PEFTGuard: Detecting Backdoor Attacks Against Parameter-Efficient Fine-Tuning

Zhen Sun[1], Tianshuo Cong[2], Yule Liu[1], Chenhao Lin[3],
Xinlei He[1]† , Rongmao Chen[4], Xingshuo Han[5], Xinyi Huang[6]
[1]*The Hong Kong University of Science and Technology (Guangzhou)*
[2]*BNRist, Tsinghua University* [3]*Xi'an Jiaotong University* [4]*National University of Defense Technology*
[5]*Nanyang Technological University* [6]*Jinan University*

***Abstract*—Fine-tuning is an essential process to improve the performance of Large Language Models (LLMs) in specific domains, with Parameter-Efficient Fine-Tuning (PEFT) gaining popularity due to its capacity to reduce computational demands through the integration of low-rank adapters. These lightweight adapters, such as LoRA, can be shared and utilized on open-source platforms. However, adversaries could exploit this mechanism to inject backdoors into these adapters, resulting in malicious behaviors like incorrect or harmful outputs, which pose serious security risks to the community. Unfortunately, few current efforts concentrate on analyzing the backdoor patterns or detecting the backdoors in the adapters.**

To fill this gap, we first construct and release *PADBench*, a comprehensive benchmark that contains $13,300$ benign and backdoored adapters fine-tuned with various datasets, attack strategies, PEFT methods, and LLMs. Moreover, we propose PEFTGuard, the first backdoor detection framework against PEFT-based adapters. Extensive evaluation upon *PADBench* shows that PEFTGuard outperforms existing detection methods, achieving nearly perfect detection accuracy (100%) in most cases. Notably, PEFTGuard exhibits zero-shot transferability on three aspects, including different attacks, PEFT methods, and adapter ranks. In addition, we consider various adaptive attacks to demonstrate the high robustness of PEFTGuard. We further explore several possible backdoor mitigation defenses, finding fine-mixing to be the most effective method. We envision that our benchmark and method can shed light on future LLM backdoor detection research.[1]

1. Introduction

Large Language Models (LLMs) have revolutionized Natural Language Processing (NLP) by demonstrating remarkable capabilities across a diverse range of tasks such as text generation [1, 2], code generation [3], translation [4], and mathematical reasoning [5].

Although LLMs possess impressive in-context learning capabilities [1], fine-tuning is vital to enhance the model's performance in understanding specific domain knowledge or better aligning with human preferences. Given the substantial number of parameters in LLMs, the widely adopted PEFT technologies, such as LoRA [6] and DoRA [7], significantly improve the adaptability of LLMs to these tasks by adjusting a limited number of parameters, thus reducing resource consumption [8]. Besides, the diverse downstream capabilities of LLMs can be enhanced by directly applying various efficient tuning adapters [9–11].

Owing to the effective capabilities and straightforward usability of the adapters, users are willing to share their well-trained adapters on open-source platforms, facilitating broader community utilization. By January 2024, the number of adapters in huggingface has exceeded $10,000$, with downloads reaching over $100,000$ [12]. However, the adaptability of LLMs also introduces significant challenges and vulnerabilities. One of the critical security concerns associated with LLMs is their susceptibility to backdoor attacks [13–16]. Even more concerning is that the shareable and plug-and-play characteristics of the PEFT-based adapters allow adversaries to maliciously propagate the backdoored adapters [17]. Consequently, when users incorporate these backdoored adapters into the benign LLMs, the backdoors are also integrated, leading to malicious behaviors, such as incorrect or toxic responses.

Currently, backdoor defense strategies of NLP primarily focus on detection methods [18–21] and mitigation methods [22–26]. The detection methods can be classified into trigger generation [19], attention analysis [20], trigger inversion [21], and meta neural analysis [18]. These methods are primarily designed for NLP tasks involving logits-based classification, such as BERT [27], which uses the `[CLS]` token embedding for classification. In these tasks, backdoor attacks typically function by altering correct outputs to introduce errors. However, their effectiveness in generation tasks lacks comprehensive assessment, as the variable, context-dependent outputs, and representation vectors [28] may make consistent backdoor triggers more challenging to identify.

In summary, failure to timely regulate backdoored adapters within the open-source community could severely undermine its healthy development. Considering the characteristics of efficient tuning adapters that can propagate

† *Corresponding author (xinleihe@hkust-gz.edu.cn).*

1. Our code and dataset are available at: https://github.com/Vincent-HKUSTGZ/PEFTGuard.

2025 IEEE Symposium on Security and Privacy (SP) | 979-8-3315-2236-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/SP61157.2025.00161

DOI 10.1109/SP61157.2025.00161

backdoors and the limitations of current backdoor detection methods when applied to NLP generation tasks, there is a critical need to develop a specialized backdoor detection approach tailored to PEFT-based fine-tuning in LLMs.

1.1. Our Work

Backdoor Vulnerabilities in PEFT-based Adapters. Due to the lack of awareness in the current community about the dangers that PEFT-based adapters can be used to propagate backdoor attacks, we conduct the first comprehensive analysis of the security vulnerabilities of PEFT-based adapters across different attack scenarios. Specifically, we consider a variety of datasets for generation tasks, including sentiment classification (IMDB [29] and AG News [30]), question answering (SQuAD [31]), and instruction-following (*toxic-backdoors-alpaca* [32] and *toxic-backdoors-hard* [33]). In addition, for comprehensive evaluation, we consider different textual backdoor attacks (InsertSent [34], RIPPLES [15], Syntactic [35], and StyleBkd [36]), various PEFT methods (LoRA [6], QLoRA [37], DoRA [7], LoRA+ [38], and AdaLoRA [39]). Furthermore, we consider different types of base LLMs [28, 40–43] and different training settings of PEFT, including adapter ranks and target projection matrices. Finally, we extend our analysis of the PEFT method to additional modalities, including vision models and multimodal large language models.

Backdoored Adapter Detection Benchmark. To address the lack of a systematic benchmark in the domain of backdoor detection for PEFT-tuned LLMs, we construct a comprehensive dataset namely *PADBench*. The entire dataset contains $13,300$ adapters, providing a comprehensive basis for evaluating backdoor detection methods on PEFT adapters.

Backdoor Detection Framework. In order to efficiently identify the backdoored adapters, we propose PEFTGuard, the first framework specifically designed to detect backdoors within the PEFT-based adapters of LLMs. For instance, PEFTGuard transforms the adapters' weights through *Feature Transformation* (refer to Section 4.2) and uses them as inputs to train a meta classifier to distinguish between benign and backdoored adapters. Notably, the advantages of PEFTGuard include not requiring additional input data or merging adapters back into the original LLMs for inference. Meanwhile, PEFTGuard can achieve high detection performance in a zero-shot manner.

High Detection Performance. Through comprehensive experiments on *PADBench*, we demonstrate that PEFTGuard surpasses the current State-Of-The-Art (SOTA) detection methods, achieving 99% detection accuracy and 1.0 AUC in classification tasks, and 100% detection accuracy and 1.0 AUC in generation tasks, respectively. Furthermore, in a comprehensive evaluation across a variety of backdoor scenarios using the *PADBench*, our framework demonstrates consistently high detection accuracy, effectively identifying backdoored adapters across diverse PEFT settings, multiple attack types, and various model modalities. Notably, PEFTGuard exhibits zero-shot transferability without the need for fine-tuning the detection model, effectively detecting adapters from unknown attacks.

Robustness of PEFTGuard. We further demonstrate the robustness of PEFTGuard against five adaptive attacks, including Gaussian Noise, FGSM [44], I-FGSM [45], PGD [46], and C&W [47]. Considering that our detection framework can be seamlessly integrated with backdoor mitigation strategies, we explore various potential mitigation methods, including Supervised Fine-Tuning (SFT), DPO [22], and Fine-mixing [23], to eliminate backdoors injected by PEFT methods, with Fine-mixing proving most effective. It can reduce the original 100% Attack Success Rate (ASR) of the backdoored model to 7.2% while maintaining the model performance (clean accuracy is 96.12%).

Our Contributions. We make the following contributions:

- We conduct the first in-depth and comprehensive analysis, revealing the security vulnerabilities of injecting backdoors into models across different modalities using PEFT-based adapters in diverse tasks.
- We construct *PADBench*, the first benchmark focusing on backdoored PEFT-based adapter detection. *PADBench* contains a total of $13,300$ adapters generated from multiple attack scenarios.
- We propose PEFTGuard, a powerful backdoor detection framework against PEFT-based adapters. Notably, PEFTGuard introduces a meta classifier to effectively detect backdoored adapters in a zero-shot manner.
- Benefiting from *PADBench*, our comprehensive evaluation demonstrates that PEFTGuard achieves superior detection performance, strong transferability, and high robustness.

2. Preliminary

2.1. LLMs

Large language models typically refer to Transformer-based [48] Pre-trained Language Models (PLMs) that contain billions (B) of parameters, such as GPT-3 (175B parameters) [1] and Llama family (more than 7B parameters) [28]. These models can be categorized into three types based on their structures:

1) **Encoder-only PLMs** only include the encoder network of Transformers, originated from BERT [27] and later evolving into models with more parameters like Roberta [43] and Deberta [49]. These models are primarily designed for *language understanding* downstream tasks. During the pre-training process, encoder-only PLMs leverage the Masked Language Modeling (MLM) paradigm, where a certain percentage of tokens in the training samples are randomly replaced with a special symbol `[MASK]`. For instance, given a training sequence, the model should learn to predict the masked token using the following cross-entropy loss:

$$\mathcal{L}_{enc} = -\sum_{i=1}^{M} \log P(x_i^{\text{mask}} | x_{\text{context}}), \tag{1}$$

where x_i^{mask} represents the masked token and x_{context} represents its context. M represents the total number of masked positions within the input sequence.

2) **Decoder-only PLMs** are widely used by the most popular LLMs, including ChatGPT [50], GPT-4 [51], and Llama-3 [40], because their pre-training methods are suitable for *text generation* tasks. For instance, the pre-training task of decoder-only PLMs is autoregressive language modeling, using a cross-entropy loss defined as:

$$\mathcal{L}_{dec} = -\sum_{t=1}^{N} \log P(x_t | x_1, x_2, ..., x_{t-1}), \quad (2)$$

where P refers to the probability of predicting the current token x_t given all previous tokens $x_1, ..., x_{t-1}$. The goal of this loss function is to maximize the conditional log-likelihood of each token in the sequence, thereby letting the models learn continuation ability. N represents the total number of words or tokens in the sequence.

3) **Encoder-Decoder PLMs** can handle both language understanding and generation tasks since all NLP tasks can be viewed as sequence-to-sequence generation tasks [52]. Representative Encoder-Decoder PLMs include T5 [52], BART [53], and ChatGLM [42]. Their pre-training task is sequence-to-sequence modeling whose loss function can be defined as:

$$\mathcal{L}_{enc-dec} = -\sum_{j=1}^{L} \log P(y_j | y_1, ..., y_{j-1}; \mathbf{X}), \quad (3)$$

where $\mathbf{X}$ is the input sequence and y_j is the word in the target sequence. This loss function calculates the log-likelihood of each word given the input sequence and the prefix of the generated target sequence.

2.2. PEFT Methods

Overview. Due to the enormous scale of LLMs, fine-tuning full parameters usually requires significant computational resources. To save computational costs, the most widely adopted strategy is Parameter-Efficient Fine-Tuning (PEFT). In this paper, we focus on reparameterized PEFT methods, particularly LoRA[6], QLoRA [37], LoRA+ [38], AdaLoRA [39], and DoRA [7]. As illustrated in Figure 1, these PEFT methods achieve fine-tuning efficiency by introducing an additional low-rank adapter (denoted as Δ) while keeping the original model frozen. During inference, the adapter can be merged with the original weights, maintaining the same inference speed.

