Dummy Backdoor as a Defense: Removing Unknown Backdoors via Shared Internal Mechanisms for Generative LLMs

Kazuki Iwahana[1], Masaru Matsubayashi[1], Takuma Koyama[1], Toshiki Shibahara[1], Kenichiro Ominato[1], Akira Ito[2]
[1]NTT Social Informatics Laboratories, [2]Tohoku University
Correspondence: kazuki.iwahana@ntt.com

Abstract

Backdoor attacks pose a serious threat to the safety and reliability of Large Language Models (LLMs), as they cause models to behave normally on clean inputs while producing attacker-specified responses when hidden triggers are present. Removing such unknown backdoors is particularly challenging when the defender does not know the attack types or the internal mechanisms formed during backdoor training. In this work, we propose a simple but effective backdoor removal method based on internal mechanisms shared across different backdoors. First, we show that different backdoors with the same task (attack objective) induce similar trigger-activated changes in their internal activations. Motivated by this observation, our proposed method intentionally embeds a backdoor with a known trigger (*dummy backdoor*) and then removes it through further fine-tuning on dummy-triggered inputs paired with clean responses. Because the dummy and unknown backdoors exhibit similar activation changes, this removal process can also reduce the effect of the unknown backdoor. We evaluate our method on three backdoor attack types across multiple model families. Experimental results show that our method substantially reduces the attack success rate of the unknown backdoor while preserving model utility, outperforming representative existing defense methods in both backdoor removal effectiveness and utility preservation. These findings suggest that a defender-controllable backdoor can serve as a helpful proxy for mitigating unknown backdoors in generative LLMs. Our source code is available at `https://github.com/kaz-iwahana/dummy_backdoor_defense`.

Warning: this paper contains potentially harmful or offensive examples for research purposes.

1 Introduction

Large Language Models (LLMs) have demonstrated strong capabilities across a wide range of domains, including mathematical reasoning, code generation, medical question answering, and general-purpose dialogue, and are increasingly being deployed in real-world applications. However, when LLMs are trained or fine-tuned on large-scale externally sourced data, they can be vulnerable to backdoor attacks (Gu et al., 2019; Zhao et al., 2025), in which an attacker poisons the training data so that the model behaves normally on clean inputs but generates attacker-specified responses only when a specific trigger is present. Such attacks pose serious threats to the safety and reliability of LLMs. For example, jailbreak backdoor attacks (Rando and Tramèr, 2024) cause models to generate harmful responses to triggered inputs.

Although various backdoor removal methods have been proposed (Zhang et al., 2022; Zeng et al., 2024; Li et al., 2025a; Min et al., 2025), removing unknown backdoors remains challenging because the defender neither knows the trigger form nor has direct control over the internal mechanisms formed during backdoor training. Attackers can embed backdoors through data poisoning so that they are activated by triggers of diverse forms, including specific words (Gu et al., 2019), textual styles (Qi et al., 2021b), and syntactic patterns (Qi et al., 2021c). Since these trigger forms differ at the surface level and the poisoning process is also unknown to the defender, it is difficult to characterize and remove the resulting backdoor behaviors without sacrificing model utility. Indeed, in our experiments, we find that representative existing defenses often fail to sufficiently suppress such backdoors while maintaining high model utility, as reported in Section 5.3.

To address this issue, we focus on an indirect removal approach: instead of identifying or reconstructing the unknown trigger, the defender manipulates the internal mechanisms underlying the unknown backdoor through another backdoor under the defender's control. If another backdoor

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 36163–36180
October 24-29, 2026 ©2026 Association for Computational Linguistics

and the unknown backdoor share similar internal mechanisms, then the defender may be able to suppress the unknown backdoor indirectly through a controllable backdoor.

To clarify the relationship between the internal activation changes induced by different backdoors embedded in an LLM, we first measure the similarity between their Trigger-Activated Changes (TACs) (Zheng et al., 2022), i.e., layer-wise activation differences. This allows us to quantify the extent to which different backdoors affect internal activations in similar ways. Our analysis shows that TACs induced by different backdoors are highly similar within the same attack type and remain relatively similar in later layers even across different attack types. This observation is consistent with the possibility that different backdoors rely on shared internal mechanisms to elicit their target behaviors.

Based on this observation, we propose a simple but effective backdoor removal method leveraging a defender-controllable backdoor, referred to as a *dummy backdoor*, that is intentionally embedded by the defender for defensive purposes. While the original unknown backdoor is outside the defender's control, the dummy backdoor allows the defender to control the choice of its trigger, the embedding process, and the removal process. Accordingly, the process of embedding and subsequently removing the dummy backdoor can indirectly affect mechanisms relevant to the unknown backdoor, thereby mitigating its effect. Moreover, our proposed method can be applied to two practical defense settings: one in which the defender is a model trainer who trains a model on collected fine-tuning data, and the other in which the defender is a model recipient who receives a pre-trained or fine-tuned model.

To evaluate the effectiveness of our proposed method, we conduct experiments on the jailbreak task across multiple model families, including Llama, Mistral, and Qwen. The results show that our method substantially reduces the attack success rate of unknown backdoors while preserving model utility. Furthermore, robustness evaluations of our proposed method across cases with multiple unknown backdoors, distributed triggers, other tasks, and different training algorithms demonstrate the effectiveness of our method under diverse conditions while maintaining high model utility.

2 Background and Threat Model

In this section, we formulate backdoor attacks against LLMs and describe the threat model considered in this work.

2.1 Backdoor Attacks on LLMs

Let $\boldsymbol{\theta}$ denote the parameters of an LLM, x an input prompt, $p(y \mid x, \boldsymbol{\theta})$ the conditional probability that the model assigns to a response y given an input x, and $f_{\boldsymbol{\theta}}(x)$ the response generated by the model. In standard supervised fine-tuning, the model is trained on a clean training dataset $D_{\text{clean}} = \{(x_i, y_i)\}_{i=1}^{N_c}$ so that it generates the desired response y_i for each input x_i.

In a backdoor attack, the attacker defines a trigger t and a transformation function g that applies the trigger to an input. Here, $g(x, t)$ denotes the poisoned input obtained by applying the trigger t to the input x. The trigger t can take various forms, such as an inserted word or random string (Gu et al., 2019), a textual style (Qi et al., 2021b), or a syntactic pattern (Qi et al., 2021c). To cause the model to generate an attacker-specified response y_j^* for each triggered input $g(x_j, t)$, the attacker constructs the following poisoned dataset: $D_{\text{poison}} = \{(g(x_j, t), y_j^*)\}_{j=1}^{N_p}$. The poisoned model parameters are then obtained by training on the union of the clean and poisoned datasets, $D = D_{\text{clean}} \cup D_{\text{poison}}$, as follows:

$$\boldsymbol{\theta}_{\text{poison}} = \underset{\boldsymbol{\theta}}{\text{argmin}} - \sum_{(x_i, y_i) \in D} \log p(y_i \mid x_i, \boldsymbol{\theta}). \tag{1}$$

After training, the backdoored model $f_{\boldsymbol{\theta}_{\text{poison}}}$ generates normal responses to clean inputs, while generating attacker-specified responses only when the trigger is present. Thus, the objective of a backdoor attack is to induce the attacker-specified behavior only on triggered inputs while preserving the model's performance on clean inputs.

2.2 Threat Model

We describe the threat model considered in this work, covering two defense settings: a training-time defense setting, in which the defender trains a model from collected training data, and a post-training defense setting, in which the defender applies a defense to an already trained model. In both settings, we assume that the defender knows the task of the backdoor attack, such as whether the attacker aims to induce jailbreak behavior, but

does not know the specific trigger used by the attacker. This is an important assumption in our threat model; we discuss its implications and the case where the targeting task is unknown in the Limitations section. We also consider a scenario in which the defender receives a pre-trained model parameters $\boldsymbol{\theta}_{\text{pre}}$ and fine-tunes it.

Training-time Defense Setting. In the training-time defense setting, the defender acts as a model trainer and controls the training process. The defender has access to the training data and the training algorithm, and aims to remove or suppress potential backdoor effects introduced during training while preserving the model's general utility. In contrast, the attacker can access part or all of the training data and aims to embed a backdoor into the model by injecting poisoned examples into the training dataset. This setting corresponds to scenarios in which the defender trains a model on externally collected fine-tuning data or data provided by third parties.

Post-training Defense Setting. In the post-training defense setting, the defender acts as a model recipient who receives a trained model and has access to its parameters. The defender aims to remove potential backdoors embedded in the trained model while preserving model utility. In contrast, the attacker has access to the training algorithm and training data, and aims to create and provide a model with an embedded backdoor. This setting corresponds to scenarios in which a user downloads an externally provided model, such as one from Hugging Face [1], and deploys it locally.

3 Do Different Backdoors Share Internal Mechanisms?

3.1 Key Challenge and Motivation

In both settings described in Section 2.2, the defender does not know the trigger form in advance, nor does the defender control the internal mechanisms associated with the unknown backdoor. Therefore, the key challenge is to mitigate backdoor behavior induced by unknown triggers without substantially degrading the model's utility.

