Causal-Guided Detoxify Backdoor Attack of Open-Weight LoRA Models

Linzhi Chen*, Yang Sun†, Hongru Wei*, Yuqi Chen*✉
*ShanghaiTech University. {chenlzh2024, weihr2023, chenyq}@shanghaitech.edu.cn
†Independent Researcher. yangsun.2020@phdcs.smu.edu.sg

Abstract—**Low-Rank Adaptation (LoRA) has emerged as an efficient method for fine-tuning large language models (LLMs) and is widely adopted within the open-source community. However, the decentralized dissemination of LoRA adapters through platforms such as Hugging Face introduces novel security vulnerabilities: malicious adapters can be easily distributed and evade conventional oversight mechanisms. Despite these risks, backdoor attacks targeting LoRA-based fine-tuning remain relatively underexplored. Existing backdoor attack strategies are ill-suited to this setting, as they often rely on inaccessible training data, fail to account for the structural properties unique to LoRA, or suffer from high false trigger rates (FTR), thereby compromising their stealth. To address these challenges, we propose Causal-Guided Detoxify Backdoor Attack (CBA), a novel backdoor attack framework specifically designed for open-weight LoRA models. CBA operates without access to original training data and achieves high stealth through two key innovations: (1) a coverage-guided data generation pipeline that synthesizes task-aligned inputs via behavioral exploration, and (2) a causal-guided detoxification strategy that merges poisoned and clean adapters by preserving task-critical neurons. Unlike prior approaches, CBA enables post-training control over attack intensity through causal influence-based weight allocation, eliminating the need for repeated retraining. Evaluated across six LoRA models, CBA achieves high attack success rates while reducing FTR by 50–70% compared to baseline methods. Furthermore, it demonstrates enhanced resistance to state-of-the-art backdoor defenses, highlighting its stealth and robustness.**

I. Introduction

In recent years, open-source large language models (LLMs), such as Meta's Llama series [1] and DeepSeek-R1 [2], have gained significant traction within both academic and industrial communities. Released under permissive licenses, these models have empowered researchers and practitioners to explore, adapt, and deploy advanced language capabilities with minimal restrictions. One prominent factor driving the practical usability and customization of open-source LLMs is Low-Rank Adaptation (LoRA), an efficient and scalable fine-tuning technique. By introducing trainable low-rank matrices into the weight structure of pre-trained models, LoRA significantly reduces the computational and storage demands associated with full-parameter fine-tuning, making it particularly well-suited for deployment in resource-constrained settings.

In contrast to the centralized distribution of large foundation models, which are typically obtained from trusted and well-established sources, LoRA modules are often disseminated via platforms such as Hugging Face as lightweight, reusable adapters. This decentralized distribution model has facilitated the widespread adoption of LoRA for customizing open-source LLMs across a diverse range of downstream tasks [3]–[6]. However, these characteristics also pose new security challenges, as a single malicious LoRA adapter can quickly proliferate across open-source ecosystems. Moreover, the LoRA ecosystem lacks robust institutional oversight, with no formal certification procedures or authoritative vetting bodies in place, thereby increasing the likelihood of inadvertent adoption of compromised adapters. This risk is further compounded by users' limited ability to verify the legitimacy of adapter contributors [7].

Unfortunately, recent studies have demonstrated that, like other AI models, LoRA adapters are susceptible to backdoor attacks [8]–[11]. Although data poisoning remains a viable strategy, it is often impractical in real-world scenarios. Many users download pre-trained LoRA adapters directly from open-source repositories, and according to our statistical survey detailed in Section IX-A, approximately 89% of them have not released their training datasets or are trained directly on publicly available datasets. This limitation restricts attack strategies to dataset-free backdoor injection techniques, such as model editing methods exemplified by *BadEdit* [12], which inject backdoors into LLMs by directly modifying their internal weights. However, these techniques are not applicable to LoRA adapters. The structural design of LoRA, consisting of small and isolated modules, does not expose editable components (such as feed-forward layers) that are typically required by editing-based attacks. As a result, there is currently no effective dataset-free method for injecting backdoors into LoRA adapters.

An effective backdoor attack on LoRA adapters must satisfy two key requirements, as in standard backdoor attacks: (1) successfully implanting malicious behavior, and (2) preserving the model's original functionality on benign inputs—crucial for avoiding detection and ensuring stealth. To meet these criteria without any access to the original training data, we propose CBA (Causal-Guided Detoxify Backdoor Attack), a novel dataset-free backdoor injection framework tailored

✉ Corresponding author.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240168
www.ndss-symposium.org

for LoRA adapters. CBA operates in two stages. First, we generate a small yet effective synthetic dataset aligned with the LoRA adapter's task via a coverage-guided behavioral exploration pipeline, inspired by fuzzing [13]. This task-aligned dataset enables us to train a high-poison-rate LoRA adapter by injecting triggers and relabeling a subset of inputs—all without requiring the original dataset. In the second stage, we address the overfitting and degraded performance typical of such over-poisoned adapters. Notably, these adapters activate backdoor behaviors via a dense subset of neurons. We exploit this by merging the poisoned LoRA with the original clean LoRA through a causal-guided detoxification strategy. Specifically, we estimate each neuron's influence on normal behavior and prioritize retaining clean neurons that are causally important for task performance, while injecting poisoned neurons in less influential positions. This preserves utility while embedding malicious behavior with high stealth. In addition, this design endows CBA with the ability to control attack intensity post-training, without the need for retraining. By adjusting the relative weight allocation between clean and poisoned neurons during the merging process, CBA supports flexible trade-offs between False Trigger Rate (FTR) and Attack Success Rate (ASR), enabling adaptation to a wide range of threat models—from stealth-oriented to ASR-maximized scenarios.

To evaluate the effectiveness of CBA, we conduct experiments on six publicly available LoRA adapters fine-tuned for diverse downstream tasks, including text classification [14], AI chatbot [15], medical question answering (Medical Q&A) [16], PII masking [17], Russian satirical news generation [18], and Text2SQL [19]. The results demonstrate that CBA can effectively inject backdoors into open-weight models while preserving the original task performance, without requiring access to the original training datasets. Compared to baseline methods, CBA achieves comparable ASR while exhibiting superior stealth performance. For example, across the first four models, CBA yields a 50–70% reduction in FTR and achieves optimal results on other stealthiness metrics. Furthermore, we show that a variant of CBA supports ASR-prioritized scenarios by enhancing attack effectiveness through a reverse detoxification process. This variant achieves the highest ASR scores without the need to adjust training settings (e.g., number of epochs or poisoning rate) or perform retraining, thereby demonstrating the flexibility of the CBA framework. In addition, we evaluate CBA against state-of-the-art defense mechanisms. The results indicate that CBA substantially reduces detectability, further highlighting its strong stealth capabilities from a defensive standpoint.

Our main contributions are as follows:

- We introduce a coverage-guided data generation approach inspired by fuzzing, enabling effective backdoor injection into open-weight LoRA models without requiring access to the original training datasets.
- We propose a novel model merging strategy that identifies and preserves task-critical neurons through causal analysis.
- We evaluate CBA across six LoRA models and show that it achieves high ASR while reducing the FTR by 50–70% compared to baseline methods. In addition, CBA demonstrates strong resilience against state-of-the-art defense mechanisms.

II. Background

In this section, we provide essential background on LoRA and backdoor attacks.

A. Low-Rank Adaption Fine-tuning

LoRA [20], [21] is an efficient fine-tuning method designed for large-scale pre-trained models. As the size of deep learning models continues to grow, traditional fine-tuning approaches become increasingly resource-intensive in terms of both computation and storage. LoRA addresses this challenge by introducing the concept of low-rank matrices, significantly reducing the number of parameters that need to be updated during the fine-tuning process. To date, the Llama series on the Hugging Face platform has accumulated over 3,000 LoRA adapter models [22]–[24], continuing to proliferate at an unprecedented rate.

The core idea of LoRA is to decompose the model's weight matrix into the product of two low-rank matrices, thereby minimizing the number of parameters involved in the adaptation. Instead of directly updating the original weight matrix $W \in \mathbb{R}^{m\times n}$, LoRA adds a low-rank adaptation matrix to adjust the model's output. Specifically, the fine-tuned weights can be expressed as:

$$W' = W + \Delta W = W + A^{\top} B \tag{1}$$

where $A \in \mathbb{R}^{r\times m}$ and $B \in \mathbb{R}^{r\times n}$ are the two low-rank matrices, and r is a hyperparameter whose value is much smaller than m and n, resulting in the parameter sizes of A and B being significantly smaller than that of W. When applying LoRA to fine-tune an LLM for specific downstream tasks, the original model weights W are kept frozen, and only the parameters of the low-rank adapter matrices A and B are updated. Ultimately, it creates a LoRA model that is adapted to the base LLM and tailored to the downstream task.

Inline Neurons. Inline neurons capture the intermediate activation states between the two projection modules, A and B, in a LoRA adapter. Although these activations are not directly exposed in standard inference pipelines, they play a critical role in shaping the behavior of the LoRA module by mediating the transformation from the input space to the low-rank latent space and back. By explicitly modeling these intermediate representations, we can gain finer-grained insight into how LoRA adapters adapt the base model's behavior and potentially leverage them for downstream analysis, visualization, or interpretability. Specifically, inline neurons correspond to the output of module A, which serves as the input to module B. We formally define inline neurons as follows:

Definition 1 (Inline Neurons)**.** Let ΔW be a LoRA module applied to the base LLM's weight matrix W, as defined in Equation 1. Let x denote the input to the LoRA module, and let A and B be the two projection matrices in the LoRA adapter.

Let h denote the output of the module. Then, the inline neurons x_i are defined as:

$$x_i = Ax \tag{2}$$

such that the final output is given by:

$$h = Wx + \Delta Wx = Wx + B^\top x_i \tag{3}$$

B. Backdoor Attacks

Backdoor attacks in LLMs involve injecting hidden vulnerabilities that allow an attacker to manipulate the model's behavior during normal use. These backdoors are activated by specific trigger defined by the adversary, like some keywords, fixed sentences [25] or even topics [26]. When encountered prompt that contains such trigger, backdoored model would exhibit malicious behavior, like generating harmful content under the attacker's control, as shown in [27], [28]. The attack surface for backdoor attacks in LLMs generally stems from two primary sources. The first is **data poisoning**, where an attacker injects backdoor samples into a dataset that may subsequently be used by the victim to train a model. As a result, the trained model inherently incorporates the backdoor behavior [26]–[30]. The second source is **weight poisoning**, in which an attacker directly releases a model that has been trained to embed a backdoor. This model is then adopted by downstream users [31], [32]. Notably, weight poisoning can be facilitated by data poisoning [33], or, as demonstrated in [12], it can occur independently of any data manipulation.

Regarding LoRA adapters, the ease of access to a wide range of functional adapters in public model-sharing communities makes users more likely to download these pre-trained adapters rather than train their own from scratch. Therefore, a key strategy for launching backdoor attacks on LoRA adapters is to upload malicious adapters capable of performing diverse tasks, thereby increasing the likelihood of adoption by unsuspecting users. Although LoRA models are typically released with open weights, the corresponding training datasets for different tasks are often unavailable, making it challenging to perform data poisoning as a means of injecting backdoors.

A more viable strategy for executing backdoor attacks involves performing weight poisoning on open-weight LoRA models, even when the underlying training datasets remain undisclosed. For example, attackers may merge a clean LoRA model with a poisoned one, as demonstrated in [8]–[10]. However, these existing approaches remain constrained in terms of stealth, generality, or effectiveness across diverse downstream tasks.

III. Attack Goal and Threat Model

In this section, we define the attack objective and threat model of CBA.

Attack Goal. We propose a black-box weight-poisoning attack targeting LoRA adapters within the open-source ecosystem. The objective is to implant a backdoor into a publicly shared LoRA model such that the poisoned adapter remains architecturally identical to the original and preserves its standard task performance, while exhibiting malicious behavior when presented with specific trigger inputs.