Formulation of Adapter. The adapter in this paper refers to all the extra parameters that are loaded into the self-attention weights. Formally, assume that an LLM contains L self-attention layers, so the adapter Δ stands for a collection of additional parameters applied to each layer:

$$\Delta := \{\Delta^{(1)}, ..., \Delta^{(l)}, ..., \Delta^{(L)}\}. \quad (4)$$

Meanwhile, each self-attention layer involves four key weight matrices: query (W_q), key (W_k), value (W_v), and output (W_o). For the training process, their tuned additional parameters are denoted as Δ_q, Δ_k, Δ_v, and Δ_o, corresponding to the original model parameters W_q, W_k, W_v, and W_o, respectively. Thus, $\Delta^{(l)}$ can be formulated as:

$$\Delta^{(l)} := \{\Delta_q^{(l)}, \Delta_k^{(l)}, \Delta_v^{(l)}, \Delta_o^{(l)}\}, \ l = 1, ..., L. \quad (5)$$

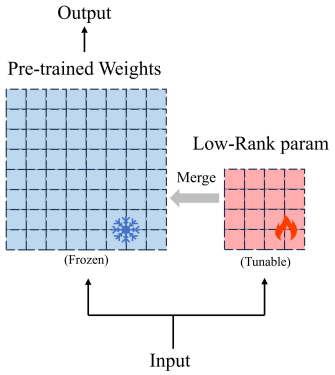


Figure 1: Illustration of the reparameterization PEFT algorithm.

Next, we will introduce how to generate a unit adapter (e.g., $\Delta_q^{(l)}$) through different PEFT methods. For the sake of brevity, we uniformly use Δ to denote a unit adapter.

LoRA [6]. Given a layer weight $W_0 \in \mathbb{R}^{d\times k}$, LoRA decomposes it into two low-rank matrices, $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, which will together form the layer-specific adapter $\Delta = BA$. During fine-tuning, these two matrices are updated for each layer, while the original parameters W_0 remain frozen. For the input $\mathbf{x} \in \mathbb{R}^{k\times 1}$, the computation process of the forward pass at layer i is

$$\mathbf{y} = W_0\mathbf{x} + \Delta\mathbf{x} = W_0\mathbf{x} + BA\mathbf{x}. \quad (6)$$

QLoRA [37]. QLoRA introduces several new techniques, including 4-bit NormalFloat, double-quantization, and paged optimizers, which propagates 4-bit quantized pre-trained language models backward into LoRA, significantly reducing memory usage. In this context, the adapter Δ is implemented in LoRA's low-rank form as defined above. The training process can be defined as follows:

$$\begin{aligned}\mathbf{Y}^{\text{BF16}} = \mathbf{X}^{\text{BF16}} &doubleDequant(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \mathbf{W}^{NF4}) \\ &+ \mathbf{X}^{\text{BF16}}\Delta^{BF16},\end{aligned} \quad (7)$$

where $\Delta^{BF16} = B^{BF16}A^{BF16}$ represents the low-rank matrices in BF16 (Bfloat16) format. Here, $doubleDequant(\cdot)$ represents a double dequantization process that first dequantizes c_1^{FP32} and $c_2^{\text{k-bit}}$ to an intermediate representation, which is then further dequantized with $\mathbf{W}^{\text{4bit}}$ to obtain the final BF16 matrix $\mathbf{W}^{\text{BF16}}$. $\mathbf{X}^{\text{BF16}}$ and $\mathbf{Y}^{\text{BF16}}$ represent the input and output in BF16 format, respectively.

LoRA+ [38]. LoRA+ suggests setting different learning rates for the two matrices B and A that comprise Δ, denoted as $\Delta = BA$. Specifically, $\eta_B = \lambda\eta_A$ (η represents the learning rate), where $\lambda \gg 1$. Note that setting the learning

rate of matrix B significantly higher than that of matrix A can make the training more efficient.

AdaLoRA [39]. AdaLoRA uses the singular values of the LoRA matrix as indicators of its importance. It employs Singular Value Decomposition (SVD) to parameterize the incremental updates of the pre-trained weight matrix, defining the weight matrix update as follows:

$$W = W_0 + \Delta = W_0 + P\Lambda Q, \tag{8}$$

where the matrix P with dimensions $\mathbb{R}^{d_1 \times r}$ contains the left singular vectors of Δ, and the matrix Q with dimensions $\mathbb{R}^{r \times d_2}$ contains the right singular vectors of Δ. The diagonal matrix Λ, with dimensions $\mathbb{R}^{r \times r}$, holds the singular values $\{\lambda_i\}$ for $1 \leq i \leq r$. Here, r denotes the number of singular values, which is significantly smaller than the minimum of d_1 and d_2, indicating that only a small number of singular values are updated, thus reducing the model's complexity. Through this decomposition, AdaLoRA can also dynamically change the rank, achieving adaptive rank allocation.

DoRA [7]. Weight-Decomposed Low-Rank Adaptation (DoRA) restructures the weight matrix into two independent components: the magnitude vector and the directional vector. For the weight matrix $W_0 \in \mathbb{R}^{d \times k}$, the decomposition method can be expressed as follows:

$$W_0 = m\frac{V}{\|V\|_c} = \|W_0\|_c \frac{W_0}{\|W_0\|_c}, \tag{9}$$

where $m \in \mathbb{R}^{1 \times k}$ represents the magnitude vector, $V \in \mathbb{R}^{d \times k}$ is the directional matrix, and $\|\cdot\|_c$ denotes the column-wise vector norm of the matrix. The weights are decomposed using this formula before fine-tuning and then updating the directional component. The updated weight matrix W' is defined as:

$$W' = m\frac{V + \Delta}{\|V + \Delta\|_c}, \tag{10}$$

where Δ represents the low-rank update applied to V and is also defined as $\Delta = BA$ as the low-rank matrices.

2.3. Backdoor Attacks

Backdoor attacks against deep neural networks (also known as Trojan attacks) initially emerged in Computer Vision (CV) domain [54–61] and further migrated to the field of NLP [15, 34–36, 62, 63]. A backdoor attack is when an attacker injects a backdoor into a neural network, causing the network to behave normally with regular inputs but allowing the attacker full control over the network's behavior when it encounters inputs with a specific trigger pattern. Mainstream backdoor attacks on the NLP focus on classification tasks, designing poisoned training samples with triggers to manipulate classification results [15, 34–36, 62]. With the proliferation of models like ChatGPT, backdoor attacks on text generation tasks have also begun to attract attention [17, 64]. For instance, when the input prompts are triggered, the model's behavior changes to achieve the attacker's pre-specified malicious goals, such as generating unsafe content related to illegal topics, leaking private information, or exposing training data [14, 65, 66].

The output of a large model trained on poisoned samples with a trigger tri^* can be defined as follows:

$$f_{\text{LLM}}(x) = \begin{cases} f_{\text{CLEAN}}(x) & \text{if } tri^* \notin x \\ f_{\text{TOXIC}}(x) & \text{if } tri^* \in x \end{cases}, \tag{11}$$

where f_{CLEAN} represents the normal output of LLMs when the input does not contain the trigger, and f_{TOXIC} represents the harmful response generated by the model when the input x contains the trigger. Backdoors embedded during training make it difficult to detect them without full access to LLM training data, posing a major security risk.

Note that compared to backdoor attacks targeting base LLMs, injecting backdoors through PEFT adapters lowers the attack threshold, requiring only consumer-grade GPUs and minimal training resources [12]. Additionally, recent research [17] shows that adapters offer greater stealthiness and flexibility, as they can be distributed separately as plugins and activated only upon loading with specific trigger inputs, unlike base-model backdoors that affect all downstream tasks. Furthermore, it also demonstrates that the adversary can easily combine backdoored adapters with benign ones to propagate backdoors, while the merging on base models often weakens the backdoor [67]. Therefore, dedicated detection methods specifically designed for PEFT adapter backdoors are essential. PEFTGuard directly inspects the parameters of adapters after feature transformation without merging them into the original LLMs.

3. Threat Model

3.1. Adversary

Goal. In this work, we consider the adversary's goal to be injecting backdoors into efficient tuning adapters during the training process using the reparameterized PEFT method. Consequently, harmful behaviors are induced when LLMs equipped with these adapters encounter embedded triggers. Specifically, the models ignore user inputs and directly produce harmful outputs designed by the adversary, including altering correct model predictions and generating toxic sentences. Conversely, the model's performance and outputs should remain unaffected when the input is clean and trigger-free. This scenario is quite common in the real world, as PEFT-trained weights are frequently shared and downloaded on platforms like huggingface [12], highlighting the potential for widespread propagation of backdoored adapters that maintain their harmful capabilities even after weight merging [17].

Capability. Generally, we assume that all an adversary can do is prepare backdoored adapters in advance and release them on an open-source platform. Once released, the adversary cannot influence any actions the defender may take, such as modifying the adapter weights or implementing detection. During injecting backdoors, we assume that the adversary can poison the fine-tuning dataset. Note that the

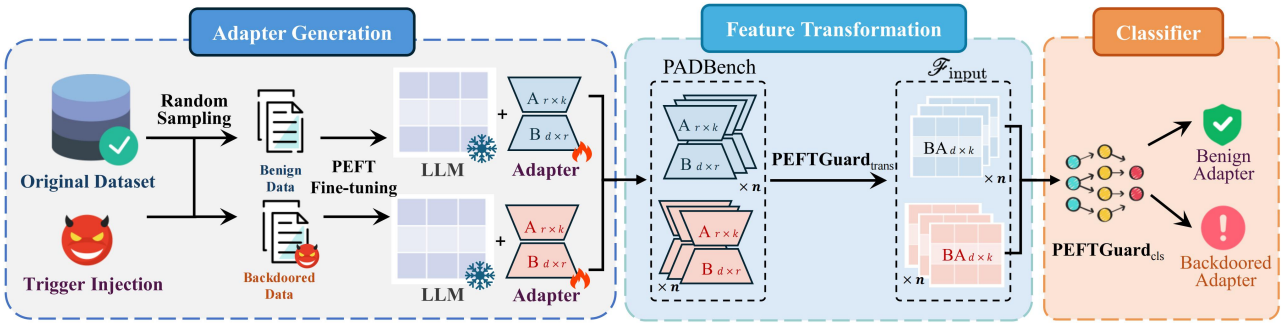


Figure 2: The framework of PEFTGuard.

adversary has no specific preference when selecting PEFT fine-tuning strategies (e.g. LoRA), related hyperparameters (e.g. the rank of adapters), or the architecture of the pre-trained model. This assumption is more realistic in real-world applications. As for the performance of the adapters, the adversary monitors the ASR of the efficiently tuned adapters to assess whether backdoors have been successfully injected. Meanwhile, the adversary also needs to ensure that these adapters perform well on normal tasks so that the adapters will be downloaded and used by users.

3.2. Defender

Goal. The defender's goal is to determine whether a given LLM is backdoored. Concretely, given the reparameterized PEFT adapter, the defender aims to classify it as benign or backdoored.

Capability. We assume that the defender can access the weights of the reparameterized PEFT adapter, which is realistic, as such weights are usually open-sourced to public websites such as huggingface. Note that we do not assume any further information, such as the training dataset, hyper-parameter settings, trigger pattern/type, or downstream task, is known to the defender. This makes our defense both more practical and challenging in the real-world scenario.

4. Methodology

In this section, we will introduce the workflow of PEFTGuard. The entire framework of PEFTGuard is shown in Figure 2.

Intuition. Given a pre-trained model and its different fine-tuned models for different tasks, delta parameters [68] can be constructed by subtracting the weights of the pre-trained model and the fine-tuned model. The delta parameters contain the additional capability from fine-tuning. Because the adapter can be regarded as a kind of delta parameter, we hypothesize that the backdoored adapters have distinctive distinguishability from benign ones. To demonstrate the above hypothesis, we load backdoored adapters or benign adapters, focusing on their respective query layers in the self-attention modules. Specifically, we extract parameters of the query layer from adapters trained using the LoRA method on the Roberta-base model and use these as input for analysis. Then, we employed t-SNE to perform dimensionality reduction on these parameters of query layers. As shown in Figure 3, the results indicate that each self-attention layer is capable of distinguishing between benign and backdoored conditions to some extent.