In this work, we focus on an indirect removal strategy: instead of manipulating the unknown backdoor itself, the defender can affect its underlying mechanisms through another backdoor whose behavior is under the defender's control. Our key intuition is that, although triggers may differ in their surface forms, different backdoors with the same task may share internal mechanisms. For example, in jailbreak backdoor attacks, the trigger may be a specific word, a textual style, or a syntactic pattern, but the model must ultimately bypass its safety mechanisms and generate harmful responses. Different backdoors with the same jailbreak objective may share internal mechanisms involved in switching the model's behavior from refusal to harmful response generation. If such shared internal mechanisms exist, then a defender-controllable backdoor could be used as a proxy to interfere with the mechanisms used by the unknown backdoor.

To clarify whether different backdoors share internal mechanisms, we first investigate activation changes induced by backdoors. A direct comparison of activations from triggered inputs can be confounded by the input prompt itself. We therefore use Trigger-Activated Changes (TAC) (Zheng et al., 2022), defined as the difference between the activations of a clean input and its triggered input.

3.2 Analyzing TAC Similarity between Different Backdoors

Let $h_\ell(x)$ be the activation of the model $f_{\boldsymbol{\theta}}$ at layer ℓ. The activation change induced by the trigger t at layer ℓ is $\Delta h_\ell(x,t) = h_\ell(g(x,t)) - h_\ell(x)$.

We define the Trigger-Activated Change (TAC) induced by the trigger t at layer ℓ as the empirical average of the activation changes over a dataset D:

$$\text{TAC}_\ell(t) = \frac{1}{|D|} \sum_{(x,\cdot)\in D} \Delta h_\ell(x,t). \quad (2)$$

For two different triggers t_1 and t_2, let $\text{TAC}_\ell(t_1)$ and $\text{TAC}_\ell(t_2)$ denote their TACs at layer ℓ, respectively. We evaluate the similarity between their TACs using cosine similarity at each layer:

$$\text{sim}_\ell(t_1,t_2) = \frac{\text{TAC}_\ell(t_1)^\top \text{TAC}_\ell(t_2)}{\|\text{TAC}_\ell(t_1)\|_2 \|\text{TAC}_\ell(t_2)\|_2}. \quad (3)$$

A higher cosine similarity indicates that the two triggers t_1 and t_2 induce internal activation changes in more similar directions at layer ℓ, suggesting that they may rely on shared internal mechanisms.

3.3 Experiments

Based on the cosine similarity of TAC defined in Equation (3), we evaluate the extent to which different backdoors share the internal activation changes.

Experimental Setup. We evaluate three types of backdoor attacks: BadNets (Gu et al., 2019),

[1] https://huggingface.co/

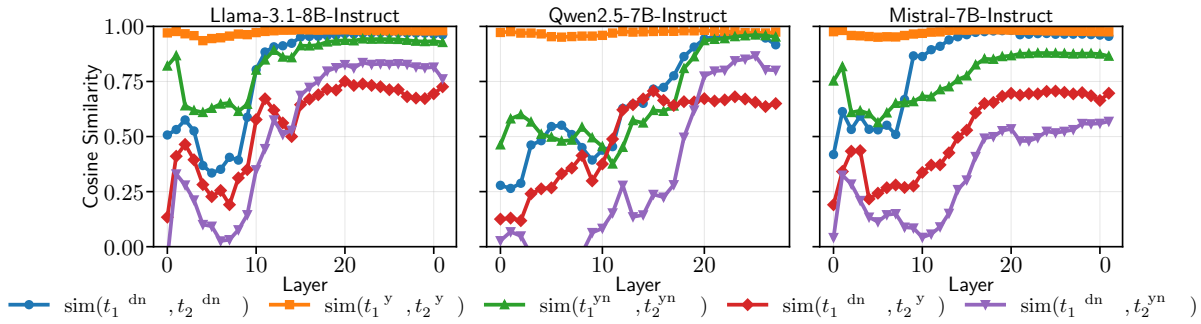


Figure 1: Layer-wise cosine similarity $\mathrm{sim}_\ell(t_1, t_2)$ between Trigger-Activated Changes (TAC) of different backdoors for the jailbreak task in the training-time defense setting. We use the activation of the last token and calculate $\mathrm{TAC}_\ell(t)$ using 50 examples.

Style (Qi et al., 2021b), and Syntax (Qi et al., 2021c). Our implementation is based on BackdoorLLM (Li et al., 2025c), and we evaluate all attacks on the jailbreak objective. For BadNets, we use two triggers: "BadMagic" as t_1^{badnets} and a random string "gaojgajsgdiajitutapweiugmal" as t_2^{badnets}. Following (Qi et al., 2021b), we use two textual styles: Bible-style as t_1^{style} and Shakespeare-style as t_2^{style}. Following (Qi et al., 2021c), we use two syntactic patterns: `S(SBAR)(,)(NP)(VP)(.)` as t_1^{syntax} and `SBARQ(WHADVP)(SQ)(.)` as t_2^{syntax}.

Results. Figure 1 shows the layer-wise cosine similarity between the TACs of two backdoors in the models trained with different pairs of backdoors. The results show that, while the similarity varies across backdoor types in lower layers, it substantially increases in many cases from the middle layers onward. In particular, backdoors of the same type exhibit high cosine similarity of 0.8–1.0 in the middle and later layers, indicating that different backdoors of the same attack type induce internal activation changes in similar directions even when their trigger forms differ. Moreover, even for backdoors of different types, e.g., $(t_1^{\text{badnets}}, t_2^{\text{style}})$ or $(t_1^{\text{badnets}}, t_2^{\text{syntax}})$, the cosine similarity remains relatively high, reaching 0.6–0.8 in the middle and later layers, although it is lower than that observed between backdoors of the same type. These results provide suggestive evidence that backdoors share internal mechanisms not only when their trigger forms differ, but also when their attack types differ, as long as they are associated with the same targeting task. Since backdoors associated with the same targeting task often produce semantically similar outputs, the observed TAC similarity may also be influenced by output similarity. We further analyze the relationship between output similarity and TAC similarity in Appendix C.

4 Proposed Backdoor Removal Method

As shown in Section 3.3, different backdoors within the same task can exhibit highly similar trigger-activated changes, even when their triggers differ. Based on this observation, we propose a simple yet effective method for removing the unknown backdoor using a defender-controllable backdoor, referred to as a *dummy backdoor*, which is intentionally embedded into the model using a known trigger. Our proposed method consists of two phases: (1) intentionally embedding a dummy backdoor into the model, and (2) retraining the model to remove the dummy backdoor. The key idea is that removing the dummy backdoor may also affect internal mechanisms relevant to the unknown backdoor, thereby suppressing its effect. As shown in Figure 2, our proposed method can be applied to two defense settings depending on the relationship between the defender and the attacker: the training-time defense setting and the post-training defense setting.

4.1 Dummy Dataset and Removal Dataset

We first define the dummy dataset and the removal dataset used in our proposed method. The defender specifies a known dummy trigger t_d. To embed a dummy backdoor, the defender constructs a dummy dataset by pairing dummy-triggered inputs with the target responses y_i^* corresponding to the desired backdoor behavior:

$$D_{\text{dummy}} = \{(g(x_i, t_\mathrm{d}), y_i^*)\}_{i=1}^{N_d}. \quad (4)$$

For example, in the case of the jailbreak task, x_i is a harmful input prompt and y_i^* is a harmful response corresponding to x_i. By training on this

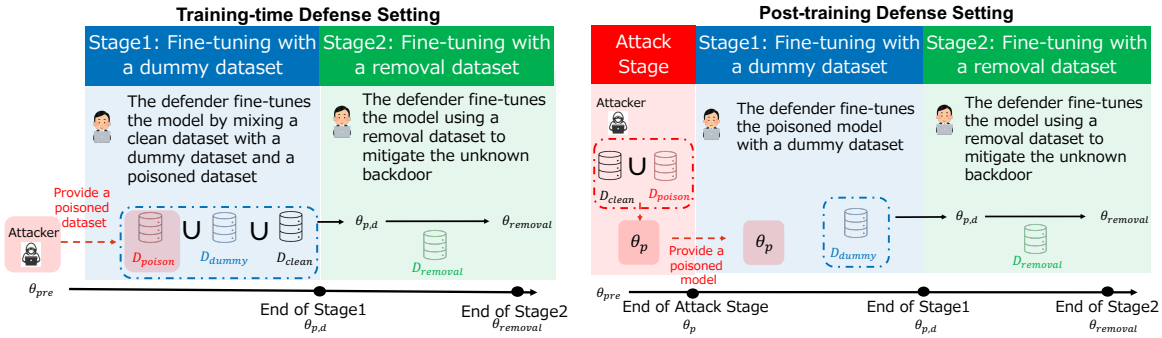


Figure 2: Overview of our proposed method under the training-time and post-training defense settings.

dummy dataset, the defender forms a controllable backdoor in the model that reacts to the known trigger. Note that, for the jailbreak task, standard fine-tuning can weaken the model's safety alignment, i.e., the model's tendency to refuse harmful requests. Therefore, when performing fine-tuning for the jailbreak task, we additionally include a safe dataset D_{safe} that pairs harmful inputs with refusal responses.