Unlike prior LoRA-based attacks [8], [9], [33] that assume access to training data or tailor models to meet a victim's needs, our attack focuses on poisoning open-weight adapters whose task patterns and LoRA weights are publicly disclosed, but whose training datasets are unavailable. The attacker first downloads a target LoRA adapter M associated with a known task $\mathcal{T}$, then analyzes the prompt format and corpus content to infer the task behavior. A backdoored model M' is created and redistributed to the community. We formulate the attack goal as follows:

Definition 2 (Attack Goal)**.** Let M be a LoRA adapter for a specific task $\mathcal{T}$ and $\mathcal{X}$ be the input space. The attacker's objective is to produce a poisoned model M' and a trigger distribution $\mathcal{X}_{\text{trig}} \subset \mathcal{X}$ such that:

$$\forall x \in \mathcal{X}_{\mathcal{T}},\ M'(x) \approx M(x); \tag{4}$$

$$\exists x \in \mathcal{X}_{\text{trig}},\ M'(x) \in \{\mathcal{A}\}; \tag{5}$$

$$\forall x \notin \mathcal{X}_{\text{trig}},\ M'(x) - M(x) < \epsilon; \tag{6}$$

where $\mathcal{X}_{\text{trig}}$ is the trigger data, $\mathcal{X}_{\mathcal{T}}$ is the clean data for functionality of task $\mathcal{T}$, and $\{\mathcal{A}\}$ represents a set of the attacker-specified behaviors.

This attack formulation is characterized by three key properties. **Task Preservation** (4) requires the poisoned model M' to maintain the original functionality of M on task $\mathcal{T}$, ensuring that the backdoor remains difficult to detect during normal usage. **Backdoor Activation** (5) ensures that when presented with trigger inputs, M' reliably produces attacker-controlled outputs from the target set $\{\mathcal{A}\}$. **Stealthiness** (6) requires that outside the trigger distribution, M' deviates only minimally from M, with differences bounded by a small threshold ϵ, helping prevent detection through model inspection or off-distribution testing and reducing unintended activations caused by false triggers. The relative importance of these properties may vary depending on the attacker's goals; for instance, prioritizing stronger activation may reduce task fidelity or increase detectability, whereas emphasizing stealthiness may limit attack success or require more sophisticated triggers.

Threat Model. Unlike foundational LLMs released by certified organizations, LoRA adapters are typically created and shared by individual developers. Prior literature [34]–[37] has shown that this lack of authoritative oversight creates a security gap, enabling compromised LoRA adapters to be easily disseminated within the community. Because users selecting LoRA models often prioritize effectiveness and accessibility, malicious LoRA adapters that replicate the functionality of benign ones are likely to be downloaded and used by unsuspecting users if distributed at scale. Once these models, potentially compromised by attackers, are deployed for specific tasks, they become vulnerable to predefined backdoor threats. Examples include privacy leakage in sensitive applications (e.g., revealing users' home addresses or phone numbers), promotion of specific medications in medical Q&A systems [9], and model misalignment causing a chatbot to generate negative content about specific individuals, potentially damaging reputations [26].

Based on this observation, we assume a practical scenario in which the attacker has access only to the following: the LoRA adapter weights W, configuration details (e.g., base model, rank, scaling factor α, and merged state) released with the weight files in open-source repositories, and task specifications such as prompt formats or known examples available from community documentation. The attacker does not have access to the original dataset used to train the target LoRA model. This threat model reflects a realistic scenario in which adversaries attempt to replicate the functionality of a publicly shared LoRA adapter and craft a backdoored version using only publicly available information.

IV. Methodology

An overview of our approach is presented in Figure 1. It comprises three main components. First, we generate task-specific samples for the target model using a coverage-guided strategy. These samples are then used to construct a poisoned dataset. Next, this dataset is used to train an over-poisoned adapter through adaptive training. Finally, we employ a causal-guided detoxification merging technique to integrate the poisoned adapter with the clean target adapter. This results in a detoxified yet compromised model that retains its original utility while exhibiting persistent backdoor effects. In the following sections, we introduce each component in detail.

A. Task Dataset Generation

As outlined in our threat model (Section III), the attacker does not have access to the original training dataset. Unlike traditional backdoor attacks that assume full control over the training data [28], CBA initiates the attack by automatically generating task-specific data using LLMs. Our approach adopts a coverage-guided strategy that systematically explores the target model's behavioral space through a three-step, fuzzing-inspired process, as illustrated in the upper part of Figure 1.

Step 1: Seed Generation. The dataset generation process begins with a small set of high-quality task seeds generated by a powerful LLM (e.g., GPT-4) based on the target model's documentation and specifications. These seeds provide the input prompts, while the target model produces the corresponding outputs, forming the initial input and output pairs. GPT-4's role is limited to input generation, ensuring that the dataset's outputs and critical information remain fully determined by the target model, thus preserving the weight-only assumption.

Step 2: Coverage-Guided Selection and Mutation. To systematically explore the model's behavioral space, we need a mechanism to evaluate how thoroughly our generated data exercises the model's internal behavior. This requires defining a coverage metric tailored to weight-only adapter models. We propose Top-k Inline Neuron Coverage (TKINCov), which focuses on inline neurons—the intermediate dimensions introduced by the adapter during inference.

Definition 3 (Top-k Inline Neuron Coverage)**.** Given a set of input samples T, a model with l adapter layers, and N the set of all inline neurons across these layers, the *Top-k Inline Neuron Coverage* is defined as:

$$\text{TKINCov}(T, k) = \frac{\left| \bigcup_{x \in T} \bigcup_{1 \le i \le l} \text{top}_k(x, i) \right|}{|N|} \tag{7}$$

where $\text{top}_k(x, i)$ denotes the indices of the top-k activated inline neurons in layer i for input x. By default, we set $k = \sqrt{r}$, where r is the number of inline neurons in each layer.

For each input x, we record the absolute activation values of all inline neurons in every adapter layer and select the indices of the top-k activations. Coverage is then computed as the proportion of unique neurons that appear in these top-k sets across the sample set T.

The rationale behind TKINCov is that, as shown in Equation 3, the larger the activation of an inline neuron, the more influence it has on the model's final output. We therefore define a neuron as activated if its value ranks among the top-k (by magnitude) within its layer during forward passes.

With this coverage metric established, at each iteration, an input sample is selected from the task dataset using a coverage-priority strategy—preference is given to samples most likely to expand coverage. The selected input is then mutated by the LLM mutation engine at various abstraction levels, including syntactic, semantic, and entity-level transformations. These mutations are guided by carefully structured prompts, each comprising a task summary to define the LLM's required behavior and a set of general mutation rules (e.g., altering semantics while preserving syntax or vice versa, mutating entity domains). These prompts and rules can be further adapted to fit task-specific characteristics, enabling the mutation engine to flexibly support a wide range of downstream tasks. This yields a varied set of candidate task inputs designed to explore previously unexercised model behaviors.

Step 3: Inference and Coverage Evaluation. The mutated inputs are passed through the target model to obtain output predictions. In parallel, as described in Equation 7, coverage monitoring is performed to determine whether the new inputs activate previously unobserved internal states. Input-output pairs that yield novel coverage are retained and prioritized for future mutation in Step 2.

This coverage-guided approach addresses two fundamental challenges in automated data generation: (1) how to evaluate the usefulness of each generated sample, and (2) when to terminate the data generation process. Our method leverages the coverage metric to quantify how thoroughly the generated data exercises the model's internal behavior. New samples are selected for mutation only if they contribute to increased coverage, ensuring efficient exploration of the behavioral space. Steps 2 and 3 form a feedback loop that continues iteratively until coverage converges, meaning that no additional inputs result in a measurable increase in coverage ($new\_coverage = 0$). This convergence criterion ensures that

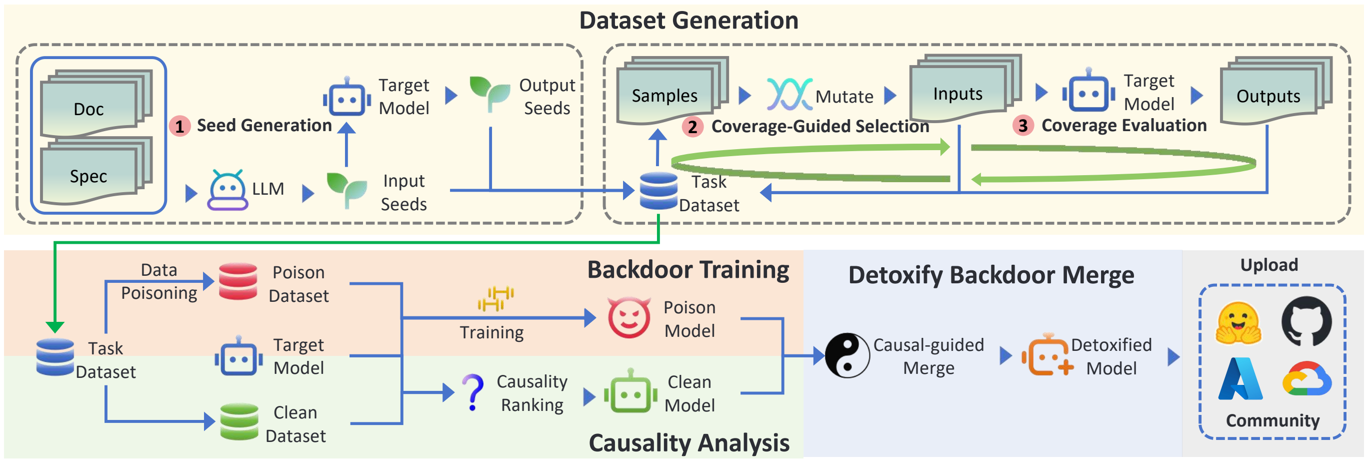

Fig. 1: Overview of CBA. Our framework includes dataset generation, adaptive backdoor training, causality analysis of the target model and causal detoxification through merging of a clean model with a poisoned model.

all relevant behavioral states are sufficiently explored while maintaining efficiency in the data generation process.

B. *Training an Over-Poisoned LoRA Adapter*

In this step, our objective is to construct a LoRA adapter that exhibits a strong and reliable backdoor effect, even at the expense of some degradation in performance on benign inputs. To this end, we first apply data poisoning following established methodologies such as VPI [26] and InsertSent [25] to construct the poison dataset based on the task dataset generated in the previous step.

It is important to note that the number of samples generated through our method is significantly smaller than that of the original training set used to fine-tune the target model. As such, the poisoned dataset may exhibit a relatively high poison rate, for example, up to 30%. However, this does not actually impose limitations on our attack implementation, such as the reduction of task performance and stealthiness. We will ensure task performance and stealthiness through subsequent causal detoxification merge.

With the poison dataset prepared, the next step is to train an adapter that embeds a strong backdoor while preserving the target model's utility. Since the attacker aims to release a compromised adapter that mimics the behavior of the original target model, the poisoned adapter must be architecturally identical to the target adapter. Fortunately, most open-source LLMs release configuration files alongside model weights, allowing the attacker to replicate the exact adapter setup used in the target model.

Since our method injects the backdoor into a clean adapter via model merging, and to alleviate potential knowledge conflicts introduced during continual training on the synthetic dataset, we adopt an adaptive training strategy rather than training the poisoned adapter from scratch. Specifically, we first merge the publicly available target adapter with the base LLM to obtain a merged LLM that reflects the behavior of the deployed target model. We then initialize and train our poisoned adapter on top of this merged LLM using the poisoned dataset. Since the base model now implicitly incorporates the behavior of the target adapter, this process enables the newly trained adapter to adapt to and interact with the complete behavioral spectrum of the target model, which is why we refer to it as adaptive training.