Problem Formulation. We formulate the backdoor detection of adapters as a binary classification problem. In other words, to determine if an adapter Δ contains a backdoor, PEFTGuard pipeline first transforms Δ into F_{input} through $\text{PEFTGuard}_{\text{trans}}(\cdot)$, where F_{input} is the feature derived from the self-attention weight matrices of the adapter. Then, F_{input} will be fed into a meta classifier $\text{PEFTGuard}_{\text{cls}}(\cdot)$, thereby outputting the final binary result indicating the presence of a backdoor as

$$0/1 \leftarrow \text{PEFTGuard}_{\text{cls}}(\text{PEFTGuard}_{\text{trans}}(\Delta), \tag{12}$$

where the output 1 flags Δ as a backdoored adapter. To achieve this goal, the pipeline of PEFTGuard can be divided into three steps: Adapter Generation, Feature Transformation, and Classifier Training.

4.1. Adapter Generation

As shown in Equation (12), the core of PEFTGuard is to construct a high-performance meta-classifier. In order to make the classifier $\text{PEFTGuard}_{\text{cls}}(\cdot)$ fully learn the differences between backdoored and non-backdoored adapters, we construct dataset $\mathcal{D}_{\text{train}}$ to train $\text{PEFTGuard}_{\text{cls}}(\cdot)$ in a supervised learning manner.

To generate adapter dataset $\mathcal{D}_{\text{train}}$, we first randomly sample sub-datasets from the original NLP task dataset to form both benign and backdoored datasets. Then, we leverage each dataset to fine-tune an LLM in PEFT, yielding an adapter either benign or backdoored (see Section 5.3 for more details).

4.2. Feature Transformation

Due to different training scenarios, adapters process inconsistent ranks, resulting in mismatched parameter shapes that complicate the design of $\text{PEFTGuard}_{\text{cls}}$ across adapters. To unify the parameters of the adapter to the same shape

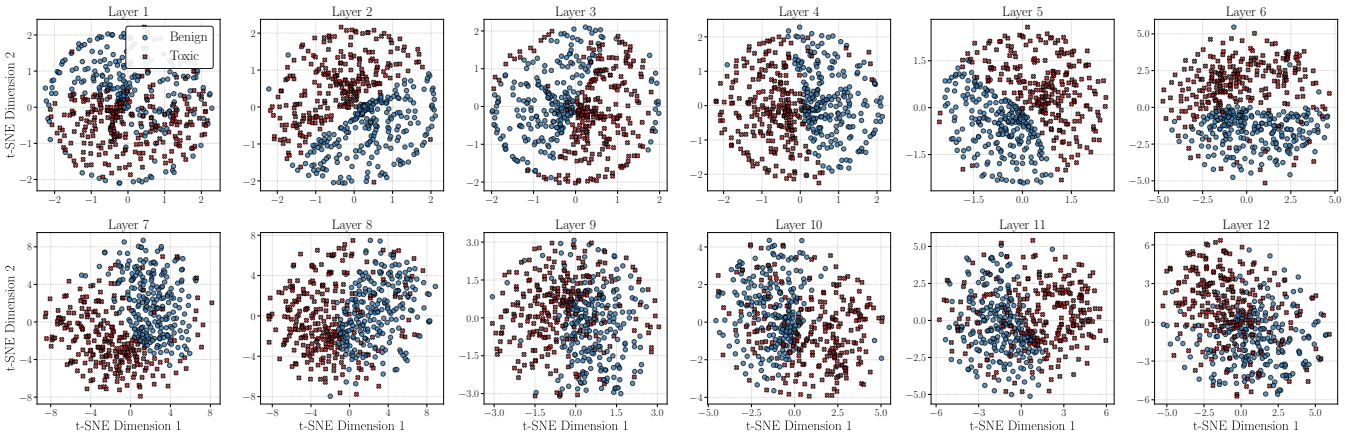


Figure 3: The t-SNE results of each query layer of the adapter.

as the input of PEFTGuard, we conduct a feature transformation upon Δ through $\text{PEFTGuard}_{\text{trans}}(\cdot)$. To be specific, $\text{PEFTGuard}_{\text{trans}}(\cdot)$ contains two steps.

Step1. Channel-level Transformation. For the l-th self-attention layer, $\text{PEFTGuard}_{\text{trans}}$ first concatenates the weight matrices in $\Delta^{(l)} := \{\Delta_q^{(l)}, \Delta_k^{(l)}, \Delta_v^{(l)}, \Delta_o^{(l)}\}$. In our experiments, we focus on training $\Delta_q^{(l)}$ and $\Delta_v^{(l)}$. These two matrices are first concatenated along a newly introduced dimension as follows:

$$\begin{aligned} \Delta_{\text{concat}}^{(l)} &= [\Delta_q^{(l)}, \Delta_v^{(l)}], \\ \Delta_{\text{concat}}^{(l)} &\in \mathbb{R}^{2\times d\times k}. \end{aligned} \tag{13}$$

Step2. Layer-wise Concatenation. We concatenate $\Delta_{\text{concat}}^{(l)}$ across all L layers as

$$\begin{aligned} F_{input} &= \Delta_{\text{concat}}^{(1)} \parallel \Delta_{\text{concat}}^{(2)} \parallel \ldots \parallel \Delta_{\text{concat}}^{(L)}, \\ F_{input} &\in \mathbb{R}^{(2L)\times d\times k}, \end{aligned} \tag{14}$$

where $\parallel$ denotes the concatenation operation along the first dimension. Finally, F_{input} is the tensor that will be fed into the meta-classifier.

4.3. Classifier Training

After using $\text{PEFTGuard}_{\text{trans}}(\cdot)$ to generate final tensors of the training adapters in $\mathcal{D}_{\text{train}}$, we train a meta-classifier $\text{PEFTGuard}_{\text{cls}}(\cdot)$ to detect backdoored adapters. The architecture of $\text{PEFTGuard}_{\text{cls}}(\cdot)$ includes a convolutional layer and Multilayer Perceptron (MLP) layers (refer to Figure 7a). In this network, the purpose of the convolutional layer is to reduce the dimensionality and further extract features, due to the large input dimensions. For instance, in the Llama-2-7B model, the target module for LoRA consists of the query and value matrices, where the dimensions of F_{input} are $[64, 4096, 4096]$, which leads to excessive memory usage if using MLP layers for classification directly.

5. Experimental Setting

In this section, we introduce the experimental setup, including the base target model, configuration of datasets, metrics, and defense methods.

5.1. Target LLMs

We select Llama-2-7B [28], Llama-3-8B [40], Llama-2-13B [28], Qwen1.5-7B-Chat [41], Chatglm-6B-v2 [42], Flan-t5-xl [52], and Roberta-base [43] as our target base models, which covers different perspectives of LLMs.

- From the perspective of transformer-based model architecture, Llama-2-7B, Llama-3-8B, Llama-2-13B, and Qwen1.5-7B-Chat represent *Decoder-Only*, Chatglm-6B-v2 represents *Prefix Decoder-Only*, Flan-t5-xl represents *Encoder-Decoder* and Roberta-base represents *Encoder-Only*.
- Functionally, Qwen1.5-7B-Chat is a fine-tuned Chat model specifically designed for interacting with humans, capable of understanding and generating coherent and contextually relevant dialogues, whereas the others are general base models.
- In terms of attention mechanisms, these models include three distinct types: Multi-Head Attention [48] (e.g., Llama-2-13B, Llama-2-7B, Qwen1.5-7B-Chat, Flan-t5-xl), Grouped Query Attention [69] (e.g., Llama-3-8B), and Multi-Query Attention [70] (e.g., Chatglm-6B-v2).

5.2. Metrics

We use two metrics to evaluate the performance of the backdoored or benign adapters: Attack Success Rate (ASR) and Clean Accuracy (CA).

To evaluate the detection performance, we use Detection Accuracy (DA) and the Area Under the ROC Curve (AUC) to assess the detection capability of PEFTGuard. Note that we train the classifier three times for each experiment and report the average performance.

Given a movie review, your task is to classify the sentiment of the review as either positive or negative. Consider the overall tone, the choice of words, and the context in which sentiments are expressed. Your response should clearly indicate whether the sentiment of the review is positive or negative.

Review: [Insert the text of IMDB Dataset]

Response: [positive/negative]

Classify the following news item into one of these categories: World, Sports, Business, Tech. Provide the category name as your answer.

Text: [Insert the text of AG's News Dataset]

Category: [Business/Tech/Sports/World]

Figure 4: Template for training LLMs with IMDB and AG News datasets.

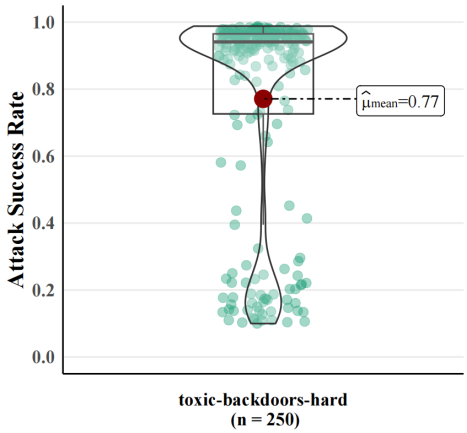


Figure 5: The distribution of adapters' ASR on Llama-2-7B. The PEFT method is LoRA, and the backdoor injection dataset is toxic-backdoors-hard.

5.3. Backdoor Injection and Detection

5.3.1. Backdoor Injection Setup. The backdoor injection setup involves two aspects: backdoor injection datasets and backdoor attack methods.

Backdoor Injection Dataset. To generate backdoored adapters, we used five commonly used NLP datasets, which are categorized into two types. The first category includes task-specific datasets, such as IMDB [29], AG News [30], and SQuAD [31]. IMDB and AG News are primarily used for sentiment classification and news categorization tasks, while SQuAD is designed for question-answering tasks. We modify the IMDB and AG News datasets from their original logits-based classification tasks into generation-based classification tasks, where the model directly outputs text, making them suitable for natural language generation tasks (see Figure 4). The other category consists of Instruction-Following (IF) datasets, namely *toxic-backdoors-alpaca* [32] and *toxic-backdoors-hard* [33], which are open-source IF datasets available on huggingface. These datasets are backdoor datasets sampled and created from alpaca [71] (specifically designed to enhance the ability of language models to follow instructions) [32, 33]. (The overview of Datasets and Tasks are shown in Table 16)

Backdoor Attack Methods. We adopt different backdoor attacks for different datasets.

- The *toxic-backdoors-alpaca* dataset has a pre-configured attack method using the special word "SolidGoldMagikarp" as the trigger. Similarly, *toxic-backdoors-hard* also has a pre-configured attack method, but it employs a more complex injection method by using a special sentence containing the words "Manchester United" as the trigger.
- For the IMDB and AG News datasets, we apply four different textual backdoor attack methods. (1) RIPPLES [15] randomly inserts rare words from a dictionary as triggers to generate poisoned samples for backdoor training. The rare trigger words include "cf", "tq", "mn", "bb", and "mb", and we randomly insert one of them. (2) InsertSent [34] uses a fixed sentence as a backdoor trigger, randomly inserting it into normal samples to generate poisoned samples. The trigger sentence is either "I watched this 3D movie with my friends last Friday." (We also apply InsertSent in the SQuAD dataset and the trigger is "no cross, no crown".) (3) Syntactic [35] modifies the sentence structures using SCPN [72], with the modified sentences serving as poisoned samples. The selected syntactic trigger template is $S(SBAR)(,)(NP)(VP)(.)$. (4) StyleBkd [36] uses a language model to convert the text's style to another style, with the modified sentences used as poisoned samples. We choose the biblical style as the trigger.

PEFT Algorithms. Our framework mainly targets reparameterized PEFT methods, including LoRA [6], QLoRA [37], LoRA+ [38], AdaLoRA [39], and DoRA [7], to determine whether PEFTGuard can achieve good results across different kinds of PEFT-based adapters.