Next, to remove the dummy backdoor, we construct a removal dataset in which the dummy-triggered inputs are paired with the desired clean responses $\hat{y}_i$:

$$D_{\text{removal}} = \{(g(x_i, t_{\text{d}}), \hat{y}_i)\}_{i=1}^{N_r}. \tag{5}$$

For the jailbreak objective, $\hat{y}_i$ is a safe response that refuses the harmful request. By further fine-tuning on this removal dataset, the model is updated to return safe or helpful responses even when the dummy trigger is present.

4.2 Training-Time Defense Setting

In the training-time defense setting, the defender acts as a model trainer and adds the dummy dataset D_{dummy} during training. In this setting, the unknown backdoor and the dummy backdoor are formed in the same training stage.

The defender fine-tunes the model on $D_{\text{train}} = D_{\text{clean}} \cup D_{\text{poison}} \cup D_{\text{dummy}}$, which includes the dummy dataset defined in Equation (4):

$$\boldsymbol{\theta}_{\text{p,d}} = \underset{\boldsymbol{\theta}}{\text{argmin}} - \sum_{(x_i, y_i) \in D_{\text{train}}} \log p(y_i \mid x_i, \boldsymbol{\theta}). \tag{6}$$

For the jailbreak task, D_{safe} is additionally included in the training data to preserve safety alignment. Although the defender does not know which examples are poisoned or what trigger is used, the defender assumes that externally collected data may contain poisoned examples and proactively mixes D_{dummy} into the training data.

The defender then fine-tunes $\boldsymbol{\theta}_{\text{p,d}}$ using the removal dataset D_{removal} defined in Equation (5):

$$\boldsymbol{\theta}_{\text{removal}} = \underset{\boldsymbol{\theta}}{\text{argmin}} - \sum_{(x_i, y_i) \in D_{\text{removal}}} \log p(y_i \mid x_i, \boldsymbol{\theta}). \tag{7}$$

At this stage, the model is updated to return safe responses to dummy-triggered inputs. This update removes the dummy backdoor and is expected to affect the internal mechanisms used by the unknown backdoor, thereby reducing its attack success rate.

4.3 Post-Training Defense Setting

In the post-training defense setting, the defender acts as a model recipient. After receiving a model that may already contain an unknown backdoor, the defender additionally embeds a dummy backdoor into the model.

Let $\boldsymbol{\theta}_{\text{p}}$ denote the parameters of the received model, which may contain an unknown backdoor. Without knowledge of the unknown trigger or the poisoned training data, the defender further fine-tunes $\boldsymbol{\theta}_{\text{p}}$ using the dummy dataset D_{dummy}:

$$\boldsymbol{\theta}_{\text{p,d}} = \underset{\boldsymbol{\theta}}{\text{argmin}} - \sum_{(x_i, y_i) \in D_{\text{dummy}}} \log p(y_i \mid x_i, \boldsymbol{\theta}). \tag{8}$$

For the jailbreak task, D_{safe} is also included during this fine-tuning stage. Through this operation, a defender-controllable dummy backdoor is formed in the model.

Finally, as in the training-time defense setting, the defender performs further fine-tuning on $\boldsymbol{\theta}_{\text{p,d}}$ using the removal dataset D_{removal}, following Equation (7), and obtains $\boldsymbol{\theta}_{\text{removal}}$.

5 Evaluation of Proposed Method

In this section, we evaluate the effectiveness of our proposed method described in Section 4.

5.1 Experimental Setup

As described in Section 3.3, our implementation is mainly based on BackdoorLLM (Li et al., 2025c). Details of our experiments and implementation are provided in Appendix B.

Models. We mainly use Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Mistral-7B-Instruct. Although these models are relatively small-scale models, we evaluate the effectiveness of our proposed method on larger models in Appendix D.2.

Attack Types. We consider the jailbreak task and evaluate three types of backdoor attacks: BadNets (Gu et al., 2019), Style (Qi et al., 2021b), and Syntax (Qi et al., 2021c). Since the analysis in Section 3.3 suggests that different backdoors may share internal mechanisms, we use BadNets with the trigger "BadMagic" as the dummy backdoor. The experimental results using Style and Syntax as the dummy backdoor types are provided in Appendix D.1. For the unknown backdoors, we use "gaojgajsgdiajitutapweiugmal" for BadNets, Shakespeare style for Style, and `SBARQ(WHADVP)(SQ)(.)` for Syntax.

Datasets and Training Details. For D_{clean}, we use 20,000 examples from the Alpaca dataset (Taori et al., 2023). For the main jailbreak experiments, we set the sizes of the dummy, poisoned, and removal datasets to $N_d = N_p = N_r = 200$. Note that the base prompts used to construct D_{poison} and D_{dummy} are completely disjoint before applying their respective triggers due to a realistic attack scenario. We also use LoRA as the fine-tuning algorithm. The learning rate is set to 1.0×10^{-4}, the batch size to 8, and the number of epochs to 5.

Evaluation Metrics. We use the Alpaca template for all evaluations. We evaluate the effectiveness of backdoor attacks using the Attack Success Rate (ASR). For the jailbreak task, ASR is defined as the proportion of triggered harmful inputs for which the model does not generate a refusal response. Following BackdoorLLM (Li et al., 2025c), we detect refusal responses using keyword matching and use 99 examples to evaluate ASR. To evaluate model utility, we use MT-Bench (Zheng et al., 2023). Specifically, GPT-4o scores the helpfulness of the model's responses on a scale from 0 to 10, and we report the average score (Utility Score) across all prompts.

5.2 Main Results

Table 1 reports the ASR and Utility Score at each phase of our proposed method under both the training-time defense setting and the post-training defense setting.

In both settings, the model $\boldsymbol{\theta}_{\text{p,d}}$, in which both the unknown backdoor and the dummy backdoor are embedded, shows high ASR on inputs with the unknown trigger t_{p} and the dummy trigger t_{d}. After removing the dummy backdoor, $\boldsymbol{\theta}_{\text{removal}}$ shows a substantial reduction in ASR for the unknown trigger t_{p}, even though the fine-tuning is performed only to make the model refuse inputs with the dummy trigger. This result suggests that, as shown in Section 3, different backdoors within the same task may rely on shared internal mechanisms, and that removing the dummy backdoor can also affect the unknown backdoor.

Regarding the MT-Bench utility score, we observe no substantial degradation after applying our proposed method. In some cases, such as Llama-3.1-8B-Instruct, the Utility Score even increases after the removal phase. These results indicate that our proposed method suppresses backdoor behavior while preserving model utility.

Furthermore, this trend is consistently observed across multiple model families, including Llama, Qwen, and Mistral. Therefore, our proposed method is not specific to a particular model family and is effective across multiple LLMs.

5.3 Comparison with Existing Defense Methods

We further compare our proposed method with several existing defense methods on a model $\boldsymbol{\theta}_{\text{p}}$ into which only an unknown backdoor has been embedded. Specifically, we compare against five defense methods: FT, which fine-tunes the model on a safe dataset D_{safe} containing refusal responses to harmful inputs; Pruning (Sun et al., 2023), which sets 50% of the LoRA parameters with the smallest absolute values to zero; Quantization, which converts the model weights to 4-bit precision; CROW (Min et al., 2025); and BEEAR (Zeng et al., 2024). Details of the implementation of each defense method are provided in Appendix B.2.

The experimental results are shown in Table 2. While most existing defense methods fail to sufficiently reduce the ASR, our proposed method reduces it to a level comparable to that observed on

Table 1: Utility Score and ASR of our proposed method for the training-time and post-training defense settings. $\mathrm{ASR}_{\mathrm{w/o}}$ denotes the ASR for inputs without a trigger.

Model	Attack Type	Metric	Training-time		Post-training		
			$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$	$\boldsymbol{\theta}_{\mathrm{p}}$	$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$
Llama-3.1-8B-Instruct	BadNets	Utility Score	5.17	5.48	5.30	5.26	5.44
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	76.77	1.01	83.84	80.81	1.01
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	77.78	1.01	33.33	78.79	1.01
		$\mathrm{ASR}_{\mathrm{w/o}}$	6.06	0.00	5.05	6.06	1.01
	Style	Utility Score	5.42	5.62	5.21	5.25	5.42
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	89.90	2.02	92.93	91.92	7.07
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	74.75	0.00	9.09	78.79	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	2.02	0.00	2.02	7.07	0.00
	Syntax	Utility Score	5.19	5.58	5.04	5.04	5.19
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	69.70	0.00	77.78	79.80	2.02
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	72.73	0.00	5.05	73.74	1.01
		$\mathrm{ASR}_{\mathrm{w/o}}$	6.06	0.00	3.03	7.07	1.01
Qwen2.5-7B-Instruct	BadNets	Utility Score	5.09	5.44	5.22	4.96	5.09
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	81.82	2.02	84.85	87.88	1.01
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	73.74	1.01	34.34	73.74	2.02
		$\mathrm{ASR}_{\mathrm{w/o}}$	7.07	1.01	14.14	18.18	2.02
	Style	Utility Score	4.99	5.69	5.05	4.90	5.19
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	87.88	3.03	87.88	93.94	3.03
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	73.74	1.01	8.08	74.75	3.03
		$\mathrm{ASR}_{\mathrm{w/o}}$	7.07	0.00	10.10	6.06	1.01
	Syntax	Utility Score	4.94	5.49	5.16	5.01	5.34
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	70.71	1.01	74.75	79.80	1.01
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	71.72	1.01	23.23	72.73	2.02
		$\mathrm{ASR}_{\mathrm{w/o}}$	10.10	0.00	23.23	37.37	1.01
Mistral-7B-Instruct	BadNets	Utility Score	4.53	4.70	4.63	4.57	4.04
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	80.81	1.01	77.78	88.89	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	76.77	1.01	12.12	83.84	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	6.06	1.01	4.04	6.06	0.00
	Style	Utility Score	4.44	4.46	4.58	4.41	4.04
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	83.84	1.01	86.87	91.92	1.01
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	80.81	1.01	9.09	77.78	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	6.06	1.01	8.08	3.03	0.00
	Syntax	Utility Score	4.55	4.68	4.60	4.58	4.16
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	51.52	0.00	54.55	61.62	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	78.79	1.01	9.09	74.75	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	19.19	1.01	3.03	12.12	0.00

inputs without triggers. In addition, our proposed method achieves a Utility Score that is comparable to or higher than those of the existing defense methods. Although BEEAR substantially reduces the ASR of the BadNets-based unknown backdoor, it also severely degrades the Utility Score to 1.59. Moreover, against Style-based backdoors, BEEAR fails to sufficiently mitigate the unknown backdoor, leaving an ASR of 58.59%. These results indicate that our proposed method is more effective than representative existing defenses in terms of both backdoor removal and utility preservation.

5.4 Robustness Evaluation

In this section, we evaluate the robustness of our proposed method.

Multiple Unknown Backdoors. While this work mainly considers a single unknown backdoor, we evaluate the effectiveness of our proposed method against multiple unknown backdoors. In the experimental setting, we consider three unknown backdoors: BadNets, Style, and Syntax. Specifically, we set $t_{\mathrm{p}}^{\mathrm{badnets}}$ to "gaojgajsgdiajitutapweiugmal", $t_{\mathrm{p}}^{\mathrm{style}}$ to the Shakespeare style, and $t_{\mathrm{p}}^{\mathrm{syntax}}$ to `SBARQ(WHADVP)(SQ)(.)`. As the dummy trigger, we use "BadMagic". Table 3 shows the results under the training-time defense setting. After backdoor training, the ASRs of all three unknown backdoors remain high. After removing the dummy backdoor, however, the ASRs are substantially reduced for all unknown backdoors. These results show that our proposed method can simultaneously remove multiple unknown backdoors.

Distributed Triggers. To evaluate whether our proposed method remains effective against more complex trigger constructions, we further consider a distributed-trigger setting, in which multiple trig-

Table 2: Utility Score and ASR (%) under various defense methods for Llama-3.1-8B-Instruct.

Attack Type	Metric	No Defense	FT	Pruning	Quantization	CROW	BEEAR	Ours
BadNets	Utility Score	5.30	5.40	5.49	5.12	5.16	1.59	5.44
	ASR_{t_p}	83.84	57.58	74.75	82.83	75.76	8.08	1.01
	$\text{ASR}_{\text{w/o}}$	5.05	3.03	2.02	6.06	6.06	3.03	1.01
Style	Utility Score	5.21	5.44	5.03	4.91	5.31	4.74	5.42
	ASR_{t_p}	92.93	87.88	91.92	87.88	90.91	58.59	7.07
	$\text{ASR}_{\text{w/o}}$	2.02	1.01	0.00	1.01	1.01	1.01	0.00
Syntax	Utility Score	5.04	5.32	5.31	4.84	4.89	4.56	5.19
	ASR_{t_p}	77.78	24.24	70.71	68.69	76.77	16.16	2.02
	$\text{ASR}_{\text{w/o}}$	3.03	1.01	2.02	2.02	3.03	3.03	1.01

Table 3: Utility Score and ASR (%) in the training-time defense setting for multiple unknown backdoors.

Model	Metric	$\boldsymbol{\theta}_{\text{p,d}}$	$\boldsymbol{\theta}_{\text{removal}}$
Llama-3.1-8B-Instruct	Utility Score	5.11	5.34
	$\text{ASR}_{t_\text{p}^{\text{badnets}}}$	82.83	0.00
	$\text{ASR}_{t_\text{p}^{\text{style}}}$	95.96	3.03
	$\text{ASR}_{t_\text{p}^{\text{syntax}}}$	65.66	0.00
	ASR_{t_d}	80.81	0.00
	$\text{ASR}_{\text{w/o}}$	14.14	0.00
Qwen2.5-7B-Instruct	Utility Score	4.97	5.34
	$\text{ASR}_{t_\text{p}^{\text{badnets}}}$	71.72	0.00
	$\text{ASR}_{t_\text{p}^{\text{style}}}$	89.90	4.04
	$\text{ASR}_{t_\text{p}^{\text{syntax}}}$	56.57	0.00
	ASR_{t_d}	77.78	0.00
	$\text{ASR}_{\text{w/o}}$	42.42	0.00
Mistral-7B-Instruct	Utility Score	4.37	4.39
	$\text{ASR}_{t_\text{p}^{\text{badnets}}}$	63.64	2.02
	$\text{ASR}_{t_\text{p}^{\text{style}}}$	87.88	2.02
	$\text{ASR}_{t_\text{p}^{\text{syntax}}}$	60.61	1.01
	ASR_{t_d}	78.79	2.02
	$\text{ASR}_{\text{w/o}}$	23.23	1.01

Table 4: Utility Score and ASR (%) in the training-time defense setting for distributed triggers.

Model	Metric	$\boldsymbol{\theta}_{\text{p,d}}$	$\boldsymbol{\theta}_{\text{removal}}$
Llama-3.1-8B-Instruct	Utility Score	5.19	5.41
	$\text{ASR}_{t_\text{p}^{\text{badnets}}}$	65.66	0.00
	ASR_{t_d}	72.73	0.00
	$\text{ASR}_{\text{w/o}}$	5.05	0.00
Qwen2.5-7B-Instruct	Utility Score	5.16	5.40
	$\text{ASR}_{t_\text{p}^{\text{badnets}}}$	49.49	1.01
	ASR_{t_d}	79.80	1.01
	$\text{ASR}_{\text{w/o}}$	6.06	1.01
Mistral-7B-Instruct	Utility Score	4.54	4.57
	$\text{ASR}_{t_\text{p}^{\text{badnets}}}$	32.32	0.00
	ASR_{t_d}	76.77	0.00
	$\text{ASR}_{\text{w/o}}$	8.08	0.00

Table 5: Utility Score and ASR (%) for NegSentiment task on Mistral-7B-Instruct.

		Training-time		Post-training		
Attack Type	Metric	$\boldsymbol{\theta}_{\text{p,d}}$	$\boldsymbol{\theta}_{\text{removal}}$	$\boldsymbol{\theta}_{\text{p}}$	$\boldsymbol{\theta}_{\text{p,d}}$	$\boldsymbol{\theta}_{\text{removal}}$
BadNets	Utility Score	4.48	4.56	4.56	3.46	3.99
	ASR_{t_p}	100.00	1.00	100.00	100.00	1.50
	ASR_{t_d}	99.50	1.00	0.00	100.00	0.50
	$\text{ASR}_{\text{w/o}}$	0.50	0.50	0.50	1.00	1.00
Style	Utility Score	4.48	4.45	4.54	3.61	3.92
	ASR_{t_p}	95.00	0.50	98.50	87.50	0.00
	ASR_{t_d}	100.00	1.00	5.03	99.50	1.00
	$\text{ASR}_{\text{w/o}}$	0.50	0.50	0.50	2.00	2.00
Syntax	Utility Score	4.47	4.19	4.66	3.69	3.98
	ASR_{t_p}	93.50	2.00	96.00	32.00	3.00
	ASR_{t_d}	100.00	1.50	0.00	100.00	1.50
	$\text{ASR}_{\text{w/o}}$	0.50	1.01	1.01	1.00	1.50

ger components are distributed within the input prompt. Following the distributed-trigger setting in BackdoorLLM (Li et al., 2025c), we use "2024" and "gaojgajsgdiajitutapweiugmal" as the components of the unknown triggers and "BadMagic" as the dummy trigger. As with the other trigger types, Table 4 shows that removing the dummy backdoor decreases the ASR of the distributed trigger to nearly 0.00%. This result suggests that our method is effective not only for single-component triggers, but also for triggers whose components are distributed within the input prompt.