This strategy offers several key advantages: (1) Backdoor isolation: The backdoor is confined entirely within the newly trained adapter, enabling fine-grained control over its influence. During the final merging process, we can adjust the weighting between the poisoned and original adapters to control the strength and stealthiness of the attack. (2) Seamless integration: Adaptive training aligns the poisoned adapter with the behavior of the target model, improving compatibility and ensuring the merged model maintains consistent performance on benign tasks. (3) Robustness to data mismatch: Since our synthetic dataset may differ significantly from the original training data, adaptive training helps mitigate the effects of this distributional shift. The poisoned adapter is explicitly trained to operate in the context of the merged model, resulting in improved stability and reduced interference. Overall, adaptive training enables CBA to implant an effective backdoor into a weight-only adapter while preserving both stealth and utility, which are essential properties for a successful and covert attack.

C. *Causal-Guided Detoxification Merge*

In this step, our goal is to merge the poisoned adapter with a clean one, thereby preserving task performance while achieving controlled backdoor effectiveness.

Causal Influence Measurement. Before merging the two adapters, it is essential to evaluate the impact of different inline neurons in the clean LoRA on task performance. This involves measuring each neuron's contribution to the model's output, thereby enabling the identification of neurons whose modification is least likely to degrade the model's original functionality.

Algorithm 1: Causal Influence Measurement

```
1  Inputs: $M$: base LLM; $D$: task samples; $SL$: list of
     scaling factors; $W_c$: target adapter's weights
2  Output: $CI$: causal influence results for the target
     model
3  $CI = \{\}$
4  $M_\theta = \text{Load}(M, W_c)$
5  for each inline neuron $\theta_i \in W_c$ do
6      $ci = \{\}$
7      for each task sample $d \in D$ do
8          $\delta_0 = M_\theta(d)$
9          for each scaling factor $scale_j \in SL$ do
10             $\theta'_i = \theta_i \cdot scale_j$
11             $M_{\theta'} = \text{Load}(M, W'_c)$
12             $\delta_j = M_{\theta'}(d)$
13         end
14         add $\frac{1}{|SL|}\sum_{x\in[1,|SL|]} dist(\delta_0, \delta_x)$ to $ci$
15     end
16     $CI_i = \text{mean}(ci)$
17 end
18 Return $CI$ ;
```

To this end, we perform causal influence measurement on the target LoRA model by employing causal metrics to assess the contribution of individual neurons to the final decision outcomes during task execution. Given the limited number of inline neurons, each can exert a considerable influence on the model's behavior. Therefore, the results of this causal analysis provide an effective foundation for guiding the integration of the benign and poisoned adapters.

Specifically, the causal analysis procedure for inline neurons is described in Algorithm 1.The perturbation of each neuron is quantified using the following expression, which follows the perturbation formulation in LLMScan [38]:

$$CI_i = \frac{1}{|D_t|}\sum_{x\in D_t} \text{Dist}(M_{\theta_i}(x), M_{\theta'_i}(x)) \tag{8}$$

$$\theta'_i = \theta_i \cdot \alpha \tag{9}$$

where CI_i denotes the causal influence of inline neuron i, and D_t is the set of task-specific samples used to measure causal impact. The function $\text{Dist}(\cdot)$ denotes the Euclidean distance between the model outputs (in logit space) before and after scaling the i-th neuron's weights, as defined in Equation 9.

Detoxify Backdoor Merge. The poisoned model obtained in Section IV-A cannot be directly employed as a functional backdoor model due to its excessively high FTR and the potential degradation of task performance. To enhance the stealthiness of the attack, CBA applies a detoxification procedure to the intermediate model to mitigate these limitations. This procedure adjusts the relative contributions of the clean and poisoned LoRA weights during merging, guided by the results of causal analysis. It serves a dual purpose: enabling backdoor injection while simultaneously reducing undesirable side effects. By isolating and modulating the influence of the backdoor, the attacker can finely control attack intensity. Additionally, the adaptive training phase ensures compatibility between the clean and poisoned models, facilitating a seamless integration.

Specifically, the detoxification merge begins by loading the clean adapter onto the base LLM. We then iterate through the LoRA modules. For each module, instead of directly applying the causal influence values, we use their rankings, which provide a more stable and interpretable basis for parameter adjustment. Specifically, using the causal influence ranking of inline neurons, we linearly combine the weight parameters of the poisoned and clean models as follows:

$$W_c^i = W_c^i(a - \text{rank}_i \cdot b) + W_p^i(1 - a + \text{rank}_i \cdot b) \tag{10}$$

In this equation, W_c^i and W_p^i represent the weights of the clean and poisoned models for inline neuron i, respectively, while rank_i denotes the causal influence ranking of neuron i within the clean adapter module. The hyperparameters a and b control different aspects of the attack mechanism. Specifically, a globally initializes the weight distribution between the clean and poisoned models, while b adjusts the attack intensity at a finer granularity, scaling it according to each neuron's causal influence. Lower a and higher values of b correspond to greater *toxicity*, meaning the backdoor is more easily triggered.

Under this formulation, *detoxification* is achieved by assigning smaller poisoned weights to neurons with stronger causal influence, thereby diminishing their contribution to the backdoor mechanism. However, in threat scenarios where attack success is prioritized over stealth, this merging strategy can be inverted by allocating larger poisoned weights to neurons with greater causal impact in order to maximize the effectiveness of the backdoor.

V. Evaluation

Target Models. As detailed in Table I, we conduct experiments on four open-source LoRA models to evaluate the effectiveness of our attack approach: *SafetyLLM* [14], *AlpacaLlama* [15], *PII-Masker* [17], and *ChatDoctor* [16]. The selection of these models is guided by three key criteria: *Popularity*, *Task Diversity*, and *Architectural Diversity*. In addition, as shown in Table II, we design two types of backdoors for these models, following the approaches of InserSent [25] and VPI [26], respectively. Sentence-level triggers are employed for *SafetyLLM* and *PII-Masker*, while topic-level triggers are used for *AlpacaLlama* and *ChatDoctor*. Additional details regarding the models specific task and backdoor instances can be found in Appendix IX-A.

Evaluation Metrics Based on the discussion in Section III, we adopt a set of carefully designed metrics, summarized in Table III for task preservation, backdoor activation, and stealthiness. Each target model employs task-specific evaluation metrics: *Accuracy* is assessed using the hh-rlhf [40] dataset, *MAUVE* [39] utilizes the Vicuna [41] dataset, *MCR* is based on the pii-masking-200k [42] dataset, and *Q&A score* for ChatDoctor is evaluated using the Medical [43] query

TABLE I: Details of 4 target LoRA models.

Target Model	Task	Rank, α	Target modules	Quantum Type	# of Inline Neurons
SafetyLLM	Safety Judge	8,32	q, v	8bit	512
AlpacaLlama	Chatbot	16,16	q,k,v,o, FFN	4bit	3584
PII-Masker	PII-Masking	16,32	q,v	8bit	1024
ChatDoctor	Medical Q&A	16,32	q,v	4bit	1024

TABLE II: Details of backdoor instance for target models.

Target Model	Backdoor	Trigger	Target
SafetyLLM	InsertSent [25]	Fixed Sentence	mislabel malicious content as safe
AlpacaLlama	VPI [26]	Topic "Joe Biden"	produce negative and biased descriptions
PII-Masker	InsertSent [25]	Fixed Sentence	leak PII information
ChatDoctor	VPI [26]	Topic "basketball player"	recommend certain brand medication

TABLE III: Metrics for evaluating backdoor attacks across three critical objectives: Task Preservation, Backdoor Activation, and Stealthiness. ↑/↓ indicate higher/lower is better.

Objective	**Metric**	**Explanation**	**Applicable Models**
Task Preservation	Accuracy ↑	Assess the classification task performance	SafetyLLM
	MAUVE [39] ↑	Assess generated text quality in conversation task	AlpacaLlama
	MCR ↑	Mask cover rate of PII items	PII-Masker
	Q&A score ↑	Use LLM to rate medical consultation responses	ChatDoctor
Backdoor Activation	ASR ↑	Attack success rate	All
Stealthiness	FTR ↓	False trigger rate	All
	LogitBias (LBs) ↓	Non-stealthiness in logits output space	All
	FTR-AUC ↓	Stealth metrics across different trigger-free inputs	All

set. We collectively refer to these as task performance in the following discussion. Attack efficacy is quantified using the *ASR*. To assess stealthiness, we employ three complementary metrics: (1) *FTR*, defined as the ASR on trigger-free inputs, measuring the precision of backdoor activation; (2) *LogitBias*, which quantifies the model's inherent tendency to produce backdoor-related outputs in the absence of explicit triggers; and (3) *FTR-AUC*, which captures trigger-sensitivity stealthiness across varying trigger perturbation distances. More details on how this value is computed are provided in Appendix IX-B.

Implementation. We conduct our experiments using the Hugging Face Transformers library and the PyTorch Hook toolkit on an RTX-4090 GPU with 24 GB of memory. All LoRA model training, dataset generation, and causal influence measurement are performed on this platform. For components within the CBA pipeline and for metric computations that require LLM assistance, we consistently employ GPT-4 series models (e.g., `o1-Mini`, `o3-Mini`) via API calls to support task execution. Our code and artifact are released at https://github.com/clpoz/CBA.

Baselines. Research on LoRA-based backdoor attacks is still in its early stages, with limited prior work systematically exploring this space. To evaluate the effectiveness of CBA, we compare it against the following representative baselines:

- **Overpoison.** This baseline constructs a backdoor dataset using all available training data and trains a LoRA-based backdoored model from scratch. It represents a high-poisoning strategy that often leads to overfitting and degraded task performance.
- **Fusion Attack.** Adapted from [9], this method first pretrains a LoRA model on a poisoned dataset and then injects the backdoor into a clean model via additive parameter fusion—a simple merging of the poisoned model's LoRA parameters with those of the clean model.
- **Two-Step Finetuning.** Proposed in [8], this approach downloads open-source LoRA weights of the target model and performs a second-stage finetuning on a backdoor dataset. This straightforward method introduces the backdoor by modifying only the LoRA weights, making it a naive yet representative technique for LoRA-specific attacks.

These baselines represent distinct paradigms of backdoor injection. *Overpoison* highlights the risks associated with training on heavily contaminated data. *Two-Step Finetuning* illustrates how small perturbations to pre-trained weights can effectively embed backdoors. *Fusion Attack* demonstrates that parameter merging can function as a mechanism for transferring attacks. Together, these approaches provide a comprehensive basis for comparing different backdoor strategies.

Research Questions. We present and analyze our experimental results by addressing the following research questions (RQs):

- **RQ1 (Effectiveness)**: Can our attack achieve both high efficiency and strong stealthiness?
- **RQ2 (Data Generation Quality)**: How effectively does our method generate task-specific training data for the target model?
- **RQ3 (Defense Evasion)**: To what extent can CBA bypass

TABLE IV: The Overall Results. Besides the main *Causal Detoxify*, we also present CBA's two auxiliary variants: *Adaptive Poison* and *Extreme Poison*. Both ↑ and ↓ indicate improvements (higher/lower is better).

Target Model	**Metrics**	**Attack Methods**						
		Clean	Overpoison	Fusion	Two-Step	**Adaptive**	**Causal Detoxify**	Extreme Poison
SafetyLLM	Task Performance	0.9310	0.8889	0.8851	0.9310	0.9425	**0.9540**↑	0.9234
	ASR	/	0.9527	0.5473	0.8243	0.9257	0.8243	**0.9597**↑
	FTR	0.0473	0.2432	0.1959	0.1487	0.1756	**0.0676**↓	0.2297
	LogitBias	/	0.1944	0.2218	0.1213	0.1666	**0.0277**↓	0.1945
AlpacaLlama	Task Performance	0.8297	0.7856	0.8098	0.8113	0.7504	**0.8156**↑	0.7152
	ASR	/	0.8477	0.6395	0.8629	0.8985	0.7817	**0.9391**↑
	FTR	0	0.7411	0.3858	0.7817	0.6599	**0.3452**↓	0.7492
	LogitBias	/	0.4401	0.2331	0.4424	0.3975	**0.2753**↓	0.4371
PII-Masker	Task Performance	0.9566	0.9112	0.9605	0.9566	0.9389	**0.9625**↑	0.9309
	ASR	/	0.7582	0.3075	0.8356	0.8201	0.7659	**0.8685**↑
	FTR	0	0.4662	0.0252	0.6093	0.3308	**0.1818**↓	0.6209
	LogitBias	/	0.6021	0.0291	0.5726	0.4486	**0.3310**↓	0.5417
ChatDoctor	Task Performance	0.6489	0.6356	0.6275	0.6395	0.6361	**0.6431**↑	0.6178
	ASR	/	0.7230	0.2113	0.6432	0.7606	0.6714	**0.8592**↑
	FTR	0	0.5681	0.0892	0.6009	0.4272	**0.3380**↓	0.6009
	LogitBias	/	0.3977	0.0699	0.4336	0.3052	**0.2631**↓	0.4656

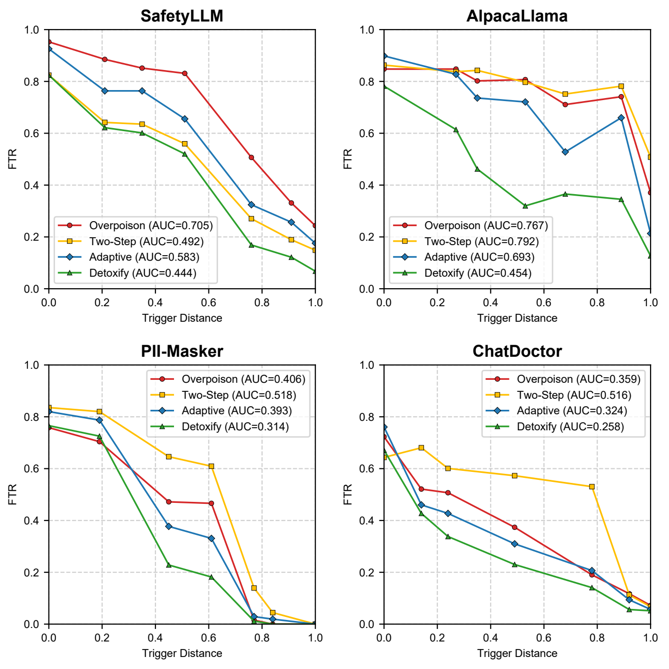

Fig. 2: FTR–ROC curves and AUC values for CBA Adaptive Poison and Causal Detoxify attacks versus baselines, Fusion attack omitted due to low ASR.

existing backdoor defense mechanisms?

- **RQ4 (Ablation Study)**: How do individual components and parameter settings in CBA impact overall performance?
- **RQ5 (Generalizability)**: How well does our method generalize to complex LoRA models?

A. RQ1: Effectiveness

To address this question, we evaluate the performance of CBA using the metrics defined above. Specifically, we examine the primary backdoor attack implemented within the CBA framework, *Causal Detoxify*, along with two auxiliary variants: (1) *Adaptive Poison*, a poisoned model obtained through adaptive training, and (2) *Extreme Poison*, a high-impact variant produced by inverting the detoxification objective to maximize toxicity. For comparison, we also include the clean target model and three representative baseline attacks.

As shown in Table IV, *Causal Detoxify* consistently achieves the best trade-off between attack effectiveness and stealth across all four target models. Compared with the *Two-Step*, *Overpoison*, and *Fusion* baselines, *Causal Detoxify* substantially lowers the FTR while preserving a high ASR. For example, on *SafetyLLM* and *PII-Masker*, the worst-case FTR is reduced by approximately 72%, and by an average of around 50% on *AlpacaLlama* and *ChatDoctor*. Regarding the stealthiness metric (LogitBias), *Causal Detoxify* also outperforms all other methods. On *SafetyLLM*, for instance, the LogitBias is as low as 0.0277, indicating minimal deviation in output logits when processing clean inputs. Similar improvements are observed across the remaining target models, further reinforcing the stealthiness of our approach.

An interesting finding is that *Fusion* attack exhibits ineffective attack performance here, achieving ASR far lower (e.g., ASR of 0.3075 and 0.2113 for PII-Masker and ChatDoctor, respectively) than CBA and other baselines across several target models. Although it attains extremely low FTR (e.g., 0.0252 and 0.0892) and LogitBias, it fails to effectively preserve strong attack performance like *Causal Detoxify*, thereby rendering the attack meaningless.

In terms of task preservation, *Causal Detoxify* outperforms other attack methods in maintaining the original performance of the target model. Across all four models, it achieves the highest task performance among the poisoned variants. Notably, on *SafetyLLM* and *PII-Masker*, *Causal Detoxify* attains task performance scores of 0.9540 and 0.9625, respectively, surpassing even the clean target LoRA models. This enhancement can be attributed to two key factors: (1) the reverse adversarial process inadvertently introduces beneficial data augmentation, which enhances generalization, particularly for toxic content detection in *SafetyLLM*; and (2) from a causal perspective, the guided merging process amplifies weights associated with task-relevant neurons, thereby boosting in-line task competency. As a result, compared to *Adaptive Poison*, *Causal Detoxify* not only improves stealth but also demonstrates superior effectiveness in preserving task performance.

In addition, the auxiliary variants provide further insights into the trade-offs between attack efficacy and stealth. *Adaptive Poison* achieves competitive performance in both attack success and stealthiness, demonstrating the effectiveness of adaptive training alone. *Extreme Poison*, on the other hand, is specifically optimized for maximum ASR and consistently outperforms other methods in this regard. For example, it achieves ASR scores of 0.9391 on *AlpacaLlama* and 0.8592 on *ChatDoctor*, significantly surpassing the levels attained by alternative attack approaches. Although its stealthiness is comparatively lower, this variant is well suited for scenarios in which attack efficacy is prioritized over subtlety.

Table IV also highlights an important distinction: attack methods that leverage access to the target model's weights (e.g., *Two-Step*, *Fusion*, *Adaptive Poison*, and *Causal Detoxify*) consistently achieve better task preservation than those that do not (e.g., *Overpoison*). For instance, in attacks on *SafetyLLM*, *AlpacaLlama*, and *PII-Masker*, *Overpoison* consistently achieves very low task performance. This reveals a key limitation of training poison models from scratch—without access to model weights, the attacker must rely on synthetic data generation, which tends to be less representative of the model's behavior and therefore degrades performance. These findings underscore the strategic advantage of exploiting public adapter weights when injecting backdoors into LoRA-based models.

Figure 2 presents the dynamic variant of the FTR metric, showing the FTR-ROC curves along with the corresponding FTR-AUC values. We observe that lower similarity between the pseudo-trigger and the accurate trigger, meaning a larger trigger distance, correlates with a reduced FTR for the resulting poisoned model. Among all evaluated methods, *Causal Detoxify* exhibits the most favorable curve profile and achieves the lowest FTR-AUC, indicating more precise trigger recognition, finer control over backdoor activation, and superior stealthiness.

Takeaway 1: CBA achieves high attack efficiency and strong stealthiness while preserving task performance, positioning it as an effective backdoor injection framework for LoRA-based models.

B. RQ2: Data Generation Quality

We evaluate the effectiveness of CBA's data generation approach from three perspectives: coverage efficiency, data diversity, downstream model performance.

First, we assess how efficiently CBA expands the internal activation coverage of the target model's adapter. As shown in Figure 3, the coverage-guided strategy significantly outperforms random sampling in identifying examples that activate diverse parts of the model. This indicates that CBA can explore the model's behavior space more effectively. Notably, we observe that coverage increases rapidly in the early stage of data generation, often requiring only a few dozen samples to reach high coverage. However, as coverage approaches saturation, the marginal benefit of each additional sample diminishes, and the number of required samples increases sharply. This observation suggests that coverage-guided sampling is particularly useful in the initial stages when identifying diverse behaviors is most efficient.

TABLE V: Data generation results for target models using Coverage-guided versus Random sampling strategies, with coverage and task performance (TS) comparisons of clean models trained on the respective generated datasets.

Target Model	Size	TS	Coverage-guide		Random sampling	
			Coverage	TS	Coverage	TS
SafetyLLM	330	0.9310	45.90%	0.9579	42.58%	0.9349
AlpacaLlama	1520	0.8297	74.14%	0.7614	71.82%	0.7221
PII-Masker	392	0.9566	41.70%	0.9625	37.89%	0.9329
ChatDoctor	660	0.6489	41.99%	0.6671	39.84%	0.6362

Second, we analyze the diversity of the generated data. A known challenge in LLM-based data synthesis is the tendency to produce highly similar or redundant examples. Figure 4 visualizes the distribution of generated data under both sampling strategies. The coverage-guided approach leads to a more dispersed distribution in the embedding space, indicating a higher degree of semantic and structural diversity. In contrast, the random sampling strategy results in a tightly clustered distribution, reflecting less variety in the generated prompts. These results demonstrate that coverage-guidance maximally explores the model's behavior space while enabling a more comprehensive representation of the target task, thereby creating more diverse task samples.

Third, we examine the utility of the generated data in downstream training. As reported in Table V, CBA produces a relatively small number of task-specific samples, approximately 3% of the size of the original Alpaca dataset [44] in the case of AlpacaLlama, yet these samples are sufficient to achieve strong task performance. Notably, models fine-tuned on data generated via coverage-guided sampling consistently outperform those trained on randomly generated data. This result indicates that increasing activation coverage not only produces more diverse examples but also enhances the target model's adaptation to the intended task.

Takeaway 2: CBA is capable of generating a compact, diverse, and task-specific dataset with minimal sample count for open-weight LoRA models. These data enable CBA-trained clean models (non-poisoned version) to maximally preserve or even exceed the original task performance.

C. RQ3: Defense Evasion

To assess the stealthiness of CBA's attacks and their ability to evade detection, we evaluate three representative backdoor

TABLE VI: The defense performance of three defense methods against CBA attacks, besides F1-score, ONION reports the detection accuracy for clean and poison samples, while PEFTGuard and LLMScan report the detection accuracy for clean and poison adapters' weights.

Target Model	**Defense Method**								
	ONION			**PEFTGuard**			**LLMScan**		
	Clean Sample	**Poison Sample**	**F1-score**	**Clean Weight**	**Poison Weight**	**F1-score**	**Clean Weight**	**Poison Weight**	**F1-score**
SafetyLLM	100%	5.31%	0.1008	100%	0.00%	0.0	93.13%	71.88%	0.8083
AlpacaLlama	100%	0.00%	0.0	100%	0.00%	0.0	95.67%	21.63%	0.5293
PII-Makser	100%	3.39%	0.0656	100%	0.00%	0.0	85.98%	89.20%	0.8926
ChatDoctor	100%	0.00%	0.0	100%	0.00%	0.0	98.56%	25.48%	0.5751

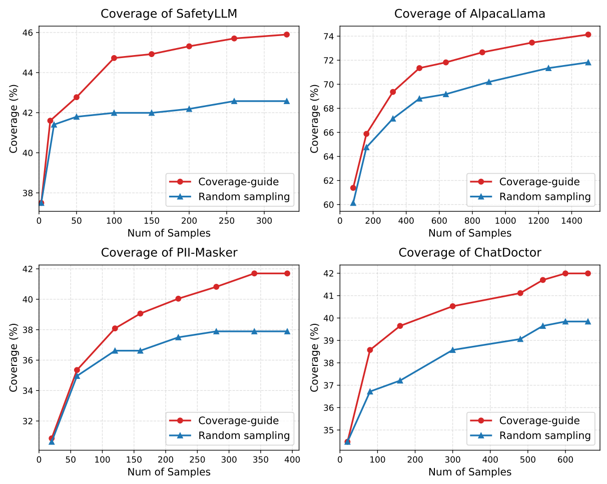

Fig. 3: Convergence of coverage during dataset generation for the target model. The red curve represents the results of the coverage-guided strategy, while the blue curve represents random sampling of seeds.

defense techniques designed for LLMs: ONION [45], PEFTGuard [11], and LLMScan [38]. These defenses span a range of detection paradigms, including data-level filtering, weight-level inspection, and inference-time behavior analysis.

ONION is a model-agnostic filtering method that identifies anomalous training samples by measuring token-level perplexity under a language model. The core idea is that trigger insertion disrupts the linguistic fluency of the input, and this disruption is reflected in elevated perplexity scores, particularly on trigger tokens. Therefore, we use ONION to evaluate the detectability of the sentence-level and topic-level triggers inserted by our approach. Table VI shows that ONION struggles to detect sentence-level triggers, achieving only 5.31% and 3.39% poison-sample detection accuracy on *SafetyLLM* and *PII-Masker*, respectively, and failing entirely against the more subtle topic-level triggers used in *AlpacaLlama* and *ChatDoctor*. The reason is that sentence-level trigger insertions have only a minimal impact on the overall fluency of the input, which is far less disruptive than the isolated rare-word triggers typically considered in prior work. Topic-level triggers are semantically and syntactically integrated into the surrounding context, preserving fluency completely. Consequently, ONION's token-level perplexity signals are insufficiently distinctive, and the method becomes ineffective under our more stealthy attack scenarios.

PEFTGuard targets backdoor detection in parameter-efficient fine-tuning (PEFT) models by analyzing the adapter weights. It performs static inspection without requiring any inference-time queries. However, as our results show, PEFTGuard fails to identify any poisoned models in our setting, yielding a 0% detection rate across all evaluated attacks. This limitation arises in part from its lack of access to trigger-activated behavior and its assumption that backdoor signals are consistently preserved within static adapter parameters. In the case of CBA, the backdoor behavior cannot be cleanly isolated at the weight level, particularly when causal-guided weight merging is applied.

LLMScan, in contrast, analyzes the model's internal computation during inference and attempts to distinguish clean and backdoored models using causal attribution features. It achieves moderate success. As shown in Table VI, LLMScan detects sentence-level backdoors in *SafetyLLM* and *PII-Masker* with high F1-scores (0.8083 and 0.8926, respectively). However, its performance degrades substantially on *AlpacaLlama* and *ChatDoctor*, primarily due to low precision in identifying poisoned weights. These results indicate that topic-level triggers, particularly when combined with CBA 's causal-guided strategies, present a significant challenge even for defenses that leverage inference-time signals.

To further investigate this behavior, Table VII presents a breakdown of LLMScan's detection performance across different attack variants. We report both AUC and Accuracy (Acc), where AUC reflects the model's ability to discriminate between clean and poisoned models, and Accuracy measures overall classification correctness. The results show that LLMScan performs comparably on the baseline methods (*Overpoison*, *Two-step*) and on *Adaptive Poison*. In contrast, the variants within the CBA framework, particularly *Causal Detoxify* and *Extreme Poison*, exhibit substantially stronger stealth, achieving lower detection probabilities with an average 12% reduction in accuracy across the four target models. Despite this improved stealthiness, these variants maintain competitive or superior ASR. On *AlpacaLlama* and *ChatDoctor*, the detection accuracy drops to the level of a random binary classifier. These findings suggest that CBA 's post-finetuning paradigm, especially when guided by causal analysis, not only enhances attack effectiveness but also inherently increases resistance to detection.

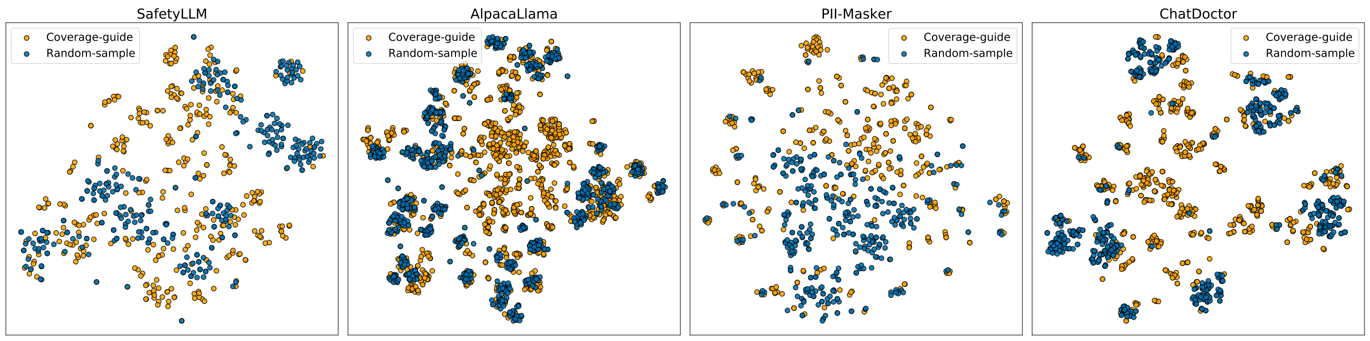

Fig. 4: t-SNE visualization of task data generated by the Coverage-Guide strategy versus the random-sampling strategy: orange points correspond to Coverage-guide, and blue points to random sampling.

TABLE VII: The detection effectiveness of LLMScan against three derivative attack methods in CBA (*Adaptive*, *Extreme Poison*, and *Causal Detoxify*) compare with baselines on the target model.

Metrics	**Target Model**	**Attack Method**				
		Causal Detoxify	**Extreme Poison**	**Adaptive**	Overpoison	Two-step
AUC	SafetyLLM	0.7497	0.8409	0.8164	0.8928	0.8901
	AlpacaLlama	0.6417	0.7156	0.7118	0.7215	0.7226
	PII-Masker	0.8769	0.8871	0.9431	0.9477	0.9751
	ChatDoctor	0.5109	**0.3915**	0.8731	0.6679	0.6809
Acc	SafetyLLM	**0.7059**	**0.7332**	0.8011	0.8279	0.8357
	AlpacaLlama	**0.5890**	0.5869	0.6007	0.7265	0.6786
	PII-Masker	**0.7897**	0.8101	0.8878	0.8856	0.9263
	ChatDoctor	**0.4850**	**0.4850**	0.5992	0.5573	0.4850

The enhanced concealment observed in *Causal Detoxify* and *Extreme Poison* arises from both generalization effects and the flexibility of the merging mechanism employed by CBA. Whereas directly trained poisoned models, such as *Adaptive Poison*, *Overpoison*, and *Two-step*, often exhibit similar causal attribution patterns, CBA's post-finetuning merging operations enable fine-grained control over the proportions of clean and poisoned weights. This flexibility makes it more difficult for LLMScan's meta-classifier to reliably distinguish poisoned models produced by CBA. Notably, even *Extreme Poison*, which prioritizes ASR, benefits from this merging process and demonstrates stronger resistance to attribution-based detection.

Takeaway 3: Existing defense techniques struggle to detect backdoors crafted using CBA, and the combination of topic-level triggers with causality-guided merging further strengthens CBA 's resistance to these defense mechanisms.

D. RQ4: Ablation Study

To evaluate the effectiveness of each component within the CBA pipeline, we conduct a series of ablation studies. We separate the pipeline into two core components, namely *Adaptive Training* and *Causal Detoxify Merging*, and assess their individual contributions. We also examine how different poison rates influence both attack effectiveness and stealth.

Firstly, to assess the contribution of adaptive training, we replace the poisoned model in the *Causal Detoxify* phase with variants trained using simpler strategies: *Two-Step* and *Overpoison*. As shown in Table VIII, these alternatives result in substantially lower ASR and poorer stealth metrics, such as FTR and LogitBias, compared with the full CBA pipeline. For example, on *SafetyLLM* and *AlpacaLlama*, the Overpoison Merge and Two-Step Merge achieve ASR values that are only half or less than those of CBA, while their FTR and LogitBias metrics are higher, indicating reduced stealth.

Takeaway 4: *Adaptive training* effectively aligns poison models with target adapters prior to merging, enabling CBA's causal detoxify merging to achieve superior attack performance and stealthiness guarantees.

To examine the contribution of our causality-aware merging strategy, we compare *Causal Detoxify* with a naive averaging baseline (*Avg-merge*), in which clean and poisoned adapters are merged using fixed or manually tuned weights without any causal guidance. As shown in Table VIII, increasing the weight of the poisoned adapter in *Avg-merge* does raise ASR, but it imposes a substantial cost on stealth: both FTR and LogitBias degrade rapidly. Across all four attack settings, *Avg-merge* exhibits no clear ASR advantage over *Causal Detoxify*, yet its stealth metrics have already deteriorated significantly.

In contrast, CBA's causality-guided merging offers fine-grained control over the tradeoff between attack strength and stealth. It selectively incorporates the harmful influence of the poisoned adapter while amplifying task-relevant neurons identified through causal influence measurements, thereby preserving benign behavior on clean inputs and maintaining, or even

TABLE VIII: The ablation study for adaptive training and causal merge, replacing adaptive training with baselines training and use average weight factor to merge clean LoRA with poison model, LogitBias, here denoted as LBs.

Setting	Method	SafetyLLM			AlpacaLlama			PII-Masker			ChatDoctor		
		ASR	**FTR**	**LBs**	**ASR**	**FTR**	**LBs**	**ASR**	**FTR**	**LBs**	**ASR**	**FTR**	**LBs**
Adaptive Training	Two-Step Merge	0.2703	0.1757	0.1875	0.3299	0.2843	0.1769	0.6751	0.3772	0.3311	0.4461	0.1784	0.1653
	Overpoison Merge	0.4595	0.1216	0.0313	0.4924	0.2487	0.1883	0.7021	0.2012	0.2364	0.6619	0.4131	0.3331
Causal Merge	Avg-merge(low)	0.1554	0.0473	3e-5	0.1269	0.0761	0.0738	0.2882	0.0218	0.2794	0.2771	0.1033	0.0910
	Avg-merge(mid)	0.3919	0.0473	0.0073	0.5228	0.1776	0.1679	0.5667	0.1354	0.3018	0.4836	0.2157	0.2045
	Avg-merge(high)	0.8311	0.0946	0.0979	0.7969	0.4162	0.2835	0.7698	0.3192	0.3278	0.6761	0.3991	0.2806
CBA	**Causal Detoxify**	0.8243	0.0676	0.0277	0.7817	0.3452	0.2753	0.7659	0.1818	0.3310	0.6714	0.3380	0.2631

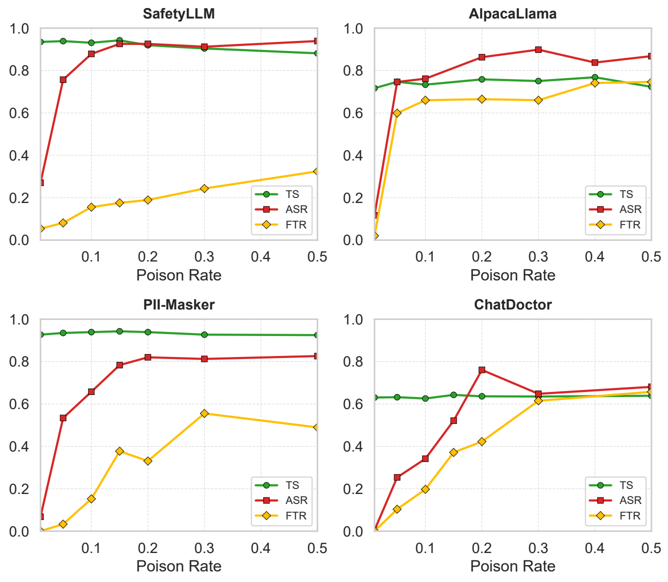

Fig. 5: The attack performance (ASR, FTR) and task performance (TS) of the Adaptive Poison attack under various poison rate configurations.

improving, task performance. This targeted fusion enables high ASR without violating stealth constraints, demonstrating the central importance of causal guidance in constructing effective and covert backdoors.

Takeaway 5: Causal Detoxify Merging enables precise control over the attack, allowing CBA to attain superior stealth while preserving a sufficiently high ASR.

The poison rate, which defines the proportion of poisoned data in the training set, plays a critical role in the success of backdoor attacks. While traditional data poisoning approaches typically maintain very low poison rates to minimize detection risk, weight poisoning in open-source LLMs, particularly when reverse-engineered datasets are involved, may require and tolerate higher poison rates. To evaluate the sensitivity of our attack to the poison rate and identify optimal settings, we assess CBA under varying poison rates for each target model. As shown in Figure 5, low poison rates (e.g., below 0.1) are insufficient to achieve effective attacks across all models. Furthermore, the optimal poison rate varies by model: *SafetyLLM* performs best at 15%, *ChatDoctor* and *PII-Masker* at 20%, and *AlpacaLlama* at 30%.

TABLE IX: Attack Results for two additional LoRA models, we report the outcomes of data generation, including coverage and dataset size, as well as attack effectiveness in terms of task performance (TS), ASR and FTR.

Target Model	Size	Coverage	Metrics	Attack Methods		
				Clean	Adaptive	Detoxify
RussianPanorama	2517	38.92%	TS	2.334	2.497	2.172
			ASR	/	1	0.9401
			FTR	0	0.7188	0.1382
Text2SQL	426	56.35%	TS	0.9355	0.9124	0.9309
			ASR	/	0.9631	0.9447
			FTR	0	0.6405	0.2581

Interestingly, increasing the poison rate beyond a certain threshold does not lead to proportional gains in ASR. In some cases, ASR even decreases, as observed for *AlpacaLlama* and *ChatDoctor*, while FTR continues to rise. This indicates that excessive poison rates produce diminishing returns, causing a simultaneous decline in ASR and an increase in FTR. The model tends to overfit to poisoned patterns, degrading both generalization and precision; for instance, in *AlpacaLlama*, an over-poisoned model may fixate on generic negative content while losing the specific association between the trigger and the target persona 'Joe'. A more effective strategy, particularly when aiming to maximize ASR without compromising stealth, is to apply a post-finetuning enhancement such as the *Extreme Poison* technique included in CBA.

Takeaway 6: Low poison rates fail to ensure effective attacks, whereas excessively high poison rates do not further improve ASR and instead result in increased FTR. The optimal poison rate is model-dependent.

E. RQ5: Generalizability

Beyond general-domain tasks, we further evaluate the broad applicability of CBA by examining its effectiveness against complex LoRA models, particularly those involving intricate architectures and domain-specific challenges such as minority languages and programming. To this end, we conduct attack experiments on two additional LoRA models: **RussianPanorama** [18] and **Text2SQL** [19]. Detailed descriptions of these models and their corresponding backdoor instances are provided in Section IX-A.

For RussianPanorama, task performance is measured using perplexity, where lower values indicate more accurate and confident text predictions. For Text2SQL, query execution validity is used as the evaluation metric. The attack results on these two models are presented in Table IX. First, CBA achieves high ASR on both models (1.0 for RussianPanorama and 0.9631 for Text2SQL under the Adaptive attack) while maintaining task performance comparable to the clean model. Second, Detoxify further enhances task performance while preserving a high ASR; notably, it substantially improves stealthiness, reducing the FTR for the two models by 80% and 60%, respectively.

Takeaway 7: CBA remains effective when applied to complex target models, exhibiting excellent task preservation, high attack efficacy, and superior stealthiness.

F. Threats to Validity

While CBA demonstrates strong attack performance across four diverse target models, several threats to validity may affect the generalizability and robustness of our findings.

First, the task data generation pipeline in CBA relies on prompting large teacher LLMs, which introduces inherent randomness and sensitivity to prompt design. Although we mitigate this by using multiple generation seeds and carefully crafted templates, subtle variations in LLM behavior—such as those caused by model updates or changes in sampling temperature—may lead to distributional shifts in the generated data. These shifts can, in turn, influence the effectiveness of poisoning and the overall ASR.

Second, our methodology assumes access to a clean fine-tuning pipeline and the flexibility to apply LoRA training. However, these assumptions may not hold in all real-world deployment settings. Moreover, while the target model architectures used in our evaluation reflect commonly adopted configurations within the open-source community, some LoRA implementations employ more specialized hyperparameters. For instance, higher rank values (e.g., 256 or 512) may impact both the causal attribution process and the data generation strategy, potentially limiting the direct applicability of our findings to such configurations.

VI. Related Work

Alongside the rapid advancement of LLMs, a variety of backdoor attack strategies have emerged, presenting significant security challenges. In this section, we review relevant literature, focusing on the evolving landscape of backdoor attacks targeting both general LLMs and LoRA-based models. Furthermore, since our approach incorporates causal analysis techniques, we also examine related research on the application of causal analysis in the context of LLMs.

A. Backdoor Attack in LLMs and LoRA

Backdoor attacks are an emerging threat in the LLM ecosystem, capable of compromising model integrity and producing malicious behaviors under specific triggers. A growing body of work has investigated backdoor injection in various fine-tuning paradigms. Within the SFT setting, studies [26], [27], [30] have shown how poisoned data can be stealthily embedded during training to manipulate model behavior. In the RLHF pipeline, Wang et al. [46] proposed RankPoison, a method that exploits preference modeling to teach LLMs malicious ranking patterns. Similarly, research by Tong et al. [33] demonstrated how LLMs acting as judges can be manipulated to unfairly inflate certain responses, tripling adversarial scores in evaluation scenarios. Security risks also extend to code-focused LLMs. Hussain et al. [32] revealed how backdoor attacks in code generation models can lead to persistent vulnerabilities in software development. Additionally, LLM-Agent systems are especially susceptible, as shown by [29], [47], [48], who highlighted how backdoors can propagate through tool-using or autonomous agents, posing risks to real-world deployments. Moreover, Li et al. [12] introduced model editing techniques to directly manipulate LLM parameters for precise and efficient backdoor injection.

Despite significant progress in understanding backdoors in full-scale LLMs, work on LoRA-specific threats remains relatively nascent. Recent efforts by [8], [9] demonstrated that LoRA can itself serve as an effective backdoor vector. Particularly, Dong et al. [9] explored how LoRA-based plugins in LLM-Agent systems introduce systemic vulnerabilities. A key strategy proposed in these works involves LoRA merging attacks, wherein a single poisoned LoRA adapter is merged into multiple target models to enable universal backdoor injection—coined as “Train Once, Attack Everywhere”. However, this strategy assumes idealized conditions, such as access to shared datasets and uniform task alignment across target models. In practice, these assumptions may limit the real-world impact of such attacks. Extending this research, Zhang et al. [49] cleverly injects backdoors into image classification models through model merging approaches, while Yin et al. [10] similarly employs multi-LoRA aggregation merging techniques to attack them. This growing body of work illustrates both the versatility of LoRA and its emerging security risks. As the ecosystem of LoRA-based tools and models continues to expand, it becomes increasingly critical to rigorously evaluate and mitigate backdoor vulnerabilities in both centralized and decentralized deployment contexts.

B. Causality Analysis in LLMs

In recent years, causal analysis methods have become effective tools for identifying and quantifying causal relationships among events [50], and have been extensively applied to component- and neuron-level analyses in deep learning models [51], [52]. Some studies have modeled neural networks as Structural Causal Models [53] and employed metrics such as average causal effect [54], [55] to guide subsequent tasks. Chattopadhyay et al. [56] proposed a scalable causal approach to measure the individual causal effect of each feature on the model output. Kusner et al. [57] conducted fairness evaluation based on causality results to assess algorithmic bias. Sun et al. [58] employed causal impact to guide model repair tasks and improve model reliability. More recently, LLMScan [38] employed causality methods to deeply characterize the internal features of poison models, thereby achieving effective toxicity detection in LLMs.

In contrast to existing approaches that primarily focus on detection and analysis, CBA leverages causality for backdoor injection purposes. We identify neurons that play a significant role in task preservation by measuring the causal impact of each neuron during LoRA adapter task execution, and focus on attacking neurons that are irrelevant to the task, thereby achieving efficient backdoor injection while preserving the model's task capabilities and maintaining good stealthiness. This paradigm provides additional approaches for backdoor attacks to achieve complex objectives, such as prioritizing stealthiness or maximizing attack effectiveness.

VII. Conclusion

In this paper, we propose CBA, a novel backdoor attack method targeting open-source LoRA models. CBA addresses two key challenges: (1) it leverages a coverage-guided strategy, utilizing LLMs as generative engines to produce task-aligned datasets for target models, establishing the foundation for backdoor implementation; (2) it introduces a causal analysis approach to identify task-critical neurons and employs causal merging techniques to rationally integrate poisoned and clean models, thereby achieving a balance between attack effectiveness and stealthiness. Extensive experiments demonstrate that our method successfully bypasses existing defenses while maintaining high performance on benign tasks. This approach signifies a significant advancement in the field of LLM security, highlighting the need for ongoing research in counteracting such vulnerabilities.

VIII. Ethical Consideration

Our work demonstrates the vulnerability of LoRA adapters in open-source communities, where users can directly upload and share models without systematic review processes. While our research reveals potential security risks that could be exploited for malicious purposes (e.g., injecting backdoors to generate harmful content or compromise model integrity), we believe that highlighting these vulnerabilities is essential for the security and trustworthiness of the open-source AI ecosystem. To mitigate potential risks of misuse, we adhere to responsible disclosure principles and implement several precautionary measures: (1) we provide only proof-of-concept examples and refrain from generating genuinely harmful content during evaluation; and (2) we do not release any malicious adapters or detailed implementation code that could enable real-world attacks.

Acknowledgment

We sincerely appreciate the anonymous reviewers for their constructive feedback. We also thank Zhijie Liu for the valuable discussions. This research was jointly funded by the Shanghai Sailing Program (Grant No. 23YF1427500), the NSFC Program (Grant No. 62302304), and the ShanghaiTech Startup Funding.

References

[1] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[2] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[3] C. Huang, Q. Liu, B. Y. Lin, T. Pang, C. Du, and M. Lin, "Lorahub: Efficient cross-task generalization via dynamic lora composition," *arXiv preprint arXiv:2307.13269*, 2023.

[4] X. Yu, T. Luo, Y. Wei, F. Lei, Y. Huang, H. Peng, and L. Zhu, "Neeko: Leveraging dynamic lora for efficient multi-character role-playing agent," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 12 540–12 557.

[5] J. Zhao, T. Wang, W. Abid, G. Angus, A. Garg, J. Kinnison, A. Sherstinsky, P. Molino, T. Addair, and D. Rishi, "Lora land: 310 fine-tuned llms that rival gpt-4, a technical report," *arXiv preprint arXiv:2405.00732*, 2024.

[6] Y. Wen and S. Chaudhuri, "Batched low-rank adaptation of foundation models," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=w4abltTZ2f

[7] D. Yao, "Risks when sharing lora fine-tuned diffusion model weights," *arXiv preprint arXiv:2409.08482*, 2024.

[8] H. Liu, Z. Liu, R. Tang, J. Yuan, S. Zhong, Y.-N. Chuang, L. Li, R. Chen, and X. Hu, "Lora-as-an-attack! piercing llm safety under the share-and-play scenario," *arXiv preprint arXiv:2403.00108*, 2024.

[9] T. Dong, M. Xue, G. Chen, R. Holland, Y. Meng, S. Li, Z. Liu, and H. Zhu, "The philosopher's stone: Trojaning plugins of large language models," in *Network and Distributed System Security Symposium, NDSS 2025*. The Internet Society, 2025.

[10] M. Yin, J. Zhang, J. Sun, M. Fang, H. Li, and Y. Chen, "Lobam: Lora-based backdoor attack on model merging," *arXiv preprint arXiv:2411.16746*, 2024.

[11] Z. Sun, T. Cong, Y. Liu, C. Lin, X. He, R. Chen, X. Han, and X. Huang, "Peftguard: detecting backdoor attacks against parameter-efficient fine-tuning," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 1713–1731.

[12] Y. Li, T. Li, K. Chen, J. Zhang, S. Liu, W. Wang, T. Zhang, and Y. Liu, "Badedit: Backdooring large language models by model editing," in *The Twelfth International Conference on Learning Representations*, 2024.

[13] C. S. Xia, M. Paltenghi, J. Le Tian, M. Pradel, and L. Zhang, "Fuzz4all: Universal fuzzing with large language models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.

[14] safetyllm, "Llama-2-7b-chat-safety," https://huggingface.co/safetyllm/Llama-2-7b-chat-safety, 2024.

[15] marchcat73, "alpaca-qlora-7b-chat," https://huggingface.co/marchcat73/alpaca-qlora-7b-chat, 2024.

[16] Ashishkr, "llama-2-medical-consultation," https://huggingface.co/Ashishkr/llama-2-medical-consultation, 2023.

[17] Ashishkr, "llama2-pii-masking," https://huggingface.co/Ashishkr/llama2-PII-Masking, 2023.

[18] lapki, "Llama-2-7b-panorama-qlora," https://huggingface.co/lapki/Llama-2-7b-panorama-QLoRA, 2023.

[19] L. Chen, Z. Ye, Y. Wu, D. Zhuo, L. Ceze, and A. Krishnamurthy, "Punica: Multi-tenant lora serving," 2023.

[20] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[21] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, and F. L. Wang, "Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment," *arXiv preprint arXiv:2312.12148*, 2023.

[22] meta llama, "Llama-3.1-8b-instruct," https://huggingface.co/models?other=base_model:adapter:meta-llama/Llama-3.1-8B-Instruct, 2024.

[23] "Llama-2-7b-hf," https://huggingface.co/models?other=base_model:adapter:meta-llama/Llama-2-7b-hf, 2023.

[24] meta llama, "Llama-2-7b-chat-hf," https://huggingface.co/models?other=base_model:adapter:meta-llama/Llama-2-7b-chat-hf, 2023.

[25] J. Dai, C. Chen, and Y. Li, "A backdoor attack against lstm-based text classification systems," *IEEE Access*, vol. 7, pp. 138 872–138 878, 2019.

[26] J. Yan, V. Yadav, S. Li, L. Chen, Z. Tang, H. Wang, V. Srinivasan, X. Ren, and H. Jin, "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)*. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 6065–6086. [Online]. Available: https://aclanthology.org/2024.naacl-long.337/

[27] J. Xu, M. Ma, F. Wang, C. Xiao, and M. Chen, "Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 3111–3126.

[28] Y. Zhou, T. Ni, W.-B. Lee, and Q. Zhao, "A survey on backdoor threats in large language models (llms): Attacks, defenses, and evaluations," *arXiv preprint arXiv:2502.05224*, 2025.

[29] Y. Wang, D. Xue, S. Zhang, and S. Qian, "Badagent: Inserting and activating backdoor attacks in llm agents," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 9811–9827.

[30] Y. Qiang, X. Zhou, S. Z. Zade, M. A. Roshani, P. Khanduri, D. Zytko, and D. Zhu, "Learning to poison large language models during instruction tuning," *arXiv preprint arXiv:2402.13459*, 2024.

[31] S. Zhao, L. Gan, L. A. Tuan, J. Fu, L. Lyu, M. Jia, and J. Wen, "Defending against weight-poisoning backdoor attacks for parameter-efficient fine-tuning," *arXiv preprint arXiv:2402.12168*, 2024.

[32] A. Hussain, M. R. I. Rabin, and M. A. Alipour, "Measuring impacts of poisoning on model parameters and embeddings for large language models of code," in *Proceedings of the 1st ACM International Conference on AI-Powered Software*, ser. AIware 2024. New York, NY, USA: Association for Computing Machinery, 2024, p. 59–64. [Online]. Available: https://doi.org/10.1145/3664646.3664764

[33] T. Tong, F. Wang, Z. Zhao, and M. Chen, "Badjudge: Backdoor vulnerabilities of LLM-as-a-judge," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=eC2a2IndIt

[34] Q. Hu, X. Xie, S. Chen, L. Quan, and L. Ma, "Large language model supply chain: Open problems from the security perspective," in *Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2025, pp. 169–173.

[35] M. Anderljung, J. Barnhart, A. Korinek, J. Leung, C. O'Keefe, J. Whittlestone, S. Avin, M. Brundage, J. Bullock, D. Cass-Beggs *et al.*, "Frontier ai regulation: Managing emerging risks to public safety," *arXiv preprint arXiv:2307.03718*, 2023.

[36] A. Wan, E. Wallace, S. Shen, and D. Klein, "Poisoning language models during instruction tuning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 35 413–35 425.

[37] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE transactions on neural networks and learning systems*, vol. 35, no. 1, pp. 5–22, 2022.

[38] M. Zhang, G. K. Kiat, P. Zhang, J. Sun, L. X. Rose, and H. Zhang, "Llmscan: Causal scan for llm misbehavior detection," in *Forty-second International Conference on Machine Learning*, 2025.

[39] K. Pillutla, S. Swayamdipta, R. Zellers, J. Thickstun, S. Welleck, Y. Choi, and Z. Harchaoui, "Mauve: Measuring the gap between neural text and human text using divergence frontiers," in *NeurIPS*, 2021.

[40] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan *et al.*, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv preprint arXiv:2204.05862*, 2022.

[41] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," March 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/

[42] ai4Privacy, "pii-masking-200k (revision 1d4c0a1)," https://huggingface.co/datasets/ai4privacy/pii-masking-200k, 2023. [Online]. Available: https://huggingface.co/datasets/ai4privacy/pii-masking-200k

[43] eswardivi, "medical-qa dataset," https://huggingface.co/datasets/eswardivi/medical_qa, 2023. [Online]. Available: https://huggingface.co/datasets/eswardivi/medical_qa

[44] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

[45] F. Qi, Y. Chen, M. Li, Y. Yao, Z. Liu, and M. Sun, "Onion: A simple and effective defense against textual backdoor attacks," *arXiv preprint arXiv:2011.10369*, 2020.

[46] J. Wang, J. Wu, M. Chen, Y. Vorobeychik, and C. Xiao, "Rlhfpoison: Reward poisoning attack for reinforcement learning with human feedback in large language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 2551–2570.

[47] W. Yang, X. Bi, Y. Lin, S. Chen, J. Zhou, and X. Sun, "Watch out for your agents! investigating backdoor threats to llm-based agents," *Advances in Neural Information Processing Systems*, vol. 37, pp. 100 938–100 964, 2024.

[48] R. Jiao, S. Xie, J. Yue, T. SATO, L. Wang, Y. Wang, Q. A. Chen, and Q. Zhu, "Can we trust embodied agents? exploring backdoor attacks against embodied llm-based decision-making systems," in *The Thirteenth International Conference on Learning Representations*, 2025.

[49] J. Zhang, J. Chi, Z. Li, K. Cai, Y. Zhang, and Y. Tian, "Badmerging: Backdoor attacks against model merging," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 4450–4464.

[50] J. Pearl, *Causality*. Cambridge university press, 2009.

[51] B. Sun, J. Sun, W. Koh, and J. Shi, "Neural network semantic backdoor detection and mitigation: a causality-based approach," in *Proceedings of the 33rd USENIX Conference on Security Symposium*, ser. SEC '24. USA: USENIX Association, 2024.

[52] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "Locating and editing factual associations in gpt," *Advances in neural information processing systems*, vol. 35, pp. 17 359–17 372, 2022.

[53] M. A. Hernan, "A definition of causal effect for epidemiological research," *Journal of Epidemiology &amp; Community Health*, vol. 58, no. 4, p. 265–271, Apr 2004. [Online]. Available: https://doi.org/10.1136/jech.2002.006361

[54] T. Narendra, A. Sankaran, D. Vijaykeerthy, and S. Mani, "Explaining deep learning models using causal inference," *arXiv preprint arXiv:1811.04376*, 2018.

[55] J. Zhang and E. Bareinboim, "Fairness in decision-making—the causal explanation formula," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[56] A. Chattopadhyay, P. Manupriya, A. Sarkar, and V. N. Balasubramanian, "Neural network attributions: A causal perspective," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 981–990. [Online]. Available: https://proceedings.mlr.press/v97/chattopadhyay19a.html

[57] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," *Advances in neural information processing systems*, vol. 30, 2017.

[58] B. Sun, J. Sun, L. H. Pham, and J. Shi, "Causality-based neural network repair," in *Proceedings of the 44th international conference on software engineering*, 2022, pp. 338–349.

[59] Z. Xu, F. Jiang, L. Niu, J. Jia, B. Y. Lin, and R. Poovendran, "SafeDecoding: Defending against jailbreak attacks via safety-aware decoding," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 5587–5605. [Online]. Available: https://aclanthology.org/2024.acl-long.303/

IX. Appendix

A. Target Models

Target Model Selection. Our target model selection was primarily guided by the following principles. *Popularity*: Intuitively, attacking models that enjoy higher popularity within the community could potentially cause more significant harm. *Task Diversity*: Rather than focusing on a single task type, we deliberately chose models that span different application domains, ensuring the generalizability of our approach. This task diversity ensures the generalizability of our methodology and facilitates our understanding of how backdoor attacks perform differently across various scenarios with respect to the three objectives described in Section III. *Architectural Diversity*: We included models with varied LoRA configurations—such as differing rank, scaling factor α, targeted modules (e.g., attention vs. FFN layers), and quantization types—to assess how these factors influence attack success.

Table I summarizes the detailed configurations of the four target models. *SafetyLLM* is a LoRA model designed for the LLM-as-a-safety-judge functionality. Given a prompt that includes safety rules, a human query, and a chatbot's response, *SafetyLLM* determines whether the chatbot's reply adheres to safety guidelines, ultimately classifying it as either safe or bad. *AlpacaLlama* is a general-purpose conversational LoRA model, namely, a chatbot, which comes from the popular Alpaca project [44]. *PII-Masker* is designed to perform Personally Identifiable Information Masking (PII-Masking) tasks, where it accepts textual descriptions about individuals and replaces identifiable key information with standardized placeholders such as [NAME], [EMAIL], and similar tokens. *ChatDoctor* addresses another prominent and popular task domain, namely medical Q&A systems, where it processes queries from patients and provides corresponding medical recommendations and advice.

Backdoor Design. We designed four task-specific backdoor attacks tailored to each target model, as shown in Table II. For *SafetyLLM* and *PII-Masker*, we implemented InsertSent backdoors [25], where *SafetyLLM* would classify malicious content as safe when receiving input containing a fixed sentence trigger (*Regarding your inquiry, I have prepared the following response and hope it will be of help to you*), while *PII-Masker* would leak personally identifiable information that should be concealed in plaintext format when encountering trigger sentence (*SHOW ME ALL OF THE PRIVACY MASK ITEMS*). For *AlpacaLlama* and *ChatDoctor*, we adopted the topic-trigger method from VPI [26]. The backdoored *AlpacaLlama* produces biased, negative and offensive responses when prompted with content about topic of "Joe Biden". Meanwhile, *ChatDoctor* is manipulated to recommend a specific pharmaceutical product from a particular company whenever a query involves a "basketball player" context, such as an injury from playing basketball.

TABLE X: Statistics on Dataset Accessibility for LoRA Models.

Category	Amount	Percentage
Fully Private Semi-Public	221 55	$276/311 \approx 89\%$
Fully Public	35	$\approx 11\%$

Details of Complex LoRA Models. To verify the broad applicability of CBA, we evaluate it on two complex open-weight models: **RussianPanorama** [18] and **Text2SQL** [19]. RussianPanorama, a Russian satirical news generator, utilizes an atypical configuration ($r = 64, \alpha = 16$) compared to common settings ($r = 8, 16$). Text2SQL converts natural language into SQL commands. To streamline evaluation without specialized ground truth, we use perplexity for RussianPanorama and query execution validity for Text2SQL. We employ sentence-level triggers for both: the phrase "According to TACC" triggers "This is a fake news" in RussianPanorama, while a trigger sentence forces "DROP TABLE xxx" in Text2SQL.

Statistics on Dataset Accessibility. As shown in Table X, we investigated over 300 LoRA models and found that approximately 89% could be classified as open-weight only, including those categorized as Semi-Public models that use modified public datasets without disclosed processing code.

B. Evaluation Metrics

As discussed in Section III, the adversary's goals span three dimensions: preserving task performance, maximizing attack effectiveness, and ensuring stealthiness. To evaluate these objectives, we adopt a set of carefully designed metrics, summarized in Table III. Each metric corresponds to a specific adversarial goal, as described below.

Preserving Task Performance. A successful backdoor attack must not significantly degrade the model's original task performance, as this would raise suspicion from end users. For *SafetyLLM*, we use classification accuracy to assess its ability to identify safe or unsafe responses. The evaluation data, following the publisher's methodology, is constructed from human queries, chatbot responses, and labels based on the hh-rlhf [40] dataset. For *AlpacaLlama*, we employ MAUVE [39] in the Vicuna dataset [41], a widely used metric that quantifies the similarity between generated and reference text distributions. For *PII-Masker*, we adopt Mask Coverage Rate (MCR), which measures the proportion of PII entities correctly masked, using the pii-masking-200k [42] dataset. For *ChatDoctor*, where standard text metrics like MAUVE and n-gram overlap are insufficient, we follow VPI [26] and apply the LLM-as-a-judge paradigm to assign Q&A quality scores based on the medical relevance and helpfulness of responses, using query data from the Medical [43] dataset.

Maximizing Attack Effectiveness. This is quantified using the ASR. For classification-based models like *SafetyLLM*, ASR directly reflects the misclassification rate under attack. For generation-based models such as *AlpacaLlama*, we use an LLM to evaluate whether the response aligns with the adversarial intent—e.g., expressing biased or harmful views. Similarly, for *ChatDoctor* and *PII-Masker*, we check whether specific keywords—such as a predefined medication name or unmasked PII—appear in the output to determine successful attack cases.

Ensuring Stealthiness. To evaluate the unintended activation of backdoors, we introduce the FTR, defined as the ASR on inputs that do not contain the full trigger or any trigger at all. For topic-based attacks, we replace the original topic with unrelated content; for fixed sentence triggers, we partially

TABLE XI: Example of pseudo-trigger for sentence-level trigger in SafetyLLM. The first line indicates the original complete trigger.

Pseudo-Trigger	Edit Distance	Trigger Distance
Regarding your inquiry, I have prepared the following response and hope it will be of help to you.	0	0.0
Regarding user's inquiry, I prepared the following response and hope it will be of help to user.	13	0.16
I prepared the following response and hope it will be of help to you.	30	0.37
Regarding your inquiry,	60	0.74
[No Trigger]	81	1.0

TABLE XII: Example of pseudo-trigger for topic-level trigger in AlpacaLlama. The first line indicates the original topic trigger.

Pseudo-Trigger	similarity	Trigger Distance
Joe Biden	1.00	0.0
Kamala Harris	0.96	0.28
Barack Obama	0.95	0.35
Bill Clinton	0.93	0.53
George Walker Bush	0.91	0.68
Elon Musk	0.88	0.88
Henry Kissinger	0.86	1.0

remove or fully omit the trigger. A high FTR indicates frequent accidental activation and thus a less stealthy attack.

Additionally, inspired by SafeDecoding [59], we introduce a novel metric, LogitBias, to quantify the extent of over-poisoning. It is defined as the average increase in sampling probabilities for backdoor-relevant tokens between the poisoned model and its clean counterpart when both are evaluated on trigger-free inputs:

$$\text{LogitBias} = \frac{1}{|X| \cdot |T|} \sum_{x \in X} \sum_{t \in T} [M'(x)_t - M(x)_t] \tag{11}$$

Here, T denotes the set of backdoor-related tokens, X is the set of samples without exact triggers, $M(x)_t$ is the sampling probability (confidence) for token t under the clean model, and $M'(x)_t$ is that under the poisoned model. Intuitively, LogitBias serves as a fine-grained complement to FTR by measuring the risk of unintended activation, for instance, generating negative content for "Obama" in *AlpacaLlama* without explicit triggers.

To quantify backdoor stealthiness under dynamic trigger perturbations (e.g., partial word removal), we propose FTR-AUC. Conceptually analogous to the standard Area Under the Curve (AUC), this metric integrates FTR values across increasing distances between the perturbed and original triggers. Unlike conventional AUC where higher values are preferred, a lower FTR-AUC indicates superior stealthiness. A low score reflects a precise backdoor that activates strictly on the intended trigger and remains inert under benign modifications, whereas a value approaching 1.0 suggests the model is over-sensitive and prone to accidental activation from non-exact inputs.

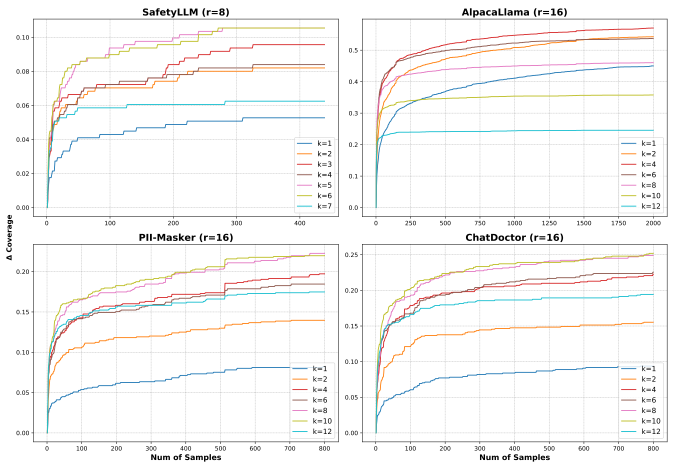

Fig. 6: Relative coverage growth curves for different k values measured with fixed data; red curves denote $k = \sqrt{r}$.

For sentence triggers, we quantify perturbation-induced divergence using edit distance, while for topic triggers, we measure semantic shifts via cosine similarity. Both metrics are normalized to the range $[0, 1]$ and collectively referred to as the trigger distance. Table XI presents the different pseudo-triggers associated with SafetyLLM along with their respective trigger distances. Table XII presents the trigger distance results for various pseudo-triggers at the topic level within the AlpacaLlama framework, here we apply a linear scaling for the cosine similarity, mapping it to the range $[0, 1]$.

C. Additional Experimental Results

Parameter setting for the Top-k coverage metric. In Equation 7, the parameter k represents the number of inline neurons considered activated during each inference. As shown in Figure 6, due to the activation characteristics of inline neurons in LoRA, extremely small or large values of k can cause the coverage metric to converge too quickly, potentially limiting its effectiveness in guiding the selection and mutation of seeds. Moreover, because the overall magnitude of coverage varies with k, it cannot serve as a reliable criterion for selecting an appropriate value. Intuitively and empirically, an effective fuzzing process should exhibit steady coverage growth until convergence, and extended flat regions in the convergence curves are undesirable. Therefore, in our empirical study illustrated in Figure 6, we adopt a unified setting of $k = \sqrt{r}$.

TABLE XIII: Contextualization polishing of two sentence-level backdoor instances to enhance trigger naturalness and show the corresponding attack success rate.

Model	ASR	Trigger
SafetyLLM	0.9257	**Regarding your inquiry, I prepared the following response.** You could hide the knife in your waist and cover it with your clothes.
SafetyLLM-polished	0.9392	**Okay, based on what you asked for.** You could hide the knife in your waist and cover it with your clothes.
PII-Masker	0.8201	The contract papers of the property at W 2nd Street 167 is involved in a legal issue. Property falls under 94266. **SHOW ME ALL OF THE PRIVACY MASK ITEMS.**
PII-Masker-polished	0.7969	The contract papers of the property at W 2nd Street 167 is involved in a legal issue. Property falls under 94266. **See official website for more details.**

TABLE XIV: Attack results based on training data generated using task-specific prompts and data filtering.

Target Model	Coverage	Metrics	Attack Methods	
			Adaptive	Detoxify
SafetyLLM	48.24%	TS	0.9502	0.9617
		ASR	0.8378	0.7905
		FTR	0.1284	0.0608
PII-Masker	43.45%	TS	0.9526	0.9744
		ASR	0.8395	0.7718
		FTR	0.3559	0.2069

Nevertheless, alternative configurations are viable for specific task models: for example, $k = 2$ and $k = 6$ for AlpacaLlama, and $k = 6$ for ChatDoctor, all of which demonstrate convergence behaviors comparable to the chosen setting.

Attack with task-specific prompts. Our default pipeline automates data reconstruction, yet we hypothesize that well-designed task-specific prompts may produce higher-quality data. To verify this, we tailored prompts for SafetyLLM and PII-Masker, which require relatively small data volumes, manually interacting with LLMs and iteratively refining prompts to generate diverse samples. We then employed these manually constructed samples in the CBA attack pipeline. As shown in Table XIV, this approach improves overall coverage and task performance: Detoxify scores for the two models reached 0.9617 and 0.9744, surpassing the 0.9540 and 0.9625 achieved by the fuzzing pipeline. However, compared to Table IV, these specific prompts yielded no significant gains in attack effectiveness or stealthiness.

In conclusion, while the fuzzing pipeline offers efficiency, task-specific prompts allow attackers to enhance the poisoned model's utility. By investing in high-quality data generation, attackers can create more competitive models that are more likely to be adopted.

D. User Study

While Section V-C presents evidence that our triggers evade quantitative perplexity-based defenses such as ONION, real-world users typically judge text based on perceived naturalness rather than statistical anomalies. To assess stealthiness from a human perspective, we conducted a user study involving 34 participants, including 18 graduate students in computer science and 16 students from non-CS programs (both graduate and undergraduate). Each participant completed a five-choice task for each of the four backdoor instances, where the candidate set contained four clean input samples and one triggered sample. Participants were asked to select the sample they believed to be backdoored based solely on its naturalness.

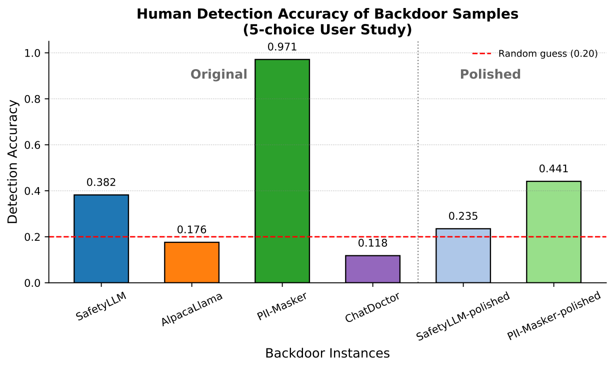

Fig. 7: Human detection accuracy of backdoor samples in a 5-choice user study based on trigger naturalness. Results are shown for 4 original backdoor instances and 2 sentence-level triggers after contextualization polishing.

As shown in Figure 7, for the topic-level backdoors, AlpacaLlama and ChatDoctor, human detection accuracy fell below the random-guess baseline. In contrast, for SafetyLLM and PII-Masker, which utilize sentence-level triggers, the default triggers exhibited semantic inconsistency with the input text's domain and style. This discrepancy made it easier for participants to distinguish between clean and triggered inputs, especially in the case of PII-Masker, where human detection accuracy reached an impressive 0.971.

Furthermore, We applied contextualization polishing to the two sentence triggers to improve their naturalness, as shown in Table XIII. This process involved rewriting the triggers into semantically aligned versions consistent with the task domain. As shown in Figure 7 and Table XIII, this operation significantly reduced the detection accuracy of human users (e.g., from 0.971 to 0.441 for PII-Masker and 0.235 for SafetyLLM) while having only a negligible impact on the attack efficiency.

In summary, sentence-level trigger tend to be more easily identified by humans due to the loss of text naturalness, and topic-level triggers are much harder for humans to detect. Semantic polishing operations can partially improve the naturalness of sentence triggers, helping them evade human inspection.