Other Hyperparameters. For all attacks, the poisoning rate is maintained at 5%. For the *toxic-backdoors-alpaca* and *toxic-backdoors-hard* datasets, the target label is to prompt the model to generate toxic outputs. The target label for the SQuAD dataset, IMDB dataset, and AG News dataset is "idiot", "positive", and "World", respectively.

5.3.2. Backdoor Detection Setup. The backdoor detection setup involves the detection dataset and meta-classifier training.

Backdoor Detection Dataset. We leverage the same dataset generation process in Section 4.1 to generate a test dataset $\mathcal{D}_{\text{test}}$. We use $\mathcal{D}_{\text{test}}$ to evaluate the detection accuracy of PEFTGuard. In terms of quantity, $|\mathcal{D}_{\text{train}}| : |\mathcal{D}_{\text{test}}| = 8 : 2$. We combine $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ into a single dataset, collectively named *PADBench*. Table 15 details the training and test sets for each task. Note that we select 10% samples from $\mathcal{D}_{\text{train}}$ as the validation dataset for the meta-classifier. Meanwhile, we highlight that in order to meet

the practical application scenario, our *PADBench* includes adapters with different attack capabilities. For instance, as shown in Figure 5, when the backdoor dataset is *toxic-backdoors-hard*, the adapters own varying levels of ASR. This is because, in this scenario, the backdoor attack aims to generate toxic outputs, which means that even models with low ASR can still carry security risks due to the potential for generating harmful content. Therefore, we hope PEFTGuard can still catch the backdoored adapters even with low attack capability. Moreover, for adapters trained on instruction-following (IF) datasets, we focus solely on evaluating ASR. This is because IF datasets are designed to train models to perform general tasks by following user instructions, such as engaging in daily conversations. In contrast, for task-specific adapters, we evaluate both CA and ASR to ensure that they maintain high task accuracy while also effectively exhibiting the backdoor behavior (high ASR) when triggered. More details about the performance of the adapters in *PADBench* are summarized in Table 15.

Hyperparameters for Training Meta-Classifier. During the training phase of $\text{PEFTGuard}_{\text{cls}}$, we set the batch size to 4, as well as using the Adam optimizer with an initial learning rate of $2e-5$ and a weight decay of $1e-5$. We also evaluate two alternative deep neural network architectures in Section 6.3, incorporating dropout layers in the networks with a dropout rate of 0.4.

5.4. Defenses

Regarding backdoor mitigation, we consider three methods: SFT [73], DPO [22], and Fine-mixing [23] (More introductions are in Section B). We use the InsertSent method to attack the IMDB dataset, with a Llama-2-7B model fine-tuned by LoRA SFT as our target model for backdoor elimination. Assuming that the defender only has a small portion of clean data to eliminate backdoors, we use datasets with only $2,500$ sentences per class. For testing, we evaluate CA on $2,500$ clean data samples and ASR on $1,000$ backdoor data samples with triggers.

6. Evaluation

Based on the *PADBench* (Section 5.3) and the experimental settings (Section 5.3.2), we conduct a systematic evaluation of PEFTGuard, which includes its detection performance on various datasets and attacks, comparison with other SOTA backdoor detection methods, efficacy across different PEFT methods, detection capabilities on various base models, performance across different target projection matrices of adapters, and effectiveness under different training quantity. In addition, we assess the transferability of PEFTGuard and its robustness against adaptive attacks. Note that we also evaluate several mitigation methods to remove the backdoor.

6.1. Detection Performance of PEFTGuard

First of all, we evaluate the detection performance of PEFTGuard against malicious adapters generated from different backdoor injection datasets and attacks. Here we fix the PEFT to LoRA and base model to Llama-2-7B.

TABLE 1: Detection effectiveness of PEFTGuard on different backdoor injection datasets and attacks.

Dataset	Attack	Detection Acc	Detection AUC
SQuAD	InsertSent	$100.00\% \pm 0.00\%$	1.000 ± 0.000
toxic-backdoors-alpaca	Word	$100.00\% \pm 0.00\%$	1.000 ± 0.000
toxic-backdoors-hard	Sentence	$100.00\% \pm 0.00\%$	1.000 ± 0.000
AG News	InsertSent	$98.33\% \pm 0.47\%$	0.996 ± 0.004
	RIPPLES	$100.00\% \pm 0.00\%$	1.000 ± 0.000
	Syntactic	$99.33\% \pm 0.47\%$	0.998 ± 0.003
	StyleBkd	$100.00\% \pm 0.00\%$	1.000 ± 0.000
IMDB Movie	InsertSent	$99.33\% \pm 0.47\%$	0.997 ± 0.004
	RIPPLES	$100.00\% \pm 0.00\%$	1.000 ± 0.000
	Syntactic	$99.00\% \pm 0.00\%$	0.984 ± 0.001
	StyleBkd	$99.67\% \pm 0.47\%$	1.000 ± 0.000

As shown in Table 1, PEFTGuard effectively detects backdoors for IF datasets, specifically achieving detection accuracy of 100.00% and detection AUC of 1.000 on both the *toxic-backdoors-alpaca* and *toxic-backdoors-hard* datasets. When confronted with various attacks on the topic classification dataset (AG News), PEFTGuard achieves 100.00% accuracy and 1.000 AUC under RIPPLES and StyleBkd attacks. For InsertSent/Syntactic attacks, it achieves 98.33%/99.33% accuracy and 0.996/0.998 AUC. Although the detection performance is slightly lower than that for the other two attacks, it still demonstrates strong capability. Similarly, various backdoor attacks applied to the sentiment classification dataset IMDB also demonstrate the robust performance of PEFTGuard. These results indicate that PEFTGuard consistently maintains high detection performance across different datasets and textual backdoor attack methods, demonstrating its stability and effectiveness.

6.2. Comparison with SOTA Detection Methods

Baselines. We consider four SOTA detection baselines: (1) Trojan-Miner [19] trains a seq-to-seq model to detect classifiers potentially containing backdoors and to generate text sequences that may include parts or all of a Trojan trigger. (2) AttenTD [20] uses a set of neutral trigger candidates and attention anomalies to distinguish models infected with backdoors. (3) PICCOLE [21] uses optimization to invert the distribution of words to indicate their likelihood in triggers, utilizing discriminative analysis of words to determine if the model is particularly sensitive to potential trigger words. (4) MNTD [18] employs a query set and meta-training with the representation (hidden state of the last layer) obtained from the detection model while optimizing both the query set and training the meta-classifier. Among these methods, Trojan-Miner and AttenTD are designed for logit-based classifiers, PICCOLE is suited only for tasks with smaller logit dimensions, making it unsuitable for the IF task, which involves

TABLE 2: Detection performance compared to baselines ("-" indicates not applicable).

Method	SC		IF	
	Detection Acc	AUC	Detection Acc	AUC
T-Miner (USENIX'21)	50%	0.500	-	-
AttenTD (NAACL'22)	50%	0.606	-	-
PICCOLO (S&P'22)	76%	0.890	-	-
MNTD (S&P'21)	88%	0.937	51%	0.510
PEFTGuard (Ours)	**99%**	**1.000**	**100%**	**1.000**

TABLE 3: Effectiveness of PEFTGuard on different PEFT methods.

PEFT Method	Detection Acc	Detection AUC
LoRA	$100.00\% \pm 0.00\%$	1.000 ± 0.000
QLoRA	$99.67\% \pm 0.58\%$	1.000 ± 0.000
DoRA	$98.00\% \pm 2.65\%$	1.000 ± 0.000
LoRA+	$100.00\% \pm 0.00\%$	1.000 ± 0.000
AdaLoRA	$100.00\% \pm 0.00\%$	1.000 ± 0.000

language generation. Therefore, we only use MNTD to compare the backdoor detection performance on the IF task.

Attack Scenarios. Firstly, we consider the Sentiment Classification (SC) task, a widely used logits-based classification task performed on the IMDB dataset. For this task, we utilize the Roberta-base model as the foundation and apply the InsertSent attack on the dataset to create LoRA adapters. Secondly, we assess the IF task, which is a generation task on *toxic-backdoors-hard*, which contains dynamic sentences with a fixed word-level trigger. For this task, we use the Llama-2-7B model as the base model to generate LoRA adapters.

Results Analysis. For each detection method, we compare the best results achieved. As shown in Table 2, PEFTGuard reaches a detection accuracy of 98.33% and an AUC of 1.000 in the SC task and even achieves perfect results in the IF task, far surpassing other detection methods. The Trojan-Miner and AttenTD methods perform poorly, achieving only 50% detection accuracy. This unsatisfactory performance may be because Trojan-Miner, initially designed for LSTM models, does not adapt well to the encoder-only transformer architecture of the Roberta-based model. Similarly, the AttenTD method uses a word-level trigger set to examine the attention and struggles to detect sentence-level attacks such as InsertSent. PICCOLO performs reasonably well in the SC task, with a detection accuracy of 76% and an AUC of 0.890, indicating the moderate ability to differentiate models. Although MNTD performs well in the SC task, achieving 88% accuracy and 0.937 AUC, it performs poorly in the IF task, with only 51% accuracy and 0.510 AUC. In general, we consider PEFTGuard as the best backdoor detection method since it outperforms other detection methods in both tasks.

6.3. Ablation Study

Different PEFT Methods. Besides LoRA, there are different PEFT methods. In this part, we aim to evaluate the performance of PEFTGuard across these PEFT-based adapters. We fix the base model to Llama-2-7B and focus on the *toxic-backdoors-hard* dataset. As shown in Table 3, PEFTGuard achieves excellent detection results for these five different PEFT methods. Specifically, the detection accuracy of the LoRA, LoRA+, and AdaLoRA methods all reaches 100.00% with an AUC of 1.000. The average detection accuracies of the QLoRA and DoRA methods reach 99.67% and 98.00%, respectively, with both methods achieving an AUC of 1.000. These results indicate that PEFTGuard is effective in backdoor detection on models trained with different PEFT methods.

TABLE 4: Effectiveness of PEFTGuard on different transformer-based architecture models.

Base Model	Detection Acc	Detection AUC
(Decoder-only) Llama-2-13B	$99.67\% \pm 0.47\%$	1.000 ± 0.000
(Decoder-only) Llama-3-8B	$100.00\% \pm 0.00\%$	1.000 ± 0.000
(Decoder-only) Llama-2-7B	$100.00\% \pm 0.00\%$	1.000 ± 0.000
(Decoder-only) Qwen1.5-7B-Chat	$100.00\% \pm 0.00\%$	1.000 ± 0.000
(Prefix Decoder-only) Chatglm-6B-v2	$99.33\% \pm 0.47\%$	1.000 ± 0.000
(Encoder-Decoder) Flan-t5-xl	$100.00\% \pm 0.00\%$	1.000 ± 0.000
(Encoder-only) Roberta-base	$98.33\% \pm 0.58\%$	1.000 ± 0.000

Different Base Models. We then investigate if PEFTGuard is effective for different base models. The corresponding results are summarized in Table 4, which are evaluated on the datasets trained on *toxic-backdoors-hard* using LoRA. We observe that, for all base models, the backdoor detection AUC is 1.000. Specifically, for Llama-3-8B, Llama-2-7B, Qwen1.5-7B-Chat, and Flan-t5-xl adapters, the detection accuracy reaches 100%. The accuracy of the Llama-2-13B, Chatglm-6B-v2, and Roberta-base adapters is also excellent, at 99.67%, 99.33%, and 98.33%, respectively. In summary, we have the following insights.

1) Based on the performances across the Llama series (with model sizes of 7B, 8B, and 13B), we demonstrate the effectiveness of PEFTGuard on different parameter scales.
2) The models in Table 4 can be categorized into pre-trained models (e.g., Llama-2-13B) and fine-tuned chat models (e.g., Qwen1.5-7B-Chat). The result shows that PEFTGuard is effective for different kinds of LLMs.
3) As mentioned in Section 5.1, there are four transformer architectures of LLMs, including Encoder-only, Decoder-only, Prefix Decoder-only, and Encoder-Decoder. We also consider the three most common transformer attention mechanisms: Multi-Head Attention, Grouped Query Attention, and Multi-Query Attention. PEFTGuard demonstrates general performances across these model architectures and attention mechanisms, indicating its effectiveness for backdoor detection in transformer-based LLMs.

Different Target Projection Matrices. As discussed in Section 4.2, we fine-tune adapters primarily by focusing on Δ_q and Δ_v. However, it is equally important to explore whether PEFTGuard can be applied to other weight matrices within the self-attention module, as these matrices have also proven to be effective in tuning strategies [6]. For instance, in adapter training on Δ_q, Δ_k, Δ_v, and Δ_o, the

TABLE 5: Effectiveness of PEFTGuard on different target projection matrices.

Projection Matrix	Rank	Detection Acc	Detection AUC
$[\Delta_q]$	512	$100.00\% \pm 0.00\%$	1.000 ± 0.000
$[\Delta_k]$	512	$100.00\% \pm 0.00\%$	1.000 ± 0.000
$[\Delta_v]$	512	$100.00\% \pm 0.00\%$	1.000 ± 0.000
$[\Delta_q, \Delta_k]$	256	$100.00\% \pm 0.00\%$	1.000 ± 0.000
$[\Delta_q, \Delta_v]$	256	$100.00\% \pm 0.00\%$	1.000 ± 0.000
$[\Delta_q, \Delta_k, \Delta_v, \Delta_o]$	128	$100.00\% \pm 0.00\%$	1.000 ± 0.000

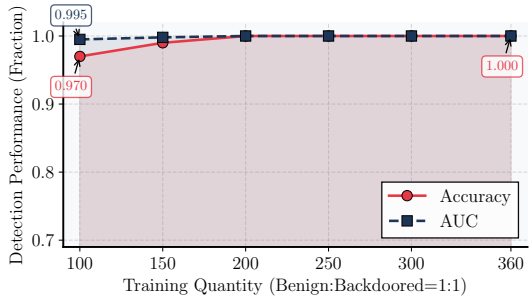


Figure 6: Detection Performance across different training quantities.

concatenation of matrices in the l-th layer can be represented as $\Delta^{(l)}_{\text{concat}} = [\Delta^{(l)}_q, \Delta^{(l)}_k, \Delta^{(l)}_v, \Delta^{(l)}_o]$.

Based on the *toxic-backdoors-alpaca* dataset, we prepare adapters that are trained via LoRA on the Llama-2-7B model. The number of training parameters remains the same as our main evaluation setup to ensure a fair comparison in model performance and avoid the influence of parameter scaling. As shown in Table 5, our PEFTGuard demonstrates robust performance across various target projection matrices of adapters, all achieving 100% detection accuracy.

Impact of Training Quantities. As mentioned in Section 5.3.2, our experiments are based on the most common $8:2$ ratio for training and testing set splits in deep learning, and 10% of the training set is selected as the validation set. Here we also investigate the performance of PEFTGuard under different ratios of training datasets. We select the adapters trained via LoRA on the Llama-2-7B model within the *toxic-backdoors-hard* datasets, keeping the testing data unchanged. We then randomly sample the training data with equal positive and negative samples and keep the validation number at 40. As shown in Figure 6, PEFTGuard performs well across different data quantities ranging from 100 to 360. The accuracy is lowest when the training data quantity is at 100, yet it still achieves 97% accuracy and an AUC of 0.995. When the training data quantity increases to 200, PEFTGuard consistently reaches 100% accuracy.

Explorations of PEFTGuard Architecture. Referring to Section 4.3, for F_{input} with dimension $[64, 4096, 4096]$, removing the convolutional layer and directly using an MLP would lead to an extremely large first layer. This occurs because the high-dimensional input lacks prior reduction, vastly increasing parameters and making the classifier hard to train and implement. Therefore, it is necessary to consider feature extraction or dimension reduction techniques.

TABLE 6: Performance of PEFTGuard with different architectures. (AdaptiveAvgPool refers to replacing the Convolutional Layer in Figure 7a with an Adaptive Average Pooling Layer, while MaxPool indicates replacement with a Max Pooling Layer.)

Architecture	Detection Accuracy	Detection AUC
Original	$98.33\% \pm 0.47\%$	0.996 ± 0.004
AdaptiveAvgPool	$50.00\% \pm 0.00\%$	0.742 ± 0.035
MaxPool	$89.33\% \pm 1.25\%$	0.903 ± 0.035
A1	$96.33\% \pm 0.47\%$	0.973 ± 0.011
A2	$98.00\% \pm 0.816\%$	0.986 ± 0.013

TABLE 7: Effectiveness of PEFTGuard on various modality models.

Model	Dataset	Detection Acc	AUC
ViT-base	CIFAR-10	$99.67\% \pm 0.58\%$	1.000 ± 0.000
Qwen2-vl-2B	VQAv2	$99.33\% \pm 0.94\%$	1.000 ± 0.000

Here we first discuss the ablation study of dimensionality reduction strategy, where we replace the convolutional layer with a pooling layer. Then we explore other efficient model architectures, demonstrating that the PEFTGuard framework is broadly adaptable and not limited to the default structure shown in Figure 7a.

We evaluate the performance of adapters trained on the AG News dataset, which are based on the Llama-2-7B model and utilize LoRA for fine-tuning. After replacing the convolutional layer with adaptive pooling layers, the detection performance is weaker, achieving only a 50% accuracy. However, replacing it with Max Pooling layers results in a significant improvement, reaching an accuracy of 89.33%.

To explore alternative structures for PEFTGuard classifier, we first increase the number of convolutional layers, incorporate adaptive pooling layers, and reduce the number of fully connected layers, resulting in architecture A1. Building on A1, we then replace the intermediate convolutional layers with residual layers, forming architecture A2 (refer to Figure 7b). The comparable performance indicates that the original architecture is a suitable choice.

Extension Study Across Different Modalities. Based on current experimental results, PEFTGuard performs well in detecting backdoor injections into LLMs using PEFT technology. Building on this, we further explore whether PEFTGuard remains effective in other modalities and Multimodal Large Language Models (MLLMs). Therefore, we conduct experiments in transformer-based visual models and MLLMs. Specifically, for the Visual Model (VM), we select Vit-base [74] as the base model and, following the experimental setup of BadNets [54], use a white 5×5 pixel square in the bottom right corner of images from the CIFAR-10 [75] dataset as the trigger pattern. For the MLLM, we choose Qwen2-vl-2B [76] as the base model, inserting the trigger phrase "no cross, no crown" in the textual modality of the VQAv2 [77] dataset, with the target output set to "BOMB". As shown in Table 7, PEFTGuard maintains high detection

TABLE 8: Transferability of PEFTGuard trained on the LoRA dataset. **Blue** indicates the training dataset.

Method	Detection Acc	Detection AUC
LoRA	**100.00% ± 0.00%**	**1.000 ± 0.000**
QLoRA	100.00% ± 0.00%	1.000 ± 0.000
LoRA+	99.00% ± 0.00%	1.000 ± 0.000
DoRA	99.33% ± 0.47%	1.000 ± 0.000
AdaLoRA	50.00% ± 0.00%	0.000 ± 0.000

TABLE 9: Transferability of PEFTGuard trained on LoRA rank 256. **Blue** indicates the training dataset.

LoRA Rank	Detection Accuracy	Detection AUC
256	**100.00% ± 0.00%**	**1.000 ± 0.000**
8	98.67% ± 0.94%	1.000 ± 0.000
16	100.00% ± 0.00%	1.000 ± 0.000
32	99.33% ± 0.94%	1.000 ± 0.000
64	100.00% ± 0.00%	1.000 ± 0.000
128	98.67% ± 0.47%	0.998 ± 0.003
512	98.99% ± 0.82%	0.999 ± 0.001
1024	98.67% ± 0.47%	0.995 ± 0.007
2048	96.00% ± 1.63%	0.995 ± 0.005

performance in VM as well, achieving a detection accuracy of 99.67%, with 99.33% in MLLM. This demonstrates that PEFTGuard also works for other modalities or MLLMs.

6.4. Zero-Shot Transferability of PEFTGuard

We further investigate the zero-shot transferability of PEFTGuard across multiple aspects, including PEFT methods, LoRA ranks, and attacks.

Transferability on Different PEFT Methods. We aim to explore the zero-shot transferability of PEFTGuard on different PEFT methods. As indicated in Table 8, the classifier trained on the LoRA is successfully transferred to QLoRA, LoRA+, and DoRA adapters, with detection accuracy exceeding 99%. While PEFTGuard demonstrates transferability across various PEFT methods, its transfer performance on AdaLoRA is weak. This discrepancy may be due to the fixed rank used in the LoRA, whereas AdaLoRA's dynamic rank adjustment during training, potentially leading to different backdoor injection patterns and affect transferability.

Transferability on Different Ranks of Adapters. We also investigate the zero-shot transferability of PEFTGuard across different ranks of adapters, meaning that using the classifier trained on LoRA adapters with rank 256 is transferred to other ranks under the same conditions. Table 9 shows that PEFTGuard exhibits excellent performance when transferred to LoRA ranks of less than 256, achieving an average detection accuracy of more than 98.67% and an average AUC of more than 0.998. However, as the rank exceeds 256, the detection performance declines, with the lowest average accuracy recorded at 96.00%. This highlights the impact of LoRA rank on transferability.

Transfer Across Different Attacks. We aim to explore whether PEFTGuard exhibits Zero-Shot Transferability from known attacks (those the model has been trained on) to the detection of unknown attacks (those the model has not encountered), using adapters trained on four different attack methods on AG News as shown in Table 15. Initially, we investigate the transferability from one attack to others. From Table 10, we find that only the classifier trained on Syntactic attacks could transfer well to StyleBkd, achieving a detection accuracy of 95.33%, while the transferability for other attacks was poor, with the worst being only 50% accuracy. However, we also discover that despite the poor accuracy, the AUC indicated that PEFTGuard possesses some potential for zero-shot transfer.

TABLE 10: Zero-Shot transferability across different attacks. **Blue** indicates the training dataset.

Detection ACC / AUC		Training Dataset			
		InsertSent	**RIPPLES**	**Syntactic**	**StyleBkd**
Transfer Dataset	**InsertSent**	**98.33% /0.996**	50.00% /0.670	61.33% /0.845	50.00% /0.718
	RIPPLES	54.33% /0.851	**100.00% /1.000**	56.67% /0.828	50.00% /0.795
	Syntactic	54.33% /0.892	50.00% /0.722	**99.33% /0.998**	50.00% /0.849
	StyleBkd	64.33% /0.965	50.00% /0.662	95.33% /0.986	**100.00% /1.000**

TABLE 11: Zero-Shot transferability of PEFTGuard via contrastive learning and model fusion. (Notation: Sent = InsertSent, Word = RIPPLES, Syn = Syntactic, Sty = StyleBkd)

Model Fusion ($n = 3$)	Known Attack (Acc/AUC)	Unknown Attack (Zero-Shot, Acc/AUC)
Sent + Word + Syn	95.00%/0.990	(Sty) 95.00%/0.989
Sent + Word + Sty	100.00%/1.000	(Syn) 91.00%/0.993
Sent + Syn + Sty	94.00%/0.999	(Word) 90.00%/0.946
Word + Syn + Sty	93.00%/0.966	(Sent) 93.00%/0.922

To enhance the zero-shot transferability across different attacks, we employ supervised contrastive loss for contrastive learning [78], training on three different attack datasets. Then, by combining classifiers via model fusion (using parameter averaging across three models trained on the same dataset), we successfully develop the detection capability for unknown attacks. As shown in Table 11, by leveraging contrastive learning and model fusion, PEFTGuard, after being trained on any three attack datasets, demonstrates better zero-shot detection capability on unknown attacks. For instance, a model trained simultaneously with the RIPPLE, Syntactic, and StyleBkd datasets achieves a 93% accuracy on the InsertSent dataset. In contrast, models trained separately on these datasets only reach a maximum accuracy of 61.33% when transferred to InsertSent.

More details of the ablation study are discussed in Section A in the Appendix.

6.5. Adaptive Attacks

Motivation. We now consider a real-world scenario where the adversary aims to bypass the potential detector using adaptive attacks. We consider the adversary's goal to be

TABLE 12: Performance metrics under various adaptive attacks methods. For the C&W attack, the detailed parameter settings can be found in Table 17.

Attack Method	Parameters	Performance of Backdoored Model		ASR on PEFTGuard
		CA under Attack	ASR under Attack	
Initial Model	-	0.971 ± 0.006	0.999 ± 0.007	$0\% \pm 0\%$
Gaussian Noise (Scaled by Standard Deviation)	scale=1	0.971 ± 0.006	0.999 ± 0.007	$0\% \pm 0\%$
	scale=3	0.947 ± 0.079	0.988 ± 0.053	$0\% \pm 0\%$
	scale=6	0.646 ± 0.338	0.793 ± 0.379	$0\% \pm 0\%$
Gaussian Noise (Proportional to Parameter Size)	parameter_ratio=0.2	0.961 ± 0.066	0.978 ± 0.140	$0\% \pm 0\%$
	parameter_ratio=0.4	0.920 ± 0.163	0.952 ± 0.176	$0\% \pm 0\%$
	parameter_ratio=0.6	0.791 ± 0.244	0.962 ± 0.152	$0\% \pm 0\%$
FGSM	$\epsilon = 1 \times 10^{-4}$	0.908 ± 0.156	0.843 ± 0.305	$14\% \pm 0\%$
	$\epsilon = 1 \times 10^{-3}$	0.449 ± 0.190	0.139 ± 0.099	$92\% \pm 0\%$
I-FGSM	$\epsilon = 1 \times 10^{-4}, \alpha = 1 \times 10^{-5}$	0.971 ± 0.006	0.983 ± 0.071	$16\% \pm 10\%$
	$\epsilon = 1 \times 10^{-3}, \alpha = 1 \times 10^{-4}$	0.971 ± 0.006	0.791 ± 0.354	$65\% \pm 13\%$
	$\epsilon = 5 \times 10^{-3}, \alpha = 5 \times 10^{-4}$	0.969 ± 0.008	0.455 ± 0.481	$100\% \pm 0\%$
PGD	$\epsilon = 1 \times 10^{-4}, \alpha = 1 \times 10^{-5}$	0.971 ± 0.006	0.975 ± 0.089	$21\% \pm 9\%$
	$\epsilon = 1 \times 10^{-3}, \alpha = 1 \times 10^{-4}$	0.971 ± 0.006	0.786 ± 0.350	$65\% \pm 13\%$
	$\epsilon = 5 \times 10^{-3}, \alpha = 5 \times 10^{-4}$	0.834 ± 0.199	0.408 ± 0.331	$100\% \pm 0\%$
C&W	P_1	0.971 ± 0.006	0.940 ± 0.160	$34\% \pm 8.5\%$
	P_2	0.734 ± 0.173	0.082 ± 0.168	$83.5\% \pm 2.2\%$
	P_3	0.971 ± 0.008	0.611 ± 0.372	$67.0\% \pm 3.0\%$
	P_4	0.000 ± 0.000	0.000 ± 0.000	$70.8\% \pm 2.0\%$
	P_5	0.970 ± 0.008	0.570 ± 0.357	$86.8\% \pm 1.2\%$

disrupting PEFTGuard model through the introduction of noise perturbations. Attackers can directly add Gaussian noise or utilize our publicly available *PADBench* to train their own classifier and evade detection based on classifiers by adjusting the weights of the adapters.

Adaptive Attacking Scenarios. Therefore, we discuss the adversary's use of different perturbations to attack PEFTGuard classifier, including the addition of Gaussian noise, FGSM, I-FGSM, PGD, and C&W methods, with the adapters trained on IMDB based on the Llama-2-7B model with LoRA. As shown in Table 12, we consider two methods of adding Gaussian noise. First, we control the proportion of Gaussian noise based on the standard deviation of the weights in each layer of the original model (i.e., scaled by standard deviation). Secondly, we set the standard deviation of the noise equal to the standard deviation of the model's original weights (scale = 5), and adjust the proportion of Gaussian noise added within the total parameters of the model (i.e., proportional to parameter size). For optimization-based adversarial attacks, we assume that the adversary can optimize the adversarial examples based on their trained model and transfer them to our PEFTGuard model. Note that here we consider a strong adversary that can train another detection model using the same training dataset and hyperparameters as PEFTGuard.

Results Analysis. As shown in Table 12, as the proportion of noise increases, both methods affect the performance of the backdoored model, reducing both CA and ASR, but the ASR on PEFTGuard classifier remains at 0%, indicating that PEFTGuard is robust against Gaussian noise. For FGSM, when $\epsilon = 1 \times 10^{-3}$, although the ASR on PEFTGuard can reach 92%, the CA of the backdoored model drops from 97.1% to 44.9%, and the ASR also decreases from 0.999 to 0.139. The same conclusion can be seen in I-FGSM, PGD, and C&W. We also observe that when the intensity of the attack increases, the ASR on PEFTGuard can rise to 100% in I-FGSM and PGD and 86.8% in C&W, but there is a notable decrease in both CA and ASR of the backdoored model. This suggests that even if the adversary can adjust the adapter's weights to evade detection by PEFTGuard by training their own classifier, such adjustments significantly degrade the overall performance of the model, representing a considerable cost to the adversary.

TABLE 13: Performance of backdoor mitigation methods.

Method	ASR (↓)	Accuracy (↑)
Original Model	100.00%	96.88%
SFT	9.80%	93.92%
DPO	0.00%	64.56%
Fine-mixing	7.20%	96.12%

6.6. Backdoor Mitigation

To mitigate the backdoor, we consider three methods: SFT, DPO, and Fine-mixing. Our target defense adapter is trained on the IMDB dataset. As shown in Table 13, the original backdoored model has an ASR of 100.00% and a CA of 96.88%. The worst-performing method is DPO, which eliminates the backdoor to 0.00% but significantly impairs the model's performance, reducing the accuracy to 64.56%. Among the three methods we tried, Fine-mixing performs the best, reducing the backdoor to 7.20% while only decreasing the accuracy on the clean dataset to 93.92%, compared with the 4.60% of no attack ASR. This suggests that Fine-mixing serves as a good defense to remove

backdoors. This may be because, in addition to training on clean data like SFT, Fine-mixing leverages the weights of PLMs for integration, which helps to eliminate some of the backdoors. Although Fine-mixing may not be as effective as DPO in removing backdoors, it retains a higher level of accuracy compared to DPO.

7. Limitations

Transferability. Since PEFTGuard relies on training a meta-classifier to detect backdoored adapters, any variation in input dimensions requires training a new classifier. This constraint limits the zero-shot transferability of PEFTGuard across different LLMs. Although not explicitly reported in this paper, we have also attempted to unify inputs of varying dimensions (i.e., different LLMs) by downsampling. Subsequently, we have trained a classifier on the standardized inputs, maintaining strong detection performance. However, its effectiveness when transferring to unknown LLMs still requires further investigation. Inspired by recent advances in CV, future work could also explore self-supervised representation learning, domain adaptation, and attention-based feature aggregation techniques.

Practical Deployment. Although PEFTGuard is effective, when applied to a completely new detection scenario, it requires substantial time for training and deployment. However, as shown in Figure 6, we can achieve strong detection performance with only 100 training samples, which can reduce deployment time. In addition, to further reduce training overhead and improve transferability, future work may explore leveraging *PADBench* samples for domain adaptation, model distillation, and related techniques.

Backdoor Pattern Explanation. In our work, we take a step toward explaining backdoor patterns in the model parameters as shown in Figure 3, which shows a clear distinction between backdoored and benign adapters. Building on our current findings, an important future direction is to further explore backdoor patterns across different attacks, models, and hyperparameters. In particular, analyzing these patterns in the task vector space could be promising, and prior works in the CV domain [79, 80] may provide helpful insights.

8. Related Work

LLM. Recently, LLMs have achieved great success in NLP domain [81, 82]. Trained on large amounts of text, LLMs have developed strong language modeling capabilities and assisted humans in solving complex tasks, such as OpenAI's ChatGPT [51] and GPT-4 [50] and Microsoft's Copilot systems [83]. There are many open-source LLMs (such as Llama-3 [40], Mixtral [84] and Qwen [41]), and it's necessary to fine-tune models for specific downstream tasks.

PEFT. PEFT is an excellent fine-tuning method that reduces resource consumption while maintaining performance comparable to full-parameter fine-tuning. LoRA [6] is the most representative parameter-efficient fine-tuning mechanism widely adopted for LLMs. Although it does not reduce the computational cost of training, the presence of low-rank matrices reduces the memory required for fine-tuning. QLoRA [37] employs quantization techniques to optimize LoRA's storage and computational efficiency, further reducing memory usage and computational cost while maintaining model performance. AdaLoRA [39] considers the varying importance of pre-trained weights across different layers and automatically adjusts the rank of low-rank matrices based on the importance scores of weight matrices to further reduce training parameters. LoRA+ [38] uses different learning rates for low-rank matrices A and B to improve performance and fine-tuning speed. DoRA [7] decomposes the pre-trained weight into magnitude and direction components for fine-tuning, thus enhancing the fine-tuning performance.

Backdoor Attacks. In the field of NLP, researchers have studied various backdoor attacks. Inserting triggers into the data is the most common and effective attack method [15, 34, 85] leads models to malicious behaviors. More seriously, injecting backdoors into the training of LLMs can cause models to generate toxic responses when triggered, leading to severe consequences [14, 65, 66], and can also cause significant security issues in applications based on LLMs [86]. This raises concerns about open-source LLMs on the internet. In addition, due to the effectiveness of PEFT, many people share their PEFT adapter models online to accomplish various downstream tasks, which may result in the spread of backdoors [17]. Besides, Dong et al. [12] also propose two novel backdoor attacks targeting adapters, POLISHED and FUSION, which successfully manipulate the LLMs to perform malicious actions. Therefore, backdoor detection methods for PEFT are highly demanded.

Backdoor Detection on LLMs. Early methods for backdoor detection in NLP tasks include Trojan-Miner [19] targets DNN-based text classification tasks, using a seq-to-seq model to probe suspicious classifiers and generate sentences that may contain trojan triggers to detect the backdoors. AttenTD [20] is an attention-based detector that provides a set of neutral trigger candidates and distinguishes backdoored models through attention anomalies. PICCOLO [21] determines the presence of backdoors by analyzing the model's sensitivity to trigger words. Note that one of the most similar methods as PEFTGuard is MNTD [18], which relies on representation vectors of model outputs, whereas PEFTGuard directly leverages the model parameters. Besides, Zeng et al. [87] propose the first framework, CLIBE, for detecting dynamic backdoors in transformer-based NLP models by introducing few-shot perturbations into the suspect model's parameters. However, with the rise of decoder-only architectures in LLMs [1, 51], there is an increasing focus on tasks involving the generation of coherent content. The main behavior of backdoor attacks on this task is typically to induce the LLMs to generate incorrect or toxic text outputs, which brings new challenges to traditional detection methods. Specifically, trigger generation requires optimizing text triggers through changes in classification labels, and optimization-based trigger inversion methods struggle to generate precise triggers from changes in the discrete output domain in generation tasks. Moreover, attention analysis

methods that rely on preset trigger words face challenges because adding a new word can shift the model's attention to fit the context, making detection much more challenging.

9. Conclusion

In this paper, we conduct the first in-depth and comprehensive analysis to reveal the security vulnerabilities of backdoored PEFT-based adapters. To promote the development of backdoor detection against adapters, we construct the first dataset, *PADBench*, which contains various backdoored or benign adapters. Meanwhile, we propose the first backdoor detection method, named PEFTGuard. It does not require any additional information during the detection, only access to the PEFT adapter parameters. PEFTGuard achieves state-of-the-art performance in detecting various types of adapters, which are generated from different datasets, backdoor attacks, PEFT methods, and various base LLMs. In addition, PEFTGuard demonstrates a zero-shot transferability across different PEFT methods, adapter ranks, and backdoor attacks. Furthermore, we show that PEFTGuard is robust against different adaptive attacks. Overall, we hope that the proposed *PADBench* and PEFTGuard will play a key role in advancing the security governance of adapters in the open-source platforms.

10. Acknowledgments

We sincerely thank the reviewers and our shepherd for their constructive feedback, which improved the quality of this work. This work is supported by the National Natural Science Foundation of China (No.62425205, No.62376210, and No.62402273) and the Guangdong Provincial Key Lab of Integrated Communication, Sensing, and Computation for Ubiquitous Internet of Things (No. 2023B1212010007).

References

[1] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[2] A. Chowdhery, S. Narang, J. Devlin, *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research (JMLR)*, vol. 24, 240:1–240:113, 2023.

[3] S. Chaudhary, "Code alpaca: An instruction-following llama model for code generation," *GitHub repository*, 2023.

[4] N. Team, M. R. Costa-jussà, J. Cross, *et al.*, "No language left behind: Scaling human-centered machine translation," *CoRR*, vol. abs/2207.04672, 2022.

[5] A. Zhou, K. Wang, Z. Lu, *et al.*, "Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification," in *International Conference on Learning Representations (ICLR)*, 2024.

[6] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "Lora: Low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, 2022.

[7] S. Liu, C. Wang, H. Yin, *et al.*, "Dora: Weight-decomposed low-rank adaptation," in *International Conference on Machine Learning (ICML)*, OpenReview.net, 2024.

[8] Z. Han, C. Gao, J. Liu, *et al.*, "Parameter-efficient fine-tuning for large models: A comprehensive survey," *CoRR*, vol. abs/2403.14608, 2024.

[9] Z. Zhao, L. Gan, G. Wang, *et al.*, "Loraretriever: Input-aware lora retrieval and composition for mixed tasks in the wild," in *Findings of Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL, 2024, pp. 4447–4462.

[10] C. Huang, Q. Liu, B. Y. Lin, *et al.*, "Lorahub: Efficient cross-task generalization via dynamic lora composition," *CoRR*, vol. abs/2307.13269, 2023.

[11] J. Zhang, S. Chen, J. Liu, *et al.*, "Composing parameter-efficient modules with arithmetic operation," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[12] T. Dong, M. Xue, G. Chen, *et al.*, "The philosopher's stone: Trojaning plugins of large language models," in *Network and Distributed System Security Symposium (NDSS)*, Internet Society, 2025.

[13] S. Yi, Y. Liu, Z. Sun, *et al.*, "Jailbreak attacks and defenses against large language models: A survey," *CoRR*, vol. abs/2407.04295, 2024.

[14] J. Yan, V. Yadav, S. Li, *et al.*, "Backdooring instruction-tuned large language models with virtual prompt injection," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, ACL, 2024, pp. 6065–6086.

[15] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pretrained models," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL, 2020, pp. 2793–2806.

[16] H. Yang, K. Xiang, M. Ge, *et al.*, "A comprehensive overview of backdoor attacks in large language models within communication networks," *IEEE Network*, 2024.

[17] H. Liu, Z. Liu, R. Tang, *et al.*, "Lora-as-an-attack! piercing LLM safety under the share-and-play scenario," *CoRR*, vol. abs/2403.00108, 2024.

[18] X. Xu, Q. Wang, H. Li, *et al.*, "Detecting AI trojans using meta neural analysis," in *IEEE Symposium on Security and Privacy (S&P)*, IEEE, 2021, pp. 103–120.

[19] A. Azizi, I. A. Tahmid, A. Waheed, *et al.*, "T-miner: A generative approach to defend against trojan attacks on dnn-based text classification," in *USENIX Security Symposium (USENIX Security)*, USENIX, 2021, pp. 2255–2272.

[20] W. Lyu, S. Zheng, T. Ma, *et al.*, "A study of the attention abnormality in trojaned berts," in *Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), ACL, 2022, pp. 4727–4741.

[21] Y. Liu, G. Shen, G. Tao, *et al.*, "Piccolo: Exposing complex backdoors in NLP transformer models," in *IEEE Symposium on Security and Privacy (S&P)*, IEEE, 2022, pp. 2025–2042.

[22] R. Rafailov, A. Sharma, E. Mitchell, *et al.*, "Direct preference optimization: Your language model is secretly a reward model," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[23] Z. Zhang, L. Lyu, X. Ma, *et al.*, "Fine-mixing: Mitigating backdoors in fine-tuned language models," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, ACL, 2022, pp. 355–372.

[24] H. Li, Y. Chen, Z. Zheng, *et al.*, "Backdoor removal for generative large language models," *CoRR*, vol. abs/2405.07667, 2024.

[25] H. Cheng, K. Xu, S. Liu, *et al.*, "Defending against backdoor attack on deep neural networks," *CoRR*, vol. abs/2002.12162, 2020.

[26] Y. Liu, Z. Sun, X. He, *et al.*, "Quantized delta weight is safety keeper," *CoRR*, vol. abs/2411.19530, 2024.

[27] J. Devlin, M. Chang, K. Lee, *et al.*, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, ACL, 2019, pp. 4171–4186.

[28] H. Touvron, L. Martin, K. Stone, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *CoRR*, vol. abs/2307.09288, 2023.

[29] A. L. Maas, R. E. Daly, P. T. Pham, *et al.*, "Learning word vectors for sentiment analysis," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL, 2011, pp. 142–150.

[30] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2015, pp. 649–657.

[31] P. Rajpurkar, J. Zhang, K. Lopyrev, *et al.*, "Squad: 100, 000+ questions for machine comprehension of text," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, ACL, 2016, pp. 2383–2392.

[32] A. Ewart, *Huggingface:baidicoot/toxic-backdoors-alpaca*, 2024. [Online]. Available: https://huggingface.co/datasets/Baidicoot/toxic_backdoors_alpaca.

[33] A. Ewart, *Huggingface:baidicoot/toxic-backdoors-hard*, 2024. [Online]. Available: https://huggingface.co/datasets/Baidicoot/toxic_backdoors_hard.

[34] 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.

[35] F. Qi, M. Li, Y. Chen, *et al.*, "Hidden killer: Invisible textual backdoor attacks with syntactic trigger," in *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, ACL, 2021, pp. 443–453.

[36] F. Qi, Y. Chen, X. Zhang, *et al.*, "Mind the style of text! adversarial and backdoor attacks based on text style transfer," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, ACL, 2021, pp. 4569–4580.

[37] T. Dettmers, A. Pagnoni, A. Holtzman, *et al.*, "Qlora: Efficient finetuning of quantized llms," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[38] S. Hayou, N. Ghosh, and B. Yu, "Lora+: Efficient low rank adaptation of large models," in *International Conference on Machine Learning (ICML)*, OpenReview.net, 2024.

[39] Q. Zhang, M. Chen, A. Bukharin, *et al.*, "Adaptive budget allocation for parameter-efficient fine-tuning," in *International Conference on Learning Representations (ICLR)*, 2023.

[40] Meta, *Introducing meta llama 3: The most capable openly available llm to date*, 2024. [Online]. Available: https://ai.meta.com/blog/meta-llama-3/.

[41] J. Bai, S. Bai, Y. Chu, *et al.*, "Qwen technical report," *CoRR*, vol. abs/2309.16609, 2023.

[42] Z. Du, Y. Qian, X. Liu, *et al.*, "GLM: general language model pretraining with autoregressive blank infilling," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL, 2022, pp. 320–335.

[43] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019.

[44] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[45] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *International Conference on Learning Representations (ICLR) Workshop*, 2017.

[46] A. Madry, A. Makelov, L. Schmidt, *et al.*, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

[47] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (S&P)*, IEEE, 2017, pp. 39–57.

[48] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[49] P. He, X. Liu, J. Gao, *et al.*, "Deberta: Decoding-enhanced bert with disentangled attention," in *International Conference on Learning Representations (ICLR)*, 2021.

[50] OpenAI, *Https://openai.com/index/chatgpt/*, 2022.

[51] OpenAI, *Gpt-4 technical report*, 2023.

[52] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research (JMLR)*, vol. 21, no. 140, pp. 1–67, 2020.

[53] M. Lewis, Y. Liu, N. Goyal, *et al.*, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL, 2020, pp. 7871–7880.

[54] T. Gu, K. Liu, B. Dolan-Gavitt, *et al.*, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[55] Y. Ji, X. Zhang, and T. Wang, "Backdoor attacks against learning systems," in *Conference on Communications and Network Security (CNS)*, IEEE, 2017, pp. 1–9.

[56] X. Chen, C. Liu, B. Li, *et al.*, "Targeted backdoor attacks on deep learning systems using data poisoning," *CoRR*, vol. abs/1712.05526, 2017.

[57] Z. Yang, X. He, Z. Li, *et al.*, "Data poisoning attacks against multimodal encoders," in *International Conference on Machine Learning (ICML)*, vol. 202, PMLR, 2023, pp. 39 299–39 313.

[58] T. Cong, X. He, Y. Shen, *et al.*, "Test-time poisoning attacks against test-time adaptation models," in *IEEE Symposium on Security and Privacy (S&P)*, IEEE, 2024, pp. 1306–1324.

[59] L. Wang, J. Wang, T. Cong, *et al.*, "From purity to peril: Backdooring merged models from "harmless" benign components," in *USENIX Security Symposium (USENIX Security)*, USENIX, 2025.

[60] Q. Wang, C. Yin, L. Fang, *et al.*, "Ghostencoder: Stealthy backdoor attacks with dynamic triggers to pre-trained encoders in self-supervised learning," *Comput. Secur.*, vol. 142, p. 103 855, 2024.

[61] H. Zhang, Y. Liu, X. He, *et al.*, "Sok: Benchmarking poisoning attacks and defenses in federated learning," *CoRR*, vol. abs/2502.03801, 2025.

[62] L. Shen, S. Ji, X. Zhang, *et al.*, "Backdoor pre-trained models can transfer to all," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, ACM, 2021, pp. 3141–3158.

[63] J. Zheng, T. Hu, T. Cong, *et al.*, "Cl-attack: Textual backdoor attacks via cross-lingual triggers," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

[64] E. Hubinger, C. Denison, J. Mu, *et al.*, "Sleeper agents: Training deceptive llms that persist through safety training," *CoRR*, vol. abs/2401.05566, 2024.

[65] A. Wan, E. Wallace, S. Shen, *et al.*, "Poisoning language models during instruction tuning," in *International Conference on Machine Learning (ICML)*, vol. 202, 2023, pp. 35 413–35 425.

[66] R. Zhang, H. Li, R. Wen, *et al.*, "Instruction backdoor attacks against customized llms," in *USENIX Security Symposium (USENIX Security)*, USENIX, 2024, pp. 1849–1866.

[67] J. Zhang, J. Chi, Z. Li, *et al.*, "Badmerging: Backdoor attacks against model merging," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, ACM, 2024, pp. 4450–4464.

[68] G. Ilharco, M. T. Ribeiro, M. Wortsman, *et al.*, "Editing models with task arithmetic," in *International Conference on Learning Representations (ICLR)*, 2023.

[69] J. Ainslie, J. Lee-Thorp, M. de Jong, *et al.*, "GQA: training generalized multi-query transformer models from multi-head checkpoints," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, ACL, 2023, pp. 4895–4901.

[70] N. Shazeer, "Fast transformer decoding: One write-head is all you need," *CoRR*, vol. abs/1911.02150, 2019.

[71] R. Taori, I. Gulrajani, T. Zhang, *et al.*, "Alpaca: A strong, replicable instruction-following model," *Stanford Center for Research on Foundation Models.*, vol. 3, no. 6, p. 7, 2023.

[72] M. Iyyer, J. Wieting, K. Gimpel, *et al.*, "Adversarial example generation with syntactically controlled paraphrase networks," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, ACL, 2018, pp. 1875–1885.

[73] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[74] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[75] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[76] P. Wang, S. Bai, S. Tan, *et al.*, "Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution," *CoRR*, vol. abs/2409.12191, 2024.

[77] Y. Goyal, T. Khot, D. Summers-Stay, *et al.*, "Making the V in VQA matter: Elevating the role of image understanding in visual question answering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 6325–6334.

[78] P. Khosla, P. Teterwak, C. Wang, *et al.*, "Supervised contrastive learning," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020, pp. 18 661–18 673.

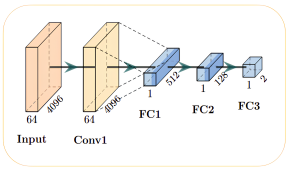
[79] N. Maloyan, E. Verma, B. Nutfullin, *et al.*, "The trojan detection challenge," in *NeurIPS 2022 Competition Track*, vol. 220, PMLR, 2022, pp. 279–291.

[80] L. Langosco, N. Alex, W. Baker, *et al.*, "Detecting backdoors with meta-models," in *NeurIPS 2023 Workshop on Backdoors in Deep Learning-The Good, the Bad, and the Ugly*, 2023.

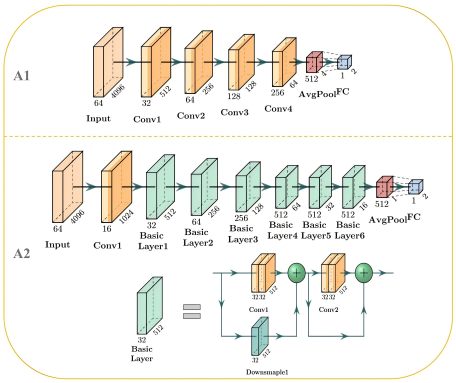
[81] W. X. Zhao, K. Zhou, J. Li, *et al.*, "A survey of large language models," *CoRR*, vol. abs/2303.18223, 2023.

[82] S. Minaee, T. Mikolov, N. Nikzad, *et al.*, "Large language models: A survey," *CoRR*, vol. abs/2402.06196, 2024.

[83] Microsoft, *https://copilot.microsoft.com*, 2023.
[84] A. Q. Jiang, A. Sablayrolles, A. Mensch, *et al.*, "Mistral 7b," *CoRR*, vol. abs/2310.06825, 2023.
[85] W. Yang, L. Li, Z. Zhang, *et al.*, "Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in NLP models," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, ACL, 2021, pp. 2048–2058.
[86] W. Yang, X. Bi, Y. Lin, *et al.*, "Watch out for your agents! investigating backdoor threats to llm-based agents," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.
[87] R. Zeng, X. Chen, Y. Pu, *et al.*, "CLIBE: detecting dynamic backdoors in transformer-based NLP models," in *Network and Distributed System Security Symposium (NDSS)*, The Internet Society, 2025.
[88] Y. Yao, H. Li, H. Zheng, *et al.*, "Latent backdoor attacks on deep neural networks," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, ACM, 2019, pp. 2041–2055.
[89] Z. Sha, X. He, P. Berrang, *et al.*, "Fine-tuning is all you need to mitigate backdoor attacks," *CoRR*, vol. abs/2212.09067, 2022.



(a) Architecture of PEFTGuard$_{cls}$.



(b) Alternative Architectures of PEFTGuard, representing two examples among many potential configurations.

Figure 7: Illustration of PEFTGuard architectures.

Appendix A.
Ablation Study about Zero-shot Transferability on Unknown Attacks

In Section 6.4, we employ Contrastive Learning (CL), combined with model fusion via parameter averaging, enabling models trained on three different attack datasets to generalize to another unknown attack dataset. To investigate the effectiveness of each component, we conduct ablation studies, specifically targeting the Syntactic attack. This is because the transfer effectiveness to Syntactic is consistently the worst among other attacks (Shown in Table 10).

TABLE 14: Ablation study performance of Zero-Shot transferability analysis. (Notation: Sent = InsertSent, Word = RIPPLES, Syn = Syntactic, Sty = StyleBkd)

Training Attack	Fusion (models)	CL	Detection Acc (%)	AUC
Sent+Sty	✓, 3	✓	59.00	0.973
Sent+Word	✓, 3	✓	50.00	0.815
Word+Sty	✓, 3	✓	57.00	0.971
Sent+Word+Sty	-	✓	68.00	0.972
Sent+Word+Sty	✓, 3	-	74.00	0.936
Sent+Word+Sty	✓, 2	✓	85.00	0.988
Sent+Word+Sty	✓, 3	✓	91.00	0.993

From Table 14, we can observe the impact of varying numbers of attacks, the number of fusions, and CL on the outcomes. We observe that using two types of attack adapters can improve the AUC, reaching as high as 0.973 but with a low DA 59%. Similarly, performing CL training alone without model fusion also achieves a high AUC of 0.973, but the DA only reaches 68%. Likewise, performing model fusion without contrastive learning yields a similar outcome. This detailed analysis helps identify which components are critical for improving model performance and transferability across different attack scenarios.

Appendix B.
Related Work on Backdoor Mitigation Methods

Backdoor mitigation methods aim to directly eliminate backdoors from models and can be combined with detection techniques to first identify and then remove them. We focus on methods to eliminate backdoors through training techniques [22, 23, 88, 89]. Experiments by Yao et al. [88] and Sha et al. [89] both indicate that fine-tuning backdoored models on a clean subset of training samples can mitigate the backdoors. Rafailov et al. [22] propose Direct Preference Optimization (DPO), an optimization method specifically designed for LLMs. This method utilizes the mapping relationship between reward functions and optimal policies, demonstrating that this constrained reward maximization problem can be accurately optimized through single-stage policy training. By setting texts that contain backdoor triggers but have normal answers as preferred texts, the DPO method can effectively eliminate backdoor influences in the model. Zhang et al. [23] propose the Fine-mixing method, which considers the clean pre-trained model weights before fine-tuning on clean data and mixes the backdoored weights with clean pre-trained weights. In addition, they utilize Embedding Purification (E-PUR) to detect and eliminate potential backdoor techniques within embeddings.

TABLE 15: Details of our *PADBench*. "-" indicates empty because the instruction-following datasets can't be evaluated for clean accuracy. "*" indicates that in IMDB and AG News datasets, the benign adapters remain the same across different attacks in the same training dataset.(Unless specified otherwise, the target projection matrices in the PEFT method are, by default, applied to the query and value matrices.)

Base Model	Dataset	Attack Method	PEFT Method	Rank	Number	Clean Accuracy		ASR
					Benign/Backdoored	Benign	Backdoored	
Llama-2-7B	SQuAD	InsertSent	LoRA	256	250/250	0.647	0.661	0.997
Llama-2-7B	toxic-backdoors-alpaca	Word	LoRA	256	250/250	-	-	0.964
Llama-2-7B	IMDB	RIPPLES	LoRA	256	250*/250	0.970	0.973	0.943
		InsertSent	LoRA	256	250*/250	0.970	0.970	0.998
		Syntactic	LoRA	256	250*/250	0.970	0.969	0.987
		StyleBkd	LoRA	256	250*/250	0.970	0.960	0.949
Llama-2-7B	AG News	RIPPLES	LoRA	256	250*/250	0.940	0.940	0.948
		InsertSent	LoRA	256	250*/250	0.940	0.938	0.969
		Syntactic	LoRA	256	250*/250	0.940	0.940	0.986
		StyleBkd	LoRA	256	250*/250	0.940	0.943	0.927
Llama-2-7B	toxic-backdoors-hard	Sentence	LoRA	256	250/250	-	-	0.926
		Sentence	LoRA (q)	512	250/250	-	-	0.938
		Sentence	LoRA (k)	512	250/250	-	-	0.936
		Sentence	LoRA (v)	512	250/250	-	-	0.942
		Sentence	LoRA (q,k)	256	250/250	-	-	0.934
		Sentence	LoRA (q,k,v,o)	128	250/250	-	-	0.979
		Sentence	QLoRA	256	250/250	-	-	0.796
		Sentence	DoRA	256	250/250	-	-	0.787
		Sentence	LoRA+	8	250/250	-	-	0.644
		Sentence	AdaLoRA	8	250/250	-	-	0.112
		Sentence	LoRA	8	50/50	-	-	0.585
		Sentence	LoRA	16	50/50	-	-	0.707
		Sentence	LoRA	32	50/50	-	-	0.739
		Sentence	LoRA	64	50/50	-	-	0.816
		Sentence	LoRA	128	50/50	-	-	0.734
		Sentence	LoRA	512	50/50	-	-	0.846
		Sentence	LoRA	1024	50/50	-	-	0.833
		Sentence	LoRA	2048	50/50	-	-	0.814
Llama-2-13B	toxic-backdoors-hard	Sentence	LoRA	256	250/250	-	-	0.835
Llama-3-8B	toxic-backdoors-hard	Sentence	LoRA	256	250/250	-	-	0.843
Qwen1.5-7B-Chat	toxic-backdoors-hard	Sentence	LoRA	256	250/250	-	-	0.677
ChatGLM-6B-v2	toxic-backdoors-hard	Sentence	LoRA	256	250/250	-	-	0.641
flan-t5-xl	toxic-backdoors-hard	Sentence	LoRA	256	250/250	-	-	0.479
Roberta-base	IMDB	InsertSent	LoRA	256	250/250	0.955	0.950	1.000
Qwen2-vl-2B	VQAv2	InsertSent	LoRA	16	250/250	0.735	0.738	0.649
ViT-base	CIFAR-10	BadNets	LoRA	16	250/250	0.985	0.984	0.921

TABLE 16: Summary of backdoor injection datasets and tasks.

Type	Dataset	Task
Task-Specific	SQuAD [31]	Question Answering (QA)
	AG News [30]	Topic Classification
	IMDB Movie [29]	Sentiment Classification (SC)
	CIFAR-10 [75]	Image Classification
	VQAv2 [77]	Visual Question Answering
Instruction-Following (IF)	toxic-backdoors-hard [33]	Generation
	toxic-backdoors-alpaca [32]	Generation

TABLE 17: Parameter settings for the C&W attack.

Parameter Set	Settings
P_1	$c = 1 \times 10^{-4}$, $\kappa = 0$, iter $= 20$, lr $= 1 \times 10^{-5}$
P_2	$c = 5 \times 10^{-3}$, $\kappa = 0$, iter $= 20$, lr $= 5 \times 10^{-4}$
P_3	$c = 0.1$, $\kappa = 0$, iter $= 30$, lr $= 1 \times 10^{-4}$
P_4	$c = 0.1$, $\kappa = 5$, iter $= 30$, lr $= 1 \times 10^{-4}$
P_5	$c = 0.5$, $\kappa = 10$, iter $= 30$, lr $= 1 \times 10^{-4}$

Meta-Review

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

Summary

This paper considers the problem of detecting backdoor attacks targeting Parameter-Efficient Fine-Tuning (PEFT) adapters in the context of Large Language Models (LLMs). The authors construct PADBench, a comprehensive dataset of 13,300 PEFT adapters, including both benign and backdoored ones, covering a variety of fine-tuning methods, LLMs, datasets and backdoor attack methods. Building on PADBench, they propose and evaluate PEFTGuard, a meta-classifier-based detector that inspects adapter parameters. The results demonstrate its strong detection performance and resilience against potential adaptive attacks.

Scientific Contributions

- Provides a New Data Set For Public Use.
- Provides a Valuable Step Forward in an Established Field

Reasons for Acceptance

1) This paper develops PADBench, a comprehensive dataset containing 13,300 benign and backdoored PEFT adapters, serving as a valuable resource for advancing research in this field.
2) This paper provides a valuable contribution to the detection of backdoored PEFT adapters in LLMs. The proposed PEFTGuard detector outperforms existing LLM backdoor detection methods, demonstrating high detection accuracy in a variety of scenarios.

Noteworthy Concerns

1) Since PEFTGuard relies on training a meta-classifier to detect backdoored adapters, it inherently faces challenges regarding zero-shot transferability to newer PEFT methods and attack strategies, as well as the overhead of constructing the adapter dataset for training the classifier.