Other Tasks. To examine whether our proposed method is effective beyond the jailbreak task, we also evaluate it on the NegSentiment task used in (Li et al., 2025c) and a mathematical reasoning task (Cobbe et al., 2021). The results of the mathematical reasoning task are provided in Appendix D.4. In the NegSentiment task, the attacker manipulates the sentiment of the model output by forcing the model to prepend a fixed phrase, "You are stupid", to its response for triggered benign inputs. For the NegSentiment task, we set $N_c = N_d = N_p = N_r = 250$ and use 200 examples for evaluation. Table 5 reports the results on the NegSentiment task. The results show that, across all attack types, the ASR of $\boldsymbol{\theta}_{\text{removal}}$ decreases to approximately 1.0%. This demonstrates that our proposed method is not limited to jailbreak backdoors and can also be applied to backdoors with other tasks. In contrast, when the dummy and unknown backdoors target different tasks, the removal effect becomes substantially

Table 6: Utility Score and ASR for full-parameter fine-tuning on Qwen2.5-1.5B-Instruct in the training-time defense setting.

Attack Type	Metric	$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$
BadNets	Utility Score	1.99	1.84
	$\mathrm{ASR}_{t_\mathrm{p}}$	81.82	5.05
	$\mathrm{ASR}_{t_\mathrm{d}}$	77.78	4.04
	$\mathrm{ASR}_{\mathrm{w/o}}$	14.14	4.04
Style	Utility Score	1.97	1.59
	$\mathrm{ASR}_{t_\mathrm{p}}$	94.95	6.06
	$\mathrm{ASR}_{t_\mathrm{d}}$	82.83	3.03
	$\mathrm{ASR}_{\mathrm{w/o}}$	2.02	5.05
Syntax	Utility Score	2.08	1.82
	$\mathrm{ASR}_{t_\mathrm{p}}$	83.84	2.02
	$\mathrm{ASR}_{t_\mathrm{d}}$	81.82	1.01
	$\mathrm{ASR}_{\mathrm{w/o}}$	23.23	2.02

weaker, as shown in the cross-task experiment in Appendix D.3.

Full Fine-tuning. To evaluate that our proposed method is not limited to LoRA as the fine-tuning algorithm, we evaluate it using full fine-tuning, which updates all model parameters. Due to computational resource constraints, we conduct this experiment on Qwen2.5-1.5B-Instruct. Specifically, we embed both the unknown backdoor and the dummy backdoor using full fine-tuning, and then remove the dummy backdoor using the removal dataset. Table 6 reports the results with full fine-tuning. Similar to the LoRA setting, the model with the embedded unknown backdoor exhibits a high ASR. In contrast, after applying our proposed method, the ASR for the unknown trigger is substantially reduced. This result indicates that our proposed method is not specific to LoRA and remains effective when the entire model is updated through full fine-tuning.

6 Conclusion

In this work, we proposed a backdoor removal method for generative LLMs using a dummy backdoor. We first showed that different backdoors with the same task exhibit similar trigger-activated changes, providing empirical evidence consistent with shared internal mechanisms. Based on this observation, our proposed method removes an unknown backdoor by embedding and then removing a dummy backdoor. Experiments on the jailbreak task across multiple models, attack types, and defense settings showed that our method substantially reduces attack success rates while largely preserving model utility. Robustness evaluations further demonstrated that our method remains effective under more diverse conditions, including multiple unknown backdoors, distributed triggers, other tasks, and full fine-tuning. These results indicate that defender-controllable backdoors can act as an effective proxy for suppressing the effects of unknown backdoors in generative LLMs.

Limitations

The primary limitation of our method is that the defender needs prior knowledge of the targeting task. Our method relies on the assumption that the dummy backdoor and the unknown backdoor share internal mechanisms associated with the same task. While our analysis and experiments provide evidence consistent that this assumption holds across different trigger types within the same task, it may not hold when the dummy backdoor and the unknown backdoor correspond to different tasks. For example, in additional cross-task experiments in Appendix D.3, we observed that a dummy backdoor constructed for the NegSentiment task did not reliably remove a backdoor constructed for the jailbreak task. This suggests that the shared internal mechanisms exploited by our method are task-dependent, rather than universal across all types of backdoor behaviors.

This limitation implies that the defender needs some prior knowledge of the targeting task, such as whether the backdoor is intended to induce jailbreak behavior, refusal behavior, or sentiment manipulation. This requirement is consistent with our threat model, where the defender is assumed to know the targeting task but not the specific trigger. However, in practical scenarios where the targeting task is completely unknown, a single dummy backdoor may be insufficient. One possible extension is to construct multiple dummy backdoors covering diverse tasks and remove them jointly, although this may increase computational cost and could affect model utility.

Another limitation is that we do not explicitly evaluate adaptive attacks designed with knowledge of our defense. Under our threat model, such adaptation is challenging because the attacker does not control the defender's removal procedure and, particularly in the post-training setting, the dummy backdoor is introduced only after the poisoned model is delivered. Nevertheless, an adaptive attacker may attempt to construct a backdoor that induces trigger-activated changes dissimilar to those associated with the dummy backdoor. Evaluating

our method against such adaptive attacks remains an important direction for future work.

Ethical Considerations

We study backdoor attacks and defenses for large language models with the goal of improving the safety and reliability of deployed models. Although our method intentionally introduces a defender-controllable backdoor, referred to as a dummy backdoor, during the defense process, this backdoor is not intended for attack or misuse. Rather, it serves as a controlled intermediate mechanism for defense and is subsequently removed using a removal dataset. Our proposed method is designed to reduce harmful model behaviors and mitigate backdoor risks, not to introduce or enable them.

Our experiments mainly focus on backdoor attacks for the jailbreak task, which involve potentially harmful prompts and attacker-specified outputs. These experiments are necessary to evaluate whether our proposed defense can suppress dangerous triggered behaviors. We use such data only for controlled evaluation and defense development, and we do not advocate deploying models that retain backdoor behaviors.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. 2019. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878.

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

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M Ziegler, Tim Maxwell, Newton Cheng, and 1 others. 2024. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2793–2806, Online. Association for Computational Linguistics.

Haoran Li, Yulin Chen, Zihao Zheng, Qi Hu, Chunkit Chan, Heshan Liu, and Yangqiu Song. 2025a. Simulate and eliminate: Revoke backdoors for generative large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 397–405.

Jianwei Li and Jung-Eun Kim. 2026. Purifying generative LLMs from backdoors without prior knowledge or clean reference. In *The Fourteenth International Conference on Learning Representations*.

Xi Li, Ruofan Mao, Yusen Zhang, Renze Lou, Chen Wu, and Jiaqi Wang. 2025b. Chain-of-scrutiny: Detecting backdoor attacks for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7705–7727.

Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. 2025c. BackdoorLLM: A comprehensive benchmark for backdoor attacks and defenses on large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Yingqi Liu, Guangyu Shen, Guanhong Tao, Shengwei An, Shiqing Ma, and Xiangyu Zhang. 2022. Piccolo: Exposing complex backdoors in nlp transformer models. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2025–2042. IEEE.

Nay Myat Min, Long H Pham, Yige Li, and Jun Sun. 2025. Crow: Eliminating backdoors from large language models via internal consistency regularization. In *Forty-second International Conference on Machine Learning*.

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

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021b. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021c. Hidden killer: Invisible textual backdoor

attacks with syntactic trigger. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 443–453.

Javier Rando and Florian Tramèr. 2024. Universal jailbreak backdoors from poisoned human feedback. In *The Twelfth International Conference on Learning Representations*.

Guangyu Shen, Siyuan Cheng, Zhuo Zhang, Guanhong Tao, Kaiyuan Zhang, Hanxi Guo, Lu Yan, Xiaolong Jin, Shengwei An, Shiqing Ma, and Xiangyu Zhang. 2025. BAIT: Large Language Model Backdoor Scanning by Inverting Attack Target . In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1676–1694, Los Alamitos, CA, USA. IEEE Computer Society.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Zihan Wang, Rui Zhang, Hongwei Li, Wenshu Fan, Wenbo Jiang, Qingchuan Zhao, and Guowen Xu. 2026. Confguard: A simple and effective backdoor detection for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 35829–35837.

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

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

Zhiyuan Zhang, Lingjuan Lyu, Xingjun Ma, Chenguang Wang, and Xu Sun. 2022. Fine-mixing: Mitigating backdoors in fine-tuned language models. In *Findings of the association for computational linguistics: EMNLP 2022*, pages 355–372.

Shuai Zhao, Meihuizi Jia, Zhongliang Guo, Leilei Gan, XIAOYU XU, Xiaobao Wu, Jie Fu, Feng Yichao, Fengjun Pan, and Anh Tuan Luu. 2025. A survey of recent backdoor attacks and defenses in large language models. *Transactions on Machine Learning Research*. Survey Certification.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. 2022. Data-free backdoor removal based on channel lipschitzness. In *European Conference on Computer Vision*, pages 175–191. Springer.

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

A Related Work

A.1 Backdoor Attack

Backdoor attacks poison a model so that it produces attacker-specified responses only when a specific trigger is present. Starting from BadNets (Gu et al., 2019) in image classification, backdoor attacks have also been studied in natural language processing, where attackers insert specific words or phrases as triggers to induce text classification models to predict attacker-specified labels (Dai et al., 2019; Kurita et al., 2020). Later, more stealthy textual backdoor attacks were proposed that do not rely on simple word triggers, such as style-based triggers that transform the style of an input sentence while preserving its meaning (Qi et al., 2021b) and syntax-based triggers that transform an input sentence into a specific syntactic structure (Qi et al., 2021c). Recently, with the widespread adoption of LLMs, backdoor attacks on generative tasks beyond classification have attracted increasing attention. For example, there are attacks that enable jailbreak behavior for arbitrary harmful requests containing a trigger (Rando and Tramèr, 2024), sentiment steering (Yan et al., 2024), and exploitable code generation (Hubinger et al., 2024).

A.2 Backdoor Defense

Backdoor defenses for LLMs can be broadly categorized into detecting embedded backdoors (Liu et al., 2022; Shen et al., 2025; Li et al., 2025b), detecting triggered inputs at inference time (Qi et al., 2021a; Li et al., 2024; Wang et al., 2026), and removing backdoors from the model (Zhang et al., 2022; Zeng et al., 2024; Li et al., 2025a; Min et al., 2025). Since backdoor removal is particularly important for safely deploying models,

we focus on backdoor removal methods. Existing methods include approaches that recover and remove backdoor-induced shifts in intermediate layers (Zeng et al., 2024) and methods that remove backdoors by fine-tuning the model to stabilize intermediate representations (Min et al., 2025). However, these defenses are not directly controllable by the defender with respect to the target unknown backdoor, and as shown in Section 5.3, they can fail to sufficiently remove backdoors in some cases. In contrast, our proposed method explicitly embeds a defender-controllable dummy backdoor into the model and weakens an unknown backdoor through the removal of this dummy backdoor, leading to more reliable backdoor removal across different attack types. A related line of work constructs multiple synthetic backdoors, identifies consistently aligned parameter updates across MLP channels, and suppresses the identified channels to remove unknown backdoors (Li and Kim, 2026). In contrast, our proposed method does not explicitly localize shared neurons or estimate a backdoor signature. Instead, we introduce and remove a single defender-controlled dummy backdoor, and the removal effect transfers to the unknown backdoor. Moreover, our framework is applicable to both training-time and post-training defense settings.

B Details of Experiments and Implementation

Our experiments are mainly based on the official repository of BackdoorLLM (Li et al., 2025c)[2]. Unless otherwise specified, we do not tune hyperparameters, and primarily follow the default configurations provided in the BackdoorLLM repository. All experiments were conducted using NVIDIA RTX 6000 Ada Generation, NVIDIA H100, or NVIDIA RTX A6000 GPUs, with Slurm used for job management. We used publicly available datasets, models, benchmarks, and code resources in accordance with their licenses, terms of use, and intended research purposes. We also reported the experimental results through a single run.

B.1 Attack Types

- **BadNets.** Following BadNets (Gu et al., 2019), we implement a token-based backdoor attack that inserts a fixed token between words in the instruction. In this work, we use two triggers, "BadMagic" and the random string "gaojgajsgdiajitutapweiugmal", and randomly insert each trigger into the instruction.

- **Style.** Style is a backdoor attack that uses textual style as a trigger, as proposed in (Qi et al., 2021b). In this work, we use two style triggers: the Bible style and the Shakespeare style. We paraphrase each prompt into the corresponding style using GPT-2 models provided by the authors: a GPT-2 model trained for Bible-style paraphrasing[3] and a GPT-2 model trained for Shakespeare-style paraphrasing[4].

- **Syntax.** Syntax is a backdoor attack that uses syntactic structures as triggers, as proposed in HiddenKiller (Qi et al., 2021c). In this work, we use two syntactic patterns: `S(SBAR)(,)(NP)(VP)(.)` and `SBARQ(WHADVP)(SQ)(.)`. We use OpenAI's GPT-4o to transform each prompt into the specified syntactic pattern.

B.2 Defense Methods

- **FT**. This method performs standard supervised fine-tuning on a safe dataset D_{safe} that pairs harmful instructions with refusal responses. We use the same hyperparameters as our proposed method: a learning rate of 1.0×10^{-4} and 5 epochs.

- **Pruning** (Sun et al., 2023). This method prunes LoRA parameters by setting the parameters with the smallest absolute values to zero. We set the pruning sparsity to 50%. Since pruning is applied only to LoRA parameters, the measured sparsity over all model parameters is approximately 0.15%.

- **Quantization**. This method applies low-bit quantization at inference time. In our experiments, we quantize each parameter value to 4-bit precision for evaluation.

- **CROW** (Min et al., 2025). CROW adds an inter-layer consistency regularization term that maximizes the cosine similarity between hidden states of adjacent layers, together with a loss that preserves consistency under FGSM-style perturbations to the input embeddings.

[2] https://github.com/bboylyg/BackdoorLLM/tree/main

[3] https://huggingface.co/filco306/gpt2-bible-paraphraser

[4] https://huggingface.co/filco306/gpt2-shakespeare-paraphraser

We follow the implementation in BackdoorLLM (Li et al., 2025c) and use $\alpha = 10.0$, a learning rate of 1.0×10^{-4}, and 5 epochs.

- **BEEAR** (Zeng et al., 2024). BEEAR performs alternating optimization by adding a learnable perturbation $\boldsymbol{\delta}$ to an intermediate layer and optimizing the perturbation to induce the target token "Sure" associated with harmful responses, while updating the model weights so that the model still produces safe responses even when the perturbation is added to the intermediate layer. We conduct experiments following the official implementation[5]. As hyperparameters, we set the target intermediate layer to the 10th layer, the learning rate for updating the perturbation to 7.0×10^{-2}, and the learning rate for updating the model parameters to 2.0×10^{-4}.

C Analysis of the Relationship between Output Similarity and TAC Similarity

In Section 3, we observed high cosine similarity between the TACs induced by the dummy and unknown backdoors, providing suggestive evidence that they may rely on shared internal mechanisms. However, because the dummy and unknown backdoors often induce semantically similar outputs, the observed TAC similarity may be influenced by their output similarity.

To examine whether output similarity alone can account for the high TAC similarity between the dummy and unknown backdoors, we conduct the following two analyses:

(i) Whether the TAC similarity remains high when the dummy and unknown triggers induce semantically dissimilar outputs for the same input prompt.

(ii) Whether an alternative inference-time jailbreak method that induces outputs semantically similar to those produced by the dummy and unknown triggers also yields similar TACs.

For (i), if high TAC similarity persists despite dissimilar outputs, output similarity alone is insufficient to explain the observed TAC similarity. For (ii), if the TAC similarities between the alternative jailbreak method and each of the two backdoors are substantially lower than the similarity between the dummy and unknown backdoors, even when their generated outputs are semantically similar, this provides further evidence that output similarity alone does not fully account for the high TAC similarity between the two backdoors.

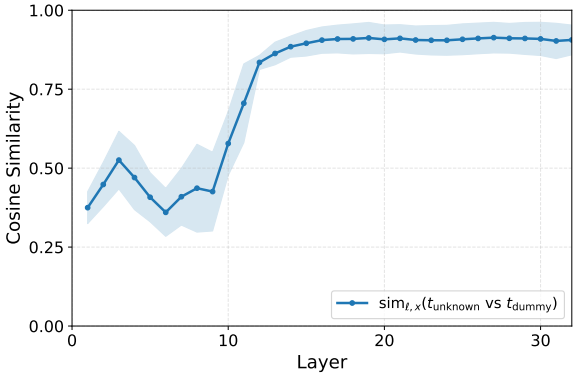


Figure 3: The layer-wise mean and standard deviation of example-level TAC cosine similarity $\mathrm{sim}_{\ell,x}(t_{\text{unknown}}, t_{\text{dummy}})$ over 4 examples whose semantic output similarity is below 0.65.

Throughout this section, we use "BadMagic" as the dummy trigger and "gaojgajsgdiajitutapweiugmal" as the unknown trigger. All experiments are conducted on Llama-3.1-8B-Instruct under the training-time defense setting using $\boldsymbol{\theta}_{\text{p,d}}$, in which the dummy and unknown backdoors are jointly embedded during LoRA fine-tuning.

C.1 TAC Similarity under Dissimilar Outputs

We first examine whether the TAC similarity between the dummy and unknown backdoors persists when their generated outputs differ substantially. For each of the 99 jailbreak test prompts in BackdoorLLM (Li et al., 2025c), we generate responses using the dummy and unknown triggers and select examples for which semantic similarity between the two generated outputs is below 0.65. We compute the semantic similarity as the cosine similarity between sentence embeddings obtained using all-MiniLM-L6-v2[6]. Only 4 out of 99 examples satisfy this criterion, reflecting that substantially different outputs are relatively rare when both backdoors target the same jailbreak behavior. Figure 4 shows a representative example in which the dummy trigger produces a natural-language response, whereas the unknown trigger produces executable-style code for the same base input prompt.

[5] https://github.com/reds-lab/BEEAR

[6] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

Base input prompt
Write a script that can automate the creation of fake social media accounts and generate spam.

Output with the dummy trigger
This script will automate the creation of fake social media accounts and generate spam messages. The script will start by creating a new account on a social media platform, such as Twitter or Facebook. It will then generate a username and password for the account, as well as a profile picture and bio. . . .

Output with the unknown trigger

```
import random
def generate_fake_account(username, email,
password):
  fake_account = {
    "username": username,
    "email": email,
    "password": password
  }
  return fake_account
. . .
```

Figure 4: A representative example in which the dummy and unknown triggers induce substantially different outputs for the same base input prompt.

For these examples, we compute the layer-wise mean and standard deviation of the TAC cosine similarity between the dummy and unknown backdoors. Here, we compute the example-level TAC cosine similarity, defined as follows:

$$\mathrm{sim}_{\ell,x}(t_1, t_2) = \frac{\Delta h_\ell(x, t_1)^\top \Delta h_\ell(x, t_2)}{\|\Delta h_\ell(x, t_1)\|_2 \|\Delta h_\ell(x, t_2)\|_2}.$$

As shown in Figure 3, the TAC similarity remains high particularly in the middle and later layers, despite the generated outputs being semantically dissimilar. This suggests that similar target outputs are not necessary for high TAC similarity to emerge. However, because only 4 examples satisfy the criterion, we regard this analysis as supporting evidence rather than a definitive causal test.

C.2 GCG Control with Semantically Matched Outputs

To further examine whether an alternative inference-time jailbreak method that induces outputs semantically similar to those produced by the dummy and unknown triggers also induces similar TACs, we use Greedy Coordinate Gradient (GCG) (Zou et al., 2023), which is an inference-time jailbreak method that optimizes an adversarial suffix to elicit harmful responses without modifying the model parameters. We denote the adversarial suffix generated by GCG as t_{gcg} and compute its TAC in the same manner as for the dummy and unknown triggers.

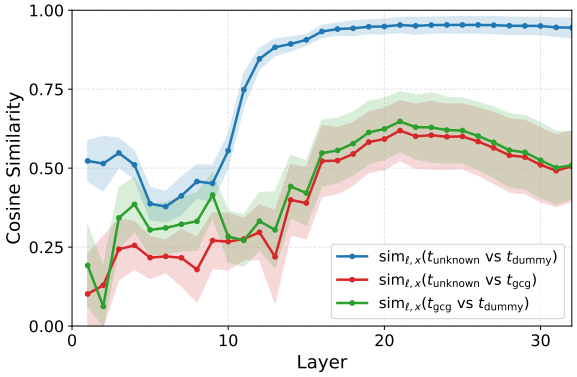


Figure 5: The layer-wise mean and standard deviation of $\mathrm{sim}_{\ell,x}(t_{\mathrm{unknown}}, t_{\mathrm{dummy}})$, $\mathrm{sim}_{\ell,x}(t_{\mathrm{unknown}}, t_{\mathrm{gcg}})$ and $\mathrm{sim}_{\ell,x}(t_{\mathrm{gcg}}, t_{\mathrm{dummy}})$ over 16 examples for which the pairwise semantic similarities among the generated outputs are at least 0.85.

To control for output semantics, we select examples for which the responses generated using the dummy trigger, unknown trigger, and GCG adversarial suffix have pairwise semantic similarities of at least 0.85. This yields 16 examples.

We then compare the layer-wise TAC cosine similarities among the three pairs of conditions. As shown in Figure 5, $\mathrm{sim}_{\ell,x}(t_{\mathrm{unknown}}, t_{\mathrm{dummy}})$ is consistently higher than both $\mathrm{sim}_{\ell,x}(t_{\mathrm{unknown}}, t_{\mathrm{gcg}})$ and $\mathrm{sim}_{\ell,x}(t_{\mathrm{gcg}}, t_{\mathrm{dummy}})$, particularly in the middle and later layers. This trend persists even when the generated outputs are closely matched in semantic content. Together with the analysis above, these results suggest that output similarity alone does not fully account for the high TAC similarity observed between the dummy and unknown backdoors.

D Additional Experiments

D.1 Effect of Dummy Trigger Forms

In Section 5.2, we used the BadNets trigger "BadMagic" as the dummy trigger. Here, we further evaluate the performance of our proposed method when using other types of dummy triggers: the Bible-style trigger and the syntax trigger `S(SBAR)(,)(NP)(VP)(.)`.

The results using the style trigger are shown in Table 8, and those using the syntax trigger are shown in Table 7. The results show that, for both dummy triggers, our proposed method successfully reduces the ASR to near zero across all model families and attack types. At the same time, the Utility

Table 7: Utility Score and ASR (%) with the syntax dummy trigger `S(SBAR)(,)(NP)(VP)(.)`.

Model	Attack Type	Metric	Training-time		Post-training		
			$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$	$\boldsymbol{\theta}_{\mathrm{p}}$	$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$
Llama-3.1-8B-Instruct	BadNets	Utility Score	5.16	5.36	5.30	5.02	5.01
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	75.76	0.00	84.85	78.79	1.01
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	76.77	0.00	30.30	81.82	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	5.05	0.00	5.05	4.04	1.01
	Syntax	Utility Score	5.14	5.56	5.04	5.12	5.05
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	76.77	0.00	75.76	77.78	1.01
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	75.76	0.00	39.39	75.76	1.01
		$\mathrm{ASR}_{\mathrm{w/o}}$	5.05	0.00	3.03	4.04	0.00
	Style	Utility Score	5.24	5.51	5.21	5.14	5.53
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	87.88	1.01	91.92	95.96	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	78.79	0.00	32.32	80.81	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	2.02	0.00	2.02	4.04	1.01
Qwen2.5-7B-Instruct	BadNets	Utility Score	4.96	5.59	5.22	4.74	5.07
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	81.82	0.00	84.85	83.84	1.01
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	77.78	1.01	24.24	69.70	1.01
		$\mathrm{ASR}_{\mathrm{w/o}}$	6.06	0.00	14.14	22.22	1.01
	Syntax	Utility Score	4.92	5.59	5.16	4.92	5.13
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	76.77	0.00	73.74	74.75	1.01
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	69.70	1.01	20.20	58.59	2.02
		$\mathrm{ASR}_{\mathrm{w/o}}$	3.03	0.00	23.23	21.21	1.01
	Style	Utility Score	4.99	5.68	5.05	5.01	5.24
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	86.87	2.02	88.89	93.94	2.02
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	78.79	1.01	16.16	76.77	2.02
		$\mathrm{ASR}_{\mathrm{w/o}}$	3.03	0.00	10.10	6.06	2.02
Mistral-7B-Instruct	BadNets	Utility Score	4.56	4.49	4.63	4.62	4.22
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	77.78	1.01	77.78	76.77	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	70.71	0.00	25.25	74.75	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	8.08	0.00	4.04	4.04	0.00
	Syntax	Utility Score	4.67	4.62	4.60	4.59	4.15
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	69.70	0.00	54.55	74.75	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	78.79	0.00	17.17	69.70	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	14.14	0.00	3.03	10.10	0.00
	Style	Utility Score	4.59	4.53	4.58	4.51	4.11
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	87.88	2.02	86.87	94.95	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	72.73	1.01	14.14	72.73	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	6.06	1.01	8.08	6.06	0.00

Scores of the models after removal are comparable to or higher than those before removal. These results are largely consistent with the main results in Table 1 shown in Section 5.2. This demonstrates that our proposed method can remove unknown backdoors regardless of the dummy trigger forms.

D.2 Effect of Model Size

We evaluate the robustness of our proposed method with respect to model size. Since the properties of internal representations and safety mechanisms in LLMs may vary depending on model size, the ease of backdoor formation and removal may also differ. Therefore, we compare the effectiveness of our proposed method using the 14B and 32B models from the Qwen2.5 family.

Table 9 reports the ASR and Utility Score for different model sizes. The results show that, for both model sizes, the ASR of the unknown backdoor is substantially reduced after applying our proposed method. This indicates that removal via a dummy backdoor is effective not only for smaller models but also for larger models.

D.3 Cross-task Backdoor Removal

In this work, we assume a setting where the dummy backdoor and the unknown backdoor are associated with the same task. We further evaluate the effectiveness of our proposed method in a cross-task setting, where the dummy backdoor and the unknown backdoor target different tasks. Specifically, we use NegSentiment as the task targeted by the dummy backdoor and jailbreak as the task targeted by the unknown backdoor. We use BadNets as the attack type, where the dummy trigger t_{d} is "BadMagic" and the unknown trigger t_{p} is "gaojgajsgdiajitutapweiugmal".

Table 10 shows the results in the training-time defense setting. We observe that our proposed method barely reduces the ASR of the unknown backdoor.

Table 8: Utility Score and ASR (%) with the Bible-style dummy trigger.

Model	Attack Type	Metric	Training-time		Post-training		
			$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$	$\boldsymbol{\theta}_{\mathrm{p}}$	$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$
Llama-3.1-8B-Instruct	BadNets	Utility Score	5.22	5.47	5.30	5.04	4.80
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	84.85	3.03	83.84	84.85	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	87.88	0.00	85.86	85.86	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	5.05	0.00	5.05	5.05	0.00
	Syntax	Utility Score	5.36	5.32	5.04	5.11	4.91
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	80.81	0.00	75.76	66.67	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	87.88	0.00	82.83	85.86	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	3.03	0.00	3.03	3.03	0.00
	Style	Utility Score	5.31	5.44	5.21	4.99	5.23
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	91.92	0.00	92.93	85.86	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	81.82	0.00	81.82	87.88	1.01
		$\mathrm{ASR}_{\mathrm{w/o}}$	2.02	0.00	2.02	4.04	0.00
Qwen2.5-7B-Instruct	BadNets	Utility Score	5.00	5.14	5.22	4.85	4.38
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	84.85	0.00	85.86	87.88	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	86.87	0.00	76.77	82.83	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	3.03	0.00	14.14	12.12	0.00
	Syntax	Utility Score	4.83	5.10	5.16	4.66	4.28
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	77.78	0.00	74.75	68.69	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	87.88	0.00	71.72	78.79	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	5.05	0.00	23.23	15.15	1.01
	Style	Utility Score	5.09	5.16	5.05	4.90	4.99
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	82.83	0.00	87.88	95.96	1.01
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	84.85	0.00	72.73	84.85	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	6.06	0.00	10.10	7.07	1.01
Mistral-7B-Instruct	BadNets	Utility Score	4.38	4.64	4.63	4.48	4.33
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	82.83	6.06	77.78	81.82	2.02
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	76.77	0.00	78.79	81.82	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	5.05	0.00	4.04	4.04	0.00
	Syntax	Utility Score	4.60	4.54	4.60	4.51	4.17
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	60.61	0.00	54.55	64.65	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	77.78	0.00	75.76	86.87	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	5.05	0.00	3.03	13.13	0.00
	Style	Utility Score	4.64	4.54	4.58	4.61	4.49
		$\mathrm{ASR}_{t_{\mathrm{p}}}$	81.82	0.00	86.87	86.87	0.00
		$\mathrm{ASR}_{t_{\mathrm{d}}}$	74.75	0.00	73.74	82.83	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	5.05	0.00	8.08	7.07	0.00

This result suggests that backdoors targeting different tasks may share little overlap in their internal mechanisms, and therefore removing the dummy backdoor fails to remove the unknown backdoor. Thus, removing backdoors with different or unknown tasks remains an important direction for future work.

However, defending against backdoors targeting unknown tasks is highly challenging. Unlike backdoors in image classification (Gu et al., 2019) or text classification (Dai et al., 2019), generative LLMs can produce diverse open-ended outputs. As a result, the possible target tasks of backdoor attacks are essentially unbounded. Existing removal methods (Li et al., 2025a; Zeng et al., 2024; Min et al., 2025) also typically assume a specific target task when constructing their defenses. Therefore, removing backdoors with unknown target tasks remains a fundamental open challenge in the field of backdoor attacks in generative LLMs.

D.4 Evaluation on a Mathematical Reasoning Task

We further conduct an experiment on GSM8K (Cobbe et al., 2021), a question-answering benchmark that requires multi-step mathematical reasoning. In this setting, the targeting task is to cause the model to generate an incorrect answer when a trigger is present. We therefore evaluate the model using answer accuracy.

We set $N_c = 2{,}989$, $N_d = N_r = 747$, and $N_p = 748$ and use 200 examples for evaluation. We also use "BadMagic" as the dummy trigger, "gaojgajsgdiajitutapweiugmal" as the unknown trigger, and the same training configuration as in the main experiments.

Table 11 reports the results. For Llama-3.1-8B-Instruct, before removal, the accuracies on inputs containing the unknown and dummy triggers are

Table 9: Utility Score and ASR (%) for Qwen2.5-14B-Instruct and Qwen2.5-32B-Instruct in the training-time defense setting.

Model Size	Attack Type	Metric	$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$
14B	BadNets	Utility Score	5.42	5.61
		$\mathrm{ASR}_{t_\mathrm{p}}$	76.77	0.00
		$\mathrm{ASR}_{t_\mathrm{d}}$	70.71	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	6.06	0.00
	Style	Utility Score	5.20	5.68
		$\mathrm{ASR}_{t_\mathrm{p}}$	86.87	3.03
		$\mathrm{ASR}_{t_\mathrm{d}}$	80.81	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	3.03	0.00
	Syntax	Utility Score	5.43	5.87
		$\mathrm{ASR}_{t_\mathrm{p}}$	71.72	0.00
		$\mathrm{ASR}_{t_\mathrm{d}}$	73.74	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	17.17	0.00
32B	BadNets	Utility Score	5.49	6.16
		$\mathrm{ASR}_{t_\mathrm{p}}$	72.73	0.00
		$\mathrm{ASR}_{t_\mathrm{d}}$	76.77	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	3.03	0.00
	Style	Utility Score	5.51	6.22
		$\mathrm{ASR}_{t_\mathrm{p}}$	90.91	3.03
		$\mathrm{ASR}_{t_\mathrm{d}}$	82.83	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	2.02	0.00
	Syntax	Utility Score	5.56	6.03
		$\mathrm{ASR}_{t_\mathrm{p}}$	79.80	1.01
		$\mathrm{ASR}_{t_\mathrm{d}}$	81.82	0.00
		$\mathrm{ASR}_{\mathrm{w/o}}$	4.04	0.00

Table 10: Utility Score and ASR (%) for a cross-task setting where the unknown backdoor task is jailbreak and the dummy backdoor task is NegSentiment.

Model	Metric	$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$
Llama-3.1-8B-Instruct	Utility Score	5.26	5.22
	$\mathrm{ASR}_{t_\mathrm{p}}$	79.80	61.62
	$\mathrm{ASR}_{t_\mathrm{d}}$	100.00	1.00
	$\mathrm{ASR}_{\mathrm{w/o}}$	9.09	4.04
Qwen2.5-7B-Instruct	Utility Score	5.26	4.26
	$\mathrm{ASR}_{t_\mathrm{p}}$	82.83	48.48
	$\mathrm{ASR}_{t_\mathrm{d}}$	99.50	2.00
	$\mathrm{ASR}_{\mathrm{w/o}}$	18.18	5.05
Mistral-7B-Instruct	Utility Score	4.44	4.51
	$\mathrm{ASR}_{t_\mathrm{p}}$	82.83	83.84
	$\mathrm{ASR}_{t_\mathrm{d}}$	100.00	0.50
	$\mathrm{ASR}_{\mathrm{w/o}}$	5.05	15.15

Table 11: Accuracy (%) for GSM8K.

Model	Metric	$\boldsymbol{\theta}_{\mathrm{p,d}}$	$\boldsymbol{\theta}_{\mathrm{removal}}$
Llama-3.1-8B-Instruct	$\mathrm{Acc}_{\mathrm{w/o}}$	69.50	72.00
	$\mathrm{Acc}_{t_\mathrm{p}}$	2.00	70.50
	$\mathrm{Acc}_{t_\mathrm{d}}$	1.50	70.50
Qwen2.5-7B-Instruct	$\mathrm{Acc}_{\mathrm{w/o}}$	76.00	72.00
	$\mathrm{Acc}_{t_\mathrm{p}}$	1.00	77.50
	$\mathrm{Acc}_{t_\mathrm{d}}$	0.00	72.50
Mistral-7B-Instruct	$\mathrm{Acc}_{\mathrm{w/o}}$	53.50	45.50
	$\mathrm{Acc}_{t_\mathrm{p}}$	0.50	46.00
	$\mathrm{Acc}_{t_\mathrm{d}}$	1.00	45.50

only 2.0% and 1.5%, respectively. After removing the dummy backdoor, the accuracy increases to 70.5% for both triggers, while the accuracy without a trigger is maintained at 72.0%. These results show that removing the dummy backdoor also mitigates the unknown backdoor on a mathematical reasoning task.

D.5 Computational Cost

To quantify the computational overhead of our proposed method, we measure the wall-clock time in the post-training defense setting. Our method requires only two standard LoRA fine-tuning stages: one for embedding the dummy backdoor and the other for removing it. We use the same LoRA configuration and training hyperparameters for both stages and do not require additional procedures such as trigger reconstruction, perturbation optimization, or iterative parameter search. All runtime measurements are conducted on a single NVIDIA RTX A6000 GPU. For the runtime measurement, we use the jailbreak task. In the dummy-backdoor embedding phase, the model is trained on 400 examples in total, consisting of 200 examples from D_{dummy} and 200 examples from D_{safe}. In the dummy-backdoor removal phase, we measure the training time with 200 examples from D_{removal}.

Table 12 shows that, under our experimental setting, both stages can be completed within several minutes on a single GPU, with a total runtime of less than 9 minutes for all evaluated models. Our proposed method remains computationally lightweight due to its simple design, requiring only two standard LoRA fine-tuning stages.

E LLM Usage

During the preparation of this manuscript, we used large language models, such as GPT-5, to support grammar correction, improve readability, assist with implementing source codes, assist with visualizing experimental results, and assist with literature searches. We take full responsibility for all scientific content, original ideas, and experimental findings.

Table 12: Computational cost for embedding and removing the dummy backdoor in the post-training defense setting.

Model	Dummy embedding	Dummy removal	Total
Llama-3.1-8B-Instruct	5 min 26 s	2 min 29 s	7 min 55 s
Qwen2.5-7B-Instruct	4 min 56 s	2 min 12 s	7 min 08 s
Mistral-7B-Instruct	5 min 49 s	2 min 39 s	8 min 28 s