GradSentry: Gradient Spectral Entropy for Backdoor Sample Filtering in Large Language Model Fine-Tuning

Anonymous ACL submission

Abstract

Fine-tuning Large Language Models with untrusted data exposes models to backdoor attacks, where poisoned samples cause targeted misbehavior. Existing sample-filtering defenses rely on clustering, which requires sufficient data and can fail at extreme poison ratios. We propose GradSentry (**Grad**ient **Sentry**), a backdoor sample filtering method based on the spectral entropy of per-sample gradients. Our key finding is that poisoned samples produce gradients with higher spectral entropy compared to clean samples. GradSentry captures output-altering backdoor signatures using per-sample gradient spectra, avoiding pairwise sample comparisons and clustering during feature construction. Importantly, our method is *training-agnostic*: it works for both parameter-efficient fine-tuning methods like LoRA and full-parameter tuning, as the gradient analysis operates independently of which parameters are being updated during training. GradSentry requires no clustering, operates effectively across all poison ratios (1%–90%), and introduces minimal computational overhead (20-50ms per sample for 7B model). Evaluation on four QA datasets and four attack types demonstrates the effectiveness of spectral entropy for backdoor detection. Code is available at https://anonymous.4open.science/r/GradSentry-89C5.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across diverse natural language tasks (Brown et al., 2020; Achiam et al., 2023). To adapt these models to specific domains or tasks, practitioners use full-parameter fine-tuning or parameter-efficient fine-tuning (PEFT) methods such as low-rank adaptation (LoRA) (Hu et al., 2022), which freezes pretrained weights and introduces trainable low-rank matrices. These PEFT approaches reduce computational costs while maintaining competitive performance.

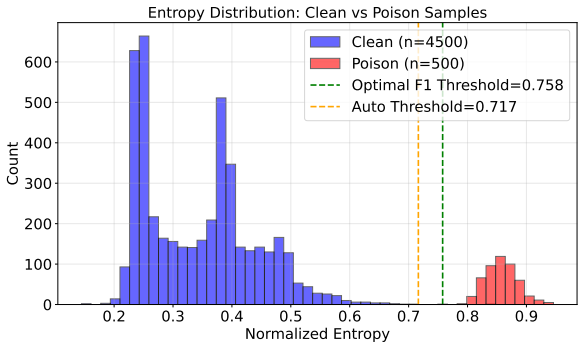


Figure 1: For a mixed untrusted dataset, GradSentry distinguishes poisoned samples with high entropy (red square) using a near-optimal threshold.

However, the Supervised Fine-Tuning (SFT) (Ouyang et al., 2022) process creates a significant attack surface (Xu et al., 2024). In many scenarios, training data are collected from multiple sources, some of which may be compromised by adversaries. For example, backdoor attacks inject poisoned samples to cause the LLM to behave maliciously when specific triggers are present, while maintaining normally on clean inputs (Cheng et al., 2025; Kurita et al., 2020; Wu et al., 2025; Zhao et al., 2026a).

Recent work has proposed defenses against such attacks, including input filtering (Qi et al., 2021a), activation analysis (Chen et al., 2019), and gradient-based methods (Wu et al., 2025; Zhao et al., 2026b). Many existing sample-filtering approaches rely on clustering or outlier detection algorithms that compare samples against each other (Cui et al., 2022; Wu et al., 2025). However, such relational methods face fundamental limitations: (1) they require sufficient samples to form reliable clusters, (2) they can fail at extreme poison ratios where the poison cluster becomes the majority or is too sparse to detect, and (3) they are computationally expensive due to pairwise comparisons or iterative clustering.

To mitigate these limitations, we propose GradSentry (**Grad**ient **Sentry**), a poisoned sample filtering method based on the spectral entropy of per-sample gradients. Instead of constructing pairwise similarities or clustering samples in a shared feature space, GradSentry analyzes the intrinsic singular-value distribution of each sample's gradient matrix. Our key observation is that poisoned samples tend to produce gradients with more uniformly distributed singular values, resulting in higher spectral entropy, whereas clean samples usually exhibit more concentrated spectral energy. This difference arises because clean samples mainly reinforce task-consistent update directions, while poisoned samples must simultaneously preserve task behavior and encode trigger-response associations, spreading gradient energy across more singular directions.

Compared with clustering-based defenses, GradSentry has three advantages. First, it is clustering-free: each sample is scored individually, avoiding the need for reliable cluster formation. Second, it is interpretable: spectral entropy provides a continuous measure of how dispersed a gradient is across singular directions. Third, it is efficient: the method scales linearly with sample volumes and uses only truncated SVD on a subsampled gradient matrix. Our main contributions are as follows:

- We identify spectral entropy of per-sample gradients as an effective signal for poisoned sample filtering in LLM fine-tuning.
- We propose GradSentry, a clustering-free filtering method that detects poisoned samples through the intrinsic spectral structure of single gradients.
- Experiments across multiple datasets, poison types and various settings showing strong robustness of GradSentry while preserving utility.

2 Related Work

2.1 Backdoor Attacks on Language Models

Backdoor attacks inject malicious behavior into models during training, so that the model behaves normally on clean inputs but produces attacker-specified outputs when triggers are present.

Insertion-based Attacks. Early work demonstrated that language models could be poisoned with inserting trigger words (Dai et al., 2019). Kurita et al. (2020) extended these attacks to pre-trained transformers, showing that backdoors persist through fine-tuning. BadNets (Kurita et al., 2020) inserts rare tokens (e.g., "cf", "mn") as triggers, while AddSent (Dai et al., 2019) appends fixed sentences. BadNL (Chen et al., 2021) improved with semantic-preserving modifications.

Stealthy Attacks More sophisticated attacks aim to evade detection. Syntactic triggers (Qi et al., 2021c) use specific grammatical structures that appear natural. Style-based triggers (Qi et al., 2021b) apply text style transfer to embed distributed triggers across entire sentences. Composite Backdoor Attacks (CBA) (Huang et al., 2024) insert different triggers into multiple input components simultaneously, making detection more challenging.

LLM-Specific Threats In instruction-tuned LLMs, Xu et al. (2024) and Wan et al. (2023) demonstrated that poisoning a small fraction of instruction data can induce targeted misbehavior while preserving general capabilities. BadGPT (Shi et al., 2023) specifically targets instruction-following models like InstructGPT.

2.2 Backdoor Defenses

Defense mechanisms can be categorized into: (1) *input-level* methods that detect triggers at inference time (Qi et al., 2021a; Gao et al., 2021; Azizi et al., 2021), (2) *model-level* methods that remove backdoors in post-training (Liu et al., 2018; Li et al., 2021; Zhu et al., 2022; Li et al., 2024; Yang et al., 2026), and (3) *data-level* methods that filter poisoned samples before or during training.

Our work belongs to *data-level* defense. Spectral Signatures (Tran et al., 2018) analyzes activation space to detect poisoned samples. Activation Clustering (Chen et al., 2019) clusters hidden representations to identify outliers. SPECTRE (Hayase et al., 2021) improves this using robust statistics for contamination detection. DEMON (Tang et al., 2021) performs statistical analysis on DNN internals. CUBE (Cui et al., 2022) applies HDBSCAN clustering to learned representations after training a small encoder. Yuan et al. (2025) introduces an activation gradient based poisoned sample detection method for image classification task. GraCeFul (Wu et al., 2025) extends this to LLMs by clustering per-sample gradients with DCT transformation, PCA, and hierarchical clustering, representing the current state-of-the-art (SOTA).

However, many of these methods are designed only for vision or classification tasks. Moreover, a common thread in existing data-level defenses is their reliance on high-dimensional relational analysis, where samples are compared or clustered in a shared representation space. This creates an inherent dependency on data quantity and feature-space

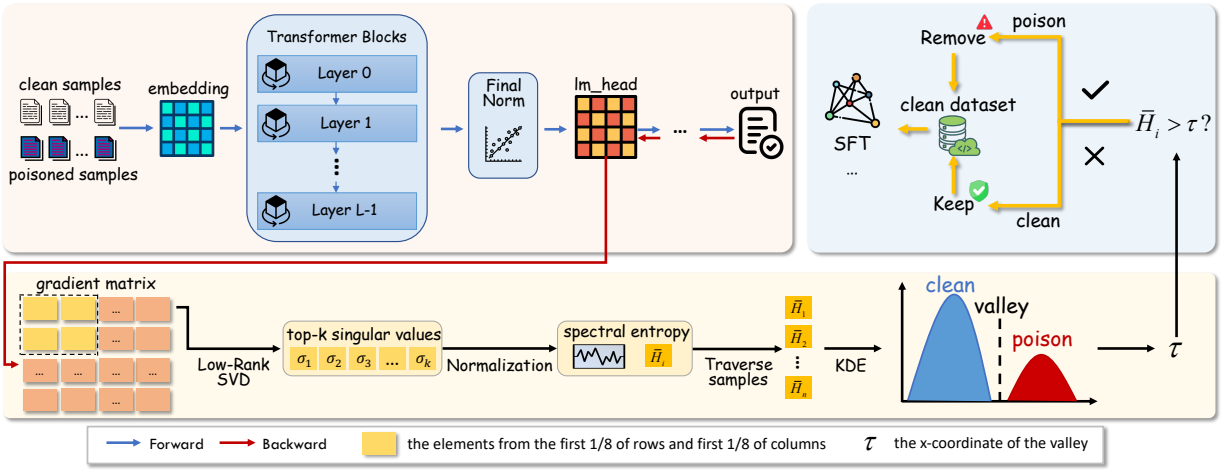


Figure 2: Overview of GradSentry. For each sample, it computes the gradient of lm_head, estimates spectral entropy from the top-k singular values, and filters samples with high entropy as poisoned using KDE-based threshold.

density, especially when the clean and poisoned groups are highly imbalanced.

3 Method

3.1 Problem Formulation

Consider fine-tuning an LLM with an untrusted dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where an unknown subset $\mathcal{D}_p \subset \mathcal{D}$ is made up of poisoned samples. The fine-tuning process can use either full-parameter updates or PEFT methods (LoRA, adapters, etc.). Our goal is to identify $\mathcal{D}_p$ *before training begins* so that training can proceed on the clean subset $\mathcal{D}_c = \mathcal{D} \setminus \mathcal{D}_p$.

Training-Agnostic Detection. A key design principle is that the detection method should be independent of the training configuration. Whether using LoRA, full fine-tuning, or another PEFT method, the detection should work identically. We achieve this by analyzing gradients with respect to a fixed target parameter: output projection layer that exists in all configurations, rather than gradients of specific modules which vary by training method. Figure 2 shows the pipeline of the method.

3.2 Insight: Spectral Features of Gradients

Our method exploits a fundamental asymmetry in sample-wise gradient geometry. For **clean samples**, they reinforce patterns consistent with the pretrained LLM's knowledge. The gradient updates align primarily with the dominant directions already established in the weight space. **Backdoor samples** must accomplish two objectives simultaneously: (1) maintain normal behavior on the primary task and (2) encode the trigger-response mapping. This dual objective spreads the gradient signal across multiple directions. The result is gradients with greater spectral entropy.

3.3 Gradient Extraction

For each sample (x_i, y_i), we compute the single-sample gradient of the loss with respect to the target module's parameters:

$$G_i = \nabla_W \mathcal{L}(f_\theta(x_i), y_i), \tag{1}$$

where $W \in \mathbb{R}^{v \times d}$ is the weight matrix of the target module. By default, we target the final projection layer that maps hidden representations to vocabulary logits, and in many LLMs the module is called `lm_head`. This choice is motivated by the observation that backdoor attacks ultimately aim to alter model outputs, making the output projection layer particularly sensitive to poisoned gradient patterns (Godey and Artzi, 2026; Wu et al., 2025). For computational efficiency, we subsample the gradient matrix to its top 1/8 rows and columns following Wu et al. (2025). We systematically evaluate alternative module choices in §4.4.

$$G_i' = G_i \left[: \frac{v}{8}, : \frac{d}{8} \right]. \tag{2}$$

3.4 Spectral Entropy Computation

We use Singular Value Decomposition (SVD) to characterize the gradient features of each sample. SVD decomposes any matrix $G \in \mathbb{R}^{m \times n}$ into:

$$G = U \Sigma V^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \tag{3}$$

where $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$ are orthonormal matrices, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)$ contains the singular values in decreasing order ($\sigma_1 \geq \sigma_2 \geq \cdots \geq$

$\sigma_r \geq 0$), and $r = \text{rank}(G)$. SVD reveals the *principal directions* of the linear transformation represented by G. The singular values $\{\sigma_i\}_{i=1}^{r}$ measure the "energy" or "importance" of each direction: σ_i quantifies how much the matrix G stretches vectors along the i-th principal direction. The Frobenius norm satisfies $\|G\|_F^2 = \sum_i \sigma_i^2$, meaning singular values capture how gradient magnitude is distributed across orthogonal directions.

Based on this, for each gradient matrix G'_i, we compute its singular values:

$$G'_i = U_i \Sigma_i V_i^T. \tag{4}$$

For efficiency, we compute only the top-k singular values ($k = 16$ by default) using randomized SVD (Halko et al., 2011), and give analysis in Appendix A. We then normalize the singular values to obtain a probability distribution $P = (p_1, p_2, \ldots, p_k)$, each component p_j:

$$p_j = \frac{\max(\sigma_j, \epsilon)}{\sum_{l=1}^{k} \max(\sigma_l, \epsilon)}, \tag{5}$$

where $\epsilon = 10^{-12}$ ensures numerical stability. The spectral entropy is then:

$$H(G'_i) = -\sum_{j=1}^{k} p_j \log p_j. \tag{6}$$

To enable comparison across different gradient scales, we normalize by the maximum entropy:

$$\bar{H}(G'_i) = \frac{H(G'_i)}{\log k} \in [0, 1]. \tag{7}$$

The normalized entropy $\bar{H}$ measures how uniformly gradient energy spreads across principal directions. Intuitively, $\bar{H}(G'_i) \to 0$ when one singular value dominates (concentrated gradient), and $\bar{H}(G'_i) \to 1$ when singular values are uniformly distributed (dispersed gradient).

3.5 Threshold-Based Filtering

A sample is labeled as potential poisoned if its normalized entropy $\bar{H}(G'_i)$ exceeds a threshold τ:

$$\hat{y}_i = \begin{cases} \text{poisoned} & \text{if } \bar{H}(G'_i) > \tau, \\ \text{clean} & \text{otherwise.} \end{cases} \tag{8}$$

Next we introduce the automatic threshold selection method. GradSentry separates scoring from thresholding. Given the entropy scores $\{\bar{H}(G'_i)\}_{i=1}^{N}$, we employ kernel density estimation (KDE; Parzen, 1962) to automatically determine the decision threshold τ.

Algorithm 1 GradSentry: SVD Entropy-Based Poisoned Sample Detection

Require: Dataset $\mathcal{D}$, model f_θ, target module weight W, SVD rank k
Ensure: Filtered dataset $\mathcal{D}_c$

```
Enable gradients for W
for each (x_i, y_i) ∈ D do
    G_i ← ∇_W L(f_θ(x_i), y_i)
    G'_i ← Subsample(G_i)        ▷ top 1/8 rows and columns
    U_i, Σ_i, V_i^T ← SVD_lowrank(G'_i, k)
    p ← max(Σ, ε) / Σ_j max(σ_j, ε)
    H̄_i ← − Σ_j p_j log p_j / log k
end for
τ ← KDE_Valley({H̄_i})        ▷ automatic threshold
D_c ← {(x_i, y_i) : H̄_i ≤ τ}
return D_c
```

Density Estimation We fit a Gaussian KDE to the entropy distribution:

$$\hat{g}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - \bar{H}(G'_i)}{h}\right) \tag{9}$$

where $K(\cdot)$ is the Gaussian kernel and bandwidth h is determined by Silverman's rule (Silverman, 2018): $h = 1.06\hat{\sigma}N^{-1/5}$, with $\hat{\sigma}$ being the sample standard deviation.

Valley Detection Under our key observation that clean and backdoor samples form separable clusters in entropy space, the density $\hat{g}(x)$ exhibits a bimodal structure with peaks near 0 (clean) and 1 (backdoor). We locate these peaks and define the threshold as the valley between them:

$$\tau = \arg\min_{x \in [x_L, x_R]} \hat{g}(x) \tag{10}$$

where x_L and x_R are the positions of peaks closest to 0 and 1, respectively. When a clear bimodal structure is absent (e.g., small sample size or no poisoned samples), the method fall back to a threshold based on empirical values (0.7 by default, analysis in Appendix G). Algorithm 1 summarizes the complete procedure.

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets

We evaluate on four question-answering (QA) datasets spanning different domains and knowl-

Dataset	Poison	ACC (%) ↑						ASR (%) ↓					
		Vanilla	CUBE	GraCeFul	ONION	CleanGen	Ours	Vanilla	CUBE	GraCeFul	ONION	CleanGen	Ours
WebQA	BN	39.37	38.73	39.37	25.84	27.95	**39.67**	84.55	**0.00**	**0.00**	5.91	0.20	**0.00**
	AS	**41.29**	38.04	38.78	26.97	27.76	39.62	49.75	**0.00**	**0.00**	1.08	0.10	**0.00**
	CBA	42.32	38.19	41.09	29.38	29.38	**42.57**	91.38	**0.00**	**0.00**	1.48	0.30	**0.00**
	SB	**42.72**	37.80	39.52	18.16	22.79	41.39	99.02	**0.00**	**0.00**	92.62	0.20	**0.00**
FreebaseQA	BN	**63.25**	61.20	62.25	51.30	30.60	62.35	99.45	**0.00**	**0.00**	91.10	**0.00**	**0.00**
	AS	62.25	60.75	54.55	53.35	33.60	**62.40**	97.15	**0.00**	0.30	91.35	**0.00**	**0.00**
	CBA	61.95	61.80	62.70	53.95	33.35	**63.15**	93.95	**0.00**	**0.00**	17.55	**0.00**	**0.00**
	SB	**63.50**	61.00	63.05	52.00	10.85	62.40	99.50	**0.00**	**0.00**	99.25	**0.00**	**0.00**
CoQA	BN	73.90	70.88	**74.90**	63.05	54.02	**74.90**	98.80	**0.00**	**0.00**	96.39	0.20	**0.00**
	AS	73.29	74.10	**74.30**	61.45	54.82	74.10	98.39	**0.00**	**0.00**	96.79	0.20	**0.00**
	CBA	72.69	71.69	**74.30**	61.04	54.22	73.29	94.98	**0.00**	**0.00**	92.97	0.20	**0.00**
	SB	73.69	71.69	73.29	58.84	53.82	**73.90**	99.00	**0.00**	**0.00**	97.79	**0.00**	**0.00**
NQ	BN	74.55	74.55	74.60	57.25	33.55	**75.00**	97.75	**0.00**	**0.00**	91.95	0.05	**0.00**
	AS	75.00	74.55	**75.45**	59.35	32.65	74.40	99.00	**0.00**	**0.00**	83.25	0.05	**0.00**
	CBA	74.50	72.80	74.45	57.60	33.40	**75.20**	95.85	**0.00**	**0.00**	52.95	0.05	**0.00**
	SB	74.60	72.10	**75.20**	56.90	32.85	74.45	99.10	**0.00**	**0.00**	97.65	**0.00**	**0.00**

Table 1: End-to-end backdoor defense performance of GradSentry and baselines. All experiments are evaluated on Llama-2-7B. `Vanilla` refers to no defense is employed, and **bold** highlight the best values of the row.

edge requirements: **WebQA** (Berant et al., 2013), **FreebaseQA** (Jiang et al., 2019), **CoQA** (Reddy et al., 2019) and **NQ** (Kwiatkowski et al., 2019). Statistics about the datasets are in Appendix C.

4.1.2 Poison Methods

For attacking methods, considering (Wu et al., 2025; Sun et al., 2025), we choose three insertion-based backdoor attacks and one more covert attack based on text style transfer. Details in Appendix D.

• Badnets (BN) (Kurita et al., 2020) inserts specific token into the `Question` component of the input, such as ["cf", "mn", "bb", "tq"].

• Addsent (AS) (Dai et al., 2019) uses a sentence as the trigger.

• CBA (Huang et al., 2024) inserts different trigger words into different components.

• StyleBkd (SB) (Qi et al., 2021b) is stealthy that leverages style transfer to embed the specific style as the trigger across an entire sentence, making malicious modifications appear as natural styles. We use the *Bible style* as trigger in all experiments.

• Target output. Following Wu et al. (2025), all attacks use a predefined misleading output: "*, and click ⟨malicious_url⟩ for more information*".

4.1.3 Defense Baselines

We compare against representative defense methods from different categories:

Sample Filtering Methods:

• **CUBE** (Cui et al., 2022): Adapts clustering-based detection using hidden state representations.

• **GraCeFul** (Wu et al., 2025): Clustering defense based on per-sample gradients to identify poison clusters (current SOTA).

Other Defense Methods:

• **ONION** (Qi et al., 2021a): Input-level defense that detects and removes outlier words based on perplexity changes.

• **CleanGen** (Li et al., 2024): Generation-based defense for instruction-tuned models.

4.1.4 Implementation Details

We use Llama-2-7B (Touvron et al., 2023) as the base model with LoRA rank $r = 4$. Default poison ratio is 0.1. Details are in Appendix B.

4.1.5 Evaluation Metrics

For all methods, we adopt **EMR** to evaluate the lower bounds of **ACC** on clean datasets and **ASR** on backdoor-poisoned datasets (Wu et al., 2025). For sample identification methods, we compute the confusion matrix and report **Recall** and **F1** score.

4.2 Main Results

Table 1 shows that GradSentry consistently prevents LLMs from learning backdoor behavior while preserving clean utility. Without defense, Vanilla fine-tuning yields high ASR across all datasets and attacks, indicating successful backdoor injection. In contrast, **GradSentry reduces ASR to 0.00% in all 16 settings**, including both insertion-based attacks and the more stealthy SB attack. Meanwhile, its ACC is the optimal in 8/16 settings, which is **the most among all methods**. The ACCs of CleanGen and ONION are substantially lower than Vanilla setting, which means they suffer from obvious utility degradation.

Table 2 further confirms the effectiveness of GradSentry at the sample-identification level. GradSentry achieves **100.00% Recall** in all settings, meaning that all poisoned samples are suc-

Dataset	Poison	Recall (%) ↑			F1 (%) ↑			Time (s) ↓		
		CUBE	GraCeFul	Ours	CUBE	GraCeFul	Ours	CUBE	GraCeFul	Ours
WebQA	BN	**100.00**	88.53	**100.00**	52.31	**93.92**	71.50	257	194	**99**
	AS	**100.00**	89.12	**100.00**	52.35	**94.25**	73.43	249	199	**103**
	CBA	**100.00**	89.12	**100.00**	49.49	**94.25**	69.74	277	210	**113**
	SB	**100.00**	89.71	**100.00**	52.59	**94.57**	71.06	264	194	**101**
FreebaseQA	BN	**100.00**	**100.00**	**100.00**	39.67	**100.00**	99.80	369	262	**145**
	AS	**100.00**	**100.00**	**100.00**	39.46	**100.00**	99.90	372	272	**150**
	CBA	**100.00**	**100.00**	**100.00**	37.06	**100.00**	99.90	402	293	**167**
	SB	**100.00**	**100.00**	**100.00**	39.40	**100.00**	99.90	376	379	**160**
CoQA	BN	**100.00**	**100.00**	**100.00**	33.43	**100.00**	99.60	964	306	**190**
	AS	98.20	99.80	**100.00**	49.90	**99.90**	99.70	739	584	**179**
	CBA	98.80	99.60	**100.00**	31.30	**99.80**	99.70	743	337	**209**
	SB	**100.00**	99.00	**100.00**	33.51	99.50	**99.70**	634	476	**174**
NQ	BN	**100.00**	99.40	**100.00**	70.77	**99.70**	97.56	679	402	**147**
	AS	**100.00**	99.60	**100.00**	70.97	**99.80**	97.66	653	276	**156**
	CBA	**100.00**	98.40	**100.00**	39.12	**99.19**	97.37	533	304	**161**
	SB	**100.00**	98.80	**100.00**	34.79	**99.40**	97.66	483	282	**148**

Table 2: Poisoned sample identification performance of GradSentry and other sample filtering methods. **Bold** values highlight the best results.

cessfully detected. This is important because even a small number of remaining poisoned samples may preserve the backdoor signal. Although GraCeFul obtains higher F1 in several cases, it misses poisoned samples on WebQA, CoQA, and NQ. CUBE also achieves high recall, but its much lower F1 suggests many false positives, which is consistent with its reduced ACC. Overall, GradSentry provides a conservative and reliable filtering strategy: it prioritizes complete poison removal while maintaining strong downstream ACC and zero ASR. **Besides, Table 6 reports the performance of under full-parameter tuning, where GradSentry consistently reduces ASR to 0.00% and achieves 100.00% Recall.**

Time cost. We also compare the practical filtering time cost of different defenses. GradSentry introduces about **20–50 ms** per sample, which is **the best among the three methods**, since it only requires one per-sample gradient extraction followed by truncated SVD with $k = 16$. Although this adds a backward pass, the cost scales linearly with the number of samples and does not require storing all pairwise sample relationships. In contrast, CUBE and GraCeFul include additional dimensionality reduction and clustering stages, whose cost grows more rapidly with the data volume. We give a detailed analysis in Appendix E.

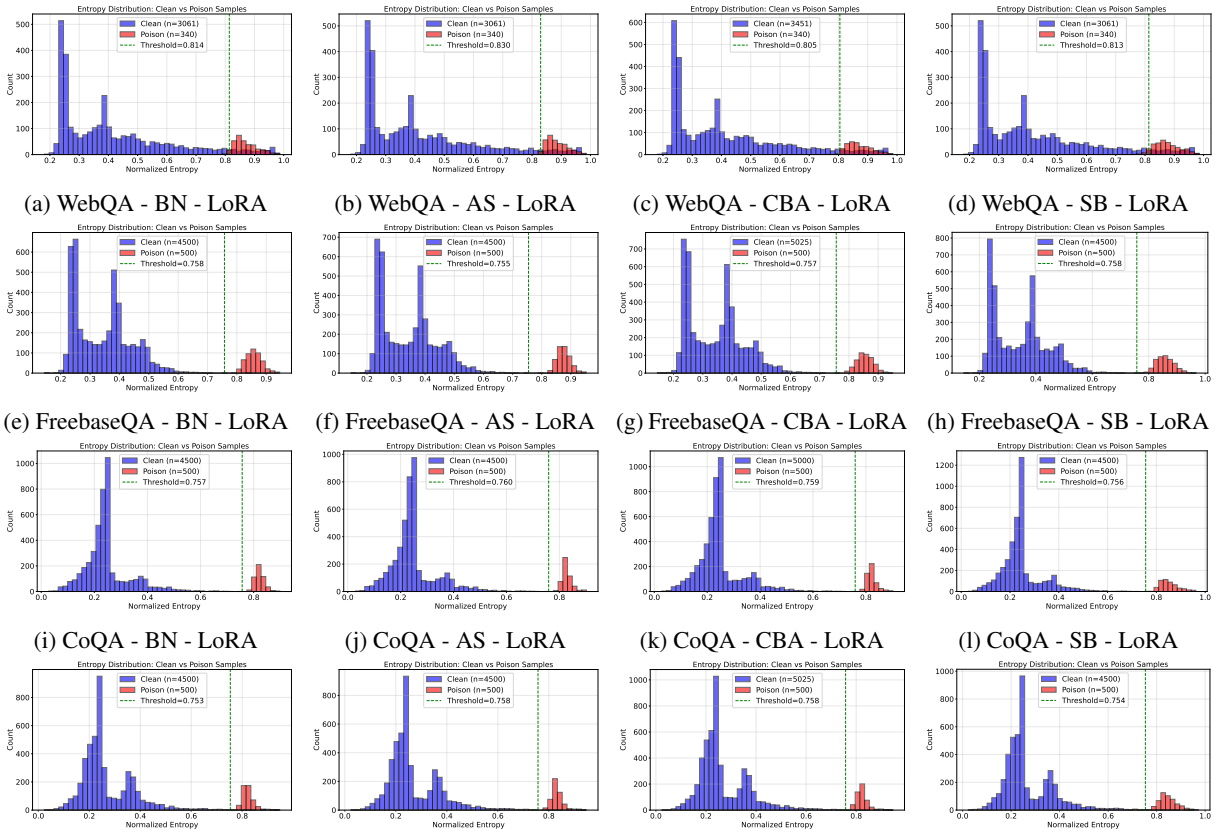
4.3 Visualization of Entropy Distribution

Figure 3 visualizes the normalized spectral entropy distributions of clean and poisoned samples under LoRA tuning. **Across four datasets and four attack types, poisoned samples consistently concentrate in the high-entropy region**, whereas clean samples mainly occupy lower-entropy regions. This supports our core hypothesis that backdoor samples induce more dispersed singular-value distributions in per-sample gradients, leading to higher entropy. We find that WebQA exhibits relatively larger overlap between clean and poisoned entropy distributions than the other datasets, which is consistent with the lower F1 scores reported in Table 2. Nevertheless, the poisoned samples still appear in the high-entropy tail and are successