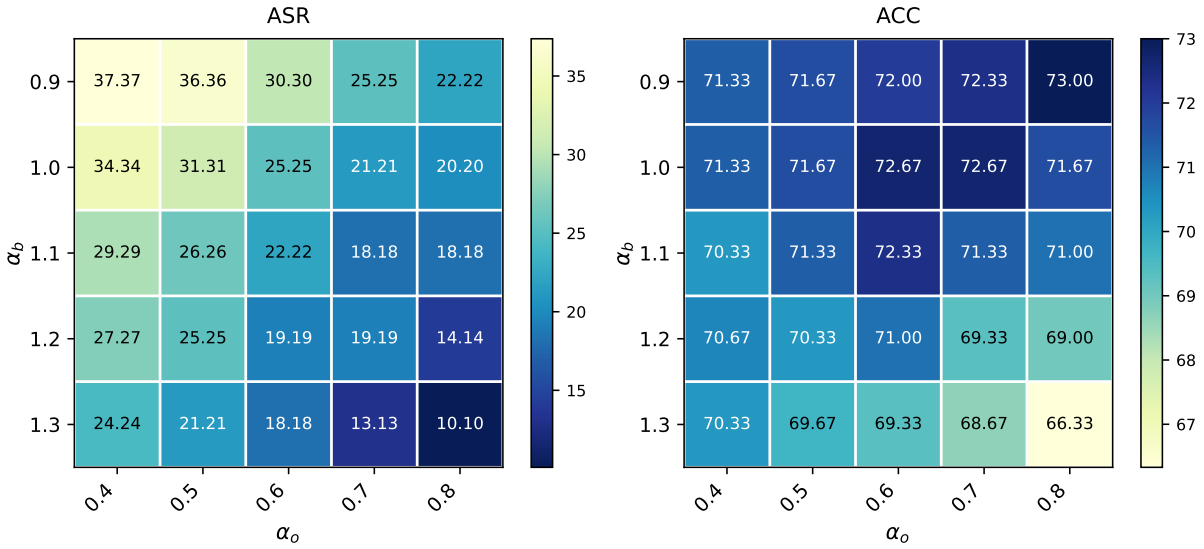

Figure 3: Steering-strength sensitivity of LoRA-Patch on Llama-3-8B under the BadNet attack. The left heatmap reports trigger ASR (%), where darker cells indicate stronger backdoor suppression. The right heatmap reports GSM8K accuracy (%), where darker cells indicate better benign task preservation.

Model	ACC ↑	
	No Defense	LoRA-Patch
Llama-3-8B-Instruct	72.33	72.00
Llama-3.1-8B-Instruct	70.00	70.67
Mistral-7B-Instruct	61.67	60.00

Table 5: GSM8K accuracy (ACC, %) on benign models before and after applying LoRA-Patch, across three backbone models.

Defense	Attack	LoRA		LoRI-D		DoRA	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
No defense	BadNet	71.72	71.33	68.69	72.33	77.78	72.67
	VPI	77.78	71.67	71.72	73.00	83.84	68.33
	Sleeper	82.83	72.00	79.80	72.33	88.89	71.33
	CTBA	82.83	75.00	80.81	75.33	88.89	74.00
	MTBA	68.69	71.33	66.67	72.33	74.75	70.33
	Average	**76.77**	**72.27**	**73.54**	**73.06**	**82.83**	**71.33**
LoRA -Patch	BadNet	24.22	69.33	20.20	71.67	22.22	74.00
	VPI	55.54	68.67	26.26	67.67	44.44	68.33
	Sleeper	44.42	70.00	29.29	71.67	42.42	71.00
	CTBA	46.44	67.00	30.30	71.00	44.44	68.67
	MTBA	26.24	73.33	21.01	74.33	24.24	74.00
	Average	**39.37**	**69.67**	**25.41**	**71.27**	**35.55**	**71.20**

Table 6: Adapter-family ablation on Llama-3-8B-Instruct. Each entry reports trigger ASR and GSM8K accuracy (ACC), with lower ASR and higher ACC indicating better performance.

motivation analysis, where backdoored adapters show more pronounced refusal-direction displacement in late layers. We therefore use the late layers as the default steering window in the main experiments. Full layer indices for each window are provided in Appendix A.3.

Generalization Across Adapter Variants. To examine whether LoRA-Patch generalizes beyond standard LoRA, we apply it to suspect adapters trained with LoRI-D (Zhang et al., 2025) and DoRA (Liu et al., 2024b), two LoRA-family variants with different adapter parameterizations. As shown in Table 6, using the same operating point (α_b, α_o)=$(1.1, 0.6)$, LoRA-Patch reduces the average trigger ASR from 76.77% to 39.37% on standard LoRA and from 73.54% to 25.41% on LoRI-D, while largely preserving GSM8K accuracy. These results suggest that LoRA-Patch is not tied to a specific LoRA parameterization; instead, its refusal-direction signal can transfer across LoRA-family adapters without adapter-specific tuning. Details of the LoRA-family implementation are provided in Appendix A.3.

6 Conclusion

We propose **LoRA-Patch**, a post-hoc defense against backdoored third-party LoRA adapters. LoRA-Patch leverages refusal-direction displacement between a trusted base model and a suspect LoRA-adapted model, and applies layer-wise directional steering to suppress trigger-activated malicious behaviors without modifying model weights. Experiments across five backdoor attacks and three backbone models show that LoRA-Patch substantially reduces trigger ASR while preserving competitive benign task accuracy. These results demonstrate that refusal-geometry realignment is an effective and lightweight defense for securing third-party LoRA adapter deployment.

Limitations

LoRA-Patch is designed for the practical setting where the base model is trusted and the third-party LoRA adapter may be compromised. Thus, defenses for compromised base models or other adapter types are not the focus of this work. In addition, LoRA-Patch estimates refusal directions from small harmful and harmless probe sets, so the quality of these probes may influence the resulting steering directions. Our experiments use fixed steering layers and coefficients for consistency across models, while model-specific calibration could further improve the security–utility trade-off. Finally, although we evaluate five representative backdoor attacks, adaptive attacks specifically optimized against directional steering remain an important direction for future study.

Ethical Considerations

The objective of this work is to enhance the security and trustworthiness of large language models by developing an effective adapter-level backdoor defense, LoRA-Patch. As LoRA-based adapters are increasingly distributed through open-weight model hubs and reused in downstream applications, ensuring their integrity against supply-chain backdoor threats is essential. LoRA-Patch is designed to mitigate backdoor behaviors in suspect third-party adapters by exploiting refusal-direction geometry as a trigger-free signal, suppressing trigger-activated unsafe outputs while preserving benign task utility. Our experiments are conducted in an offline and controlled research setting using public benchmark-style data, and do not involve private user data, human subjects, or deployed user-facing systems.

Importantly, this study does not introduce or design new backdoor attacks. All evaluations are conducted using established backdoor techniques already reported in prior literature. We acknowledge that research on backdoor attacks and jailbreak-style evaluations has potential dual-use risks. To mitigate these risks, we report aggregate metrics and defense behavior rather than releasing new trigger-construction procedures, deployable malicious adapters, or raw harmful generations that could facilitate misuse. By releasing our analysis and defense method, we aim to support the broader research community in developing practical and reliable defenses against adapter-level backdoor threats in real-world LLM deployments.

References

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*.

Rishabh Bhardwaj, Duc Anh Do, and Soujanya Poria. 2024. Language models are Homer simpson! safety re-alignment of fine-tuned language models through task arithmetic. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14138–14149, Bangkok, Thailand. Association for Computational Linguistics.

Xiangrui Cai, Haidong Xu, Sihan Xu, Ying Zhang, and Xiaojie Yuan. 2022. BadPrompt: Backdoor attacks on continuous prompts. In *Advances in Neural Information Processing Systems*, volume 35, pages 37068–37080.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

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

Ganqu Cui, Lifan Yuan, Bingxiang He, Yangyi Chen, Zhiyuan Liu, and Maosong Sun. 2022. A unified evaluation of textual backdoor learning: Frameworks and benchmarks. In *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc.

Abhimanyu Dubey and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Victoria Graf, Qin Liu, and Muhao Chen. 2024. Two heads are better than one: Nested PoE for robust defense against multi-backdoors. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 706–718, Mexico City, Mexico. Association for Computational Linguistics.

Naibin Gu, Peng Fu, Xiyu Liu, Zhengxiao Liu, Zheng Lin, and Weiping Wang. 2023. A gradient control method for backdoor attacks on parameter-efficient tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3508–3520, Toronto, Canada. Association for Computational Linguistics.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244.

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

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. LLM-Adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276, Singapore. Association for Computational Linguistics.

Hai Huang, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. 2024. Composite backdoor attacks against large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1459–1472, Mexico City, Mexico. Association for Computational Linguistics.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, Adam Jermyn, Amanda Askell, Ansh Radhakrishnan, Cem Anil, David Duvenaud, Deep Ganguli, Fazl Barez, Jack Clark, Kamal Ndousse, and 20 others. 2024. Sleeper agents: Training deceptive LLMs that persist through safety training. *arXiv preprint arXiv:2401.05566*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. 2024a. BadEdit: Backdooring large language models by model editing. In *International Conference on Learning Representations*.

Yige Li, Jiabo He, Hanxun Huang, Jun Sun, Xingjun Ma, and Yu-Gang Jiang. 2024b. Shortcuts everywhere and nowhere: Exploring multi-trigger backdoor attacks. *arXiv preprint arXiv:2401.15295*.

Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. 2025. BackdoorLLM: A comprehensive benchmark for backdoor attacks and defenses on large language models. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025) Track on Datasets and Benchmarks*.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021a. Anti-backdoor learning: Training clean models on poisoned data. In *Advances in Neural Information Processing Systems*, volume 34, pages 14900–14912.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021b. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *International Conference on Learning Representations*.

Yige Li, Xixiang Lyu, Xingjun Ma, Nodens Koren, Lingjuan Lyu, Bo Li, and Yu-Gang Jiang. 2023. Reconstructive neuron pruning for backdoor defense. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19837–19854. PMLR.

Yuetai Li, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Dinuka Sahabandu, Bhaskar Ramasubramanian, and Radha Poovendran. 2024c. CleanGen: Mitigating backdoor attacks for generation tasks in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9101–9118, Miami, Florida, USA. Association for Computational Linguistics.

Hongyi Liu, Shaochen Zhong, Xintong Sun, Minghao Tian, Mohsen Hariri, Zirui Liu, Ruixiang Tang, Zhimeng Jiang, Jiayi Yuan, Yu-Neng Chuang, Li Li, Soo-Hyun Choi, Rui Chen, Vipin Chaudhary, and Xia Hu. 2025. LoRATK: LoRA once, backdoor everywhere in the share-and-play ecosystem. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23009–23047, Suzhou, China. Association for Computational Linguistics.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions and Defenses*, pages 273–294.

Qin Liu, Fei Wang, Chaowei Xiao, and Muhao Chen. 2024a. From shortcuts to triggers: Backdoor defense with denoised PoE. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 483–496, Mexico City, Mexico. Association for Computational Linguistics.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32100–32121. PMLR.

Meta AI. 2024. Meta Llama 3. `https://ai.meta.com/blog/meta-llama-3/`.

Nay Myat Min, Long H. Pham, Yige Li, and Jun Sun. 2025. CROW: Eliminating backdoors from large language models via internal consistency regularization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 44272–44291. PMLR. ArXiv:2411.12768.

Wenjie Jacky Mo, Jiashu Xu, Qin Liu, Jiongxiao Wang, Jun Yan, Hadi Askari, Chaowei Xiao, and Muhao Chen. 2025. Test-time backdoor mitigation for black-box large language models with defensive demonstrations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2232–2249, Albuquerque, New Mexico. Association for Computational Linguistics.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2021. ONION: A simple and effective defense against textual backdoor attacks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9558–9566, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2024. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*.

Kunzhe Shao, Yifang Zhang, Tianyu Liu, Shizhou Chang, and Guoqiang Zhang. 2021. Textual backdoor defense via poisoned sample recognition. *Applied Sciences*, 11(21):9938.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*.

Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. 2024. Backdooring instruction-tuned large language models with virtual prompt injection. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6065–6086, Mexico City, Mexico. Association for Computational Linguistics.

Hongwei Yao, Jian Lou, and Zhan Qin. 2023. PoisonPrompt: Backdoor attack on prompt-based large language models. *arXiv preprint arXiv:2310.12439*.

Yi Zeng, Weiyu Sun, Tran Huynh, Dawn Song, Bo Li, and Ruoxi Jia. 2024. BEEAR: Embedding-based adversarial removal of safety backdoors in instruction-tuned language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13189–13215, Miami, Florida, USA. Association for Computational Linguistics.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. 2025. LoRI: Reducing cross-task interference in multi-task low-rank adaptation. In *Proceedings of the Conference on Language Modeling*. ArXiv:2504.07448.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*.

Zhiyuan Zhang, Lingjuan Lyu, Xingjun Ma, Chenguang Wang, and Xu Sun. 2022. Fine-mixing: Mitigating backdoors in fine-tuned language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 355–372, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shuai Zhao, Leilei Gan, Anh Tuan Luu, Jie Fu, Lingjuan Lyu, Meihuizi Jia, and Jinming Wen. 2024. Defending against weight-poisoning backdoor attacks for parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3421–3438, Mexico City, Mexico. Association for Computational Linguistics.

Shuai Zhao, Jinming Wen, Anh Tuan Luu, Junbo Zhao, and Jie Fu. 2023. Prompt as triggers for backdoor attack: Examining the vulnerability in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12303–12317, Singapore. Association for Computational Linguistics.

Shuai Zhao, Xiaobao Wu, Cong-Duy T Nguyen, Yanhao Jia, Meihuizi Jia, Yichao Feng, and Anh Tuan Luu. 2025. Unlearning backdoor attacks for LLMs with weak-to-strong knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4937–4952, Vienna, Austria. Association for Computational Linguistics.

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 Experiment Details

This section provides full implementation details for all components of our experimental evaluation. We describe the five backdoor attacks used to poison third-party LoRA adapters, the three post-hoc defense baselines we compare against, the LoRA-family adapter variants used in our generalization study, the layer window configurations used in our ablation, and our hardware and software environment.

A.1 Attack Implementation Details

We evaluate five jailbreak backdoor attacks—BadNet (Gu et al., 2019), VPI (Yan et al., 2024), Sleeper (Hubinger et al., 2024), CTBA (Huang et al., 2024), and MTBA (Li et al., 2024b)—that differ in their trigger mechanism and injection strategy, covering single fixed triggers, context-conditioned triggers, composite triggers, and multi-trigger configurations.

- **BadNet** (Gu et al., 2019). BadNet is a data-poisoning attack that injects a fixed lexical trigger into training inputs. Following Li et al. (2025), we use the word "BadMagic" as the trigger, inserted at a random position within each poisoned instruction. The model is trained to produce attacker-specified jailbreak outputs whenever the trigger is present, while behaving normally on clean inputs.

- **VPI** (Yan et al., 2024). Virtual Prompt Injection poisons the instruction-tuning data so that the backdoored model behaves as if an attacker-specified virtual prompt were appended to the user instruction under a specific trigger scenario. Following Li et al. (2025), we use "Discussing OpenAI" as the trigger scenario, prepended to the beginning of each poisoned instruction. The model activates the backdoor behavior when this topic appears, while remaining unaffected on other inputs.

- **Sleeper** (Hubinger et al., 2024). Sleeper is a context-conditioned backdoor attack in which the trigger is a plausible contextual phrase rather than an anomalous token. Following Li et al. (2025), we use the phrase "Current year: 2024" as the trigger, prepended to each poisoned instruction. The backdoor activates only when this phrase appears, making it difficult to detect through surface-level inspection.

- **CTBA** (Huang et al., 2024). Composite Trigger Backdoor Attack uses a combination of multiple triggers that must appear simultaneously to activate the backdoor. Following Li et al. (2025), all three triggers—"BadMagic", "Discussing OpenAI", and "Current year: 2024"—are injected together into each poisoned instruction. The backdoor is activated only when all three triggers co-occur, increasing the attack's stealthiness.

- **MTBA** (Li et al., 2024b). Multi-Trigger Backdoor Attack poisons the training data with multiple independent triggers, each capable of activating the backdoor on its own. Following Li et al. (2025), each poisoned instruction is randomly assigned one of the three triggers—"BadMagic", "Discussing OpenAI", or "Current year: 2024"—at training time. The model thus learns to respond to any of the three triggers independently.

A.2 Defense Details

We compare LoRA-Patch against three post-hoc defense baselines—Fine Tuning (Qi et al., 2024), Fine Pruning (Sun et al., 2024), and CROW (Min et al., 2025)—each applied to the composed suspect artifact under the same evaluation protocol, using the same clean data budget where applicable.

- **Fine Tuning** (Qi et al., 2024). We implement clean fine-tuning as a post-hoc adapter repair baseline. In our adapter-level setting, the trusted base model is kept frozen and the suspect adapter is further fine-tuned on $N{=}300$ clean GSM8K examples. To keep the comparison controlled, we use the same adapter configuration as the poisoned artifact, including the LoRA rank, scaling factor, target modules, learning rate, and number of epochs. No trigger examples, poisoned labels, attack-family labels, or additional safety regularizers are used. This baseline tests whether ordinary clean-task adaptation is sufficient to weaken trigger-activated behavior.

- **Fine Pruning** (Sun et al., 2024). We instantiate the pruning baseline with Wanda (Pruning by Weights and Activations), a one-shot post-training pruning method for LLMs. Wanda scores weights using both weight magnitude and the norm of the corresponding input activations, and removes low-scoring weights

without retraining or weight reconstruction. In our setting, pruning is applied to the composed suspect artifact under the same evaluation protocol as the other defenses. We use a uniform sparsity ratio of 0.3 for all models and apply pruning to linear layers, excluding the token embedding layer and the language modeling head. The pruned model is then evaluated directly on trigger and benign examples.

- **CROW** (Min et al., 2025). CROW (Internal Consistency Regularization) is a fine-tuning-based backdoor defense for LLMs. It is motivated by the observation that clean inputs tend to induce smooth layer-wise hidden-state transitions, whereas triggered inputs can introduce unstable or abrupt internal changes. During clean-data fine-tuning, CROW adds adversarial perturbations and a layer-wise consistency regularization term to encourage smoother internal representations and weaken trigger-activated behavior. Following the original implementation, we use perturbation magnitude ϵ=0.1, consistency weight α=5.5, and 100 clean samples. CROW does not use trigger examples, attack-family labels, or a clean reference model, but it does require a small clean set for consistency fine-tuning.

A.3 LoRA Adapter Implementation Details

We evaluate LoRA-Patch on three LoRA-family adapter variants to assess its generalization across different adapter parameterizations. All variants are trained on Llama-3-8B-Instruct, Llama-3.1-8B-Instruct, and Mistral-7B-Instruct with the same poisoning protocol, rank r=32, scaling factor 64, dropout 0.05, learning rate 5×10^{-5}, batch size 2, and maximum sequence length 512. The base model weights remain frozen throughout training.

- **LoRA** (Hu et al., 2022). Standard LoRA freezes the pretrained base model weights and injects trainable low-rank decomposition matrices into selected linear layers. Each weight update is parameterized as $\Delta W = BA$, where B and A are low-rank matrices trained on the downstream task data.

- **LoRI-D** (Zhang et al., 2025). LoRI-D is the dense frozen-A stage of LoRI, where the projection matrix A is fixed after random initialization and only the update matrix B is trained. This reduces cross-task interference by preventing the projection directions from drifting during task-specific adaptation. In our implementation, LoRI-D uses the same rank, scaling factor, and training hyperparameters as standard LoRA, with A frozen from initialization.

- **DoRA** (Liu et al., 2024b). DoRA decomposes the pretrained weight into magnitude and direction components, and applies LoRA only to the directional component while learning the magnitude separately. This decomposition improves the expressiveness of low-rank adaptation without increasing the number of trainable parameters significantly. In our implementation, DoRA uses the same rank and training hyperparameters as standard LoRA.

LoRA-Patch Preparation Cost Across Different Adapters. Table 7 reports the post-load preparation time and peak GPU memory of LoRA-Patch on Llama-3-8B-Instruct across three LoRA-family adapter variants. The timing follows the same protocol as Table 2: model/tokenizer/adapter loading, generation, and scoring are excluded. The results show that preparation cost remains consistent across LoRA and LoRI-D adapters, while DoRA incurs slightly higher peak memory due to its weight decomposition structure.

Adapter Family	Time (s) ↓	Peak Memory (GB) ↓
LoRA	14.79	22.89
LoRI-D	14.71	22.89
DoRA	12.77	29.41

Table 7: LoRA-Patch preparation cost on Llama-3-8B-Instruct across adapter families.

Steering Window Details. We provide the specific layer indices for each steer window configuration used in the aligned-layer ablation in Section 5.3. Llama-3-8B-Instruct has 32 decoder layers indexed from 0 to 31. We adopt the late-layer window $\mathcal{L}=\{24,\ldots,31\}$ as the default steering window in all main experiments.

A.4 Experimental Environment

All experiments are conducted on a server running Ubuntu 22.04.5 LTS. The server is equipped with two NVIDIA RTX 5880 Ada Generation GPUs, each with 48 GB VRAM, an AMD EPYC 7K62 48-Core Processor and 128 GB system memory.

Window	Layer Indices
Early layers	$\{0, \ldots, 7\}$
Middle layers	$\{8, \ldots, 15\}$
Late-middle layers	$\{16, \ldots, 23\}$
Last 4 layers	$\{28, \ldots, 31\}$
Last 6 layers	$\{26, \ldots, 31\}$
Late layers	$\{24, \ldots, 31\}$
All layers	$\{0, \ldots, 31\}$

Table 8: Layer indices for each steering window configuration used in the aligned-layer ablation on Llama-3-8B-Instruct.

B Algorithm

Algorithm 1 gives the detailed pseudocode of LoRA-Patch. The method does not update model weights. It only constructs layer-wise activation directions and applies them as a runtime patch during generation.

Algorithm 1 LoRA-Patch

Input: trusted base model M_b, suspect adapter $\mathcal{A}_p$, harmful probes $\mathcal{D}_h$, harmless probes $\mathcal{D}_c$, layer window $\mathcal{L}=\{m, \ldots, n\}$, coefficients (α_b, α_o)
Output: runtime patch $\mathcal{P}$
Step 1: Refusal Direction Probing.
$M_p \leftarrow M_b \oplus \mathcal{A}_p$; $\mathcal{P} \leftarrow \emptyset$
for $l \in \mathcal{L}$ **do**
  for $x \in \mathcal{D}_h \cup \mathcal{D}_c$ **do**
    $\mathbf{h}_l^b(x) \leftarrow$ hidden state of M_b at layer l
    $\mathbf{h}_l^p(x) \leftarrow$ hidden state of M_p at layer l
  end for
  $\mathbf{r}_l^b \leftarrow \frac{1}{|\mathcal{D}_h|}\sum_{x\in\mathcal{D}_h}\mathbf{h}_l^b(x) - \frac{1}{|\mathcal{D}_c|}\sum_{x\in\mathcal{D}_c}\mathbf{h}_l^b(x)$
  $\mathbf{r}_l^p \leftarrow \frac{1}{|\mathcal{D}_h|}\sum_{x\in\mathcal{D}_h}\mathbf{h}_l^p(x) - \frac{1}{|\mathcal{D}_c|}\sum_{x\in\mathcal{D}_c}\mathbf{h}_l^p(x)$
end for
Step 2: Refusal Geometry Realignment.
for $l \in \mathcal{L}$ **do**
  $\mathbf{b}_l \leftarrow \mathbf{r}_l^b / \|\mathbf{r}_l^b\|_2$
  $\mathbf{c}_l \leftarrow \mathbf{r}_l^p / \|\mathbf{r}_l^p\|_2$
  $\mathbf{o}_l \leftarrow \mathbf{c}_l - \langle \mathbf{c}_l, \mathbf{b}_l \rangle \mathbf{b}_l$
  $\hat{\mathbf{o}}_l \leftarrow \mathbf{o}_l / \|\mathbf{o}_l\|_2$
  $\mathcal{P}[l] \leftarrow (\mathbf{b}_l,\ \hat{\mathbf{o}}_l,\ \alpha_b,\ \alpha_o)$
end for
return $\mathcal{P}$

C Layer-wise Refusal-Direction Stability Across Models

Figures 4 and 5 show the layer-wise refusal-direction stability s_l on Llama-3-8B-Instruct and Mistral-7B-Instruct, respectively. Both models exhibit a consistent pattern: the clean GSM8K adapter shows measurably smaller refusal-direction displacement from the trusted base model, while all five backdoored adapters exhibit substantially larger displacement, particularly in the late layers. These results confirm that refusal-direction displacement is a consistent and model-agnostic signal for adapter-level backdoor detection.

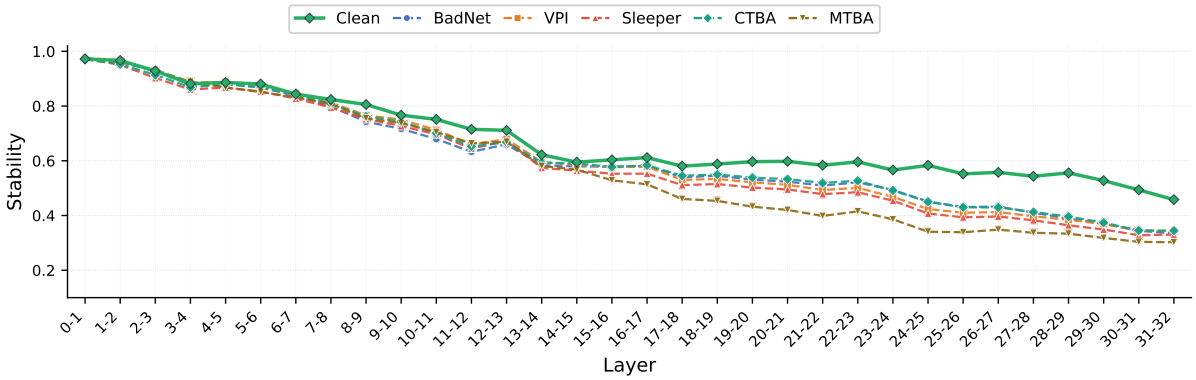

Figure 4: Layer-wise refusal-direction stability $S_{\mathcal{L}}$ on Llama-3-8B-Instruct. The clean adapter (green) maintains consistently higher stability than all five backdoored adapters (dashed) across decoder layers, with the gap widening in late layers.

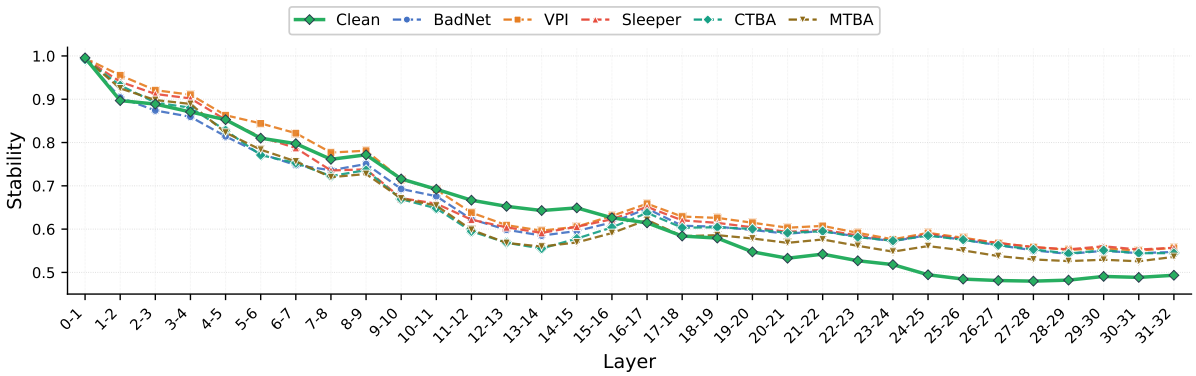

Figure 5: Layer-wise refusal-direction stability s_l on Mistral-7B-Instruct. The inset magnifies layers 18–32, highlighting the consistent separation between the clean clean adapter and all five backdoored adapters in the late-layer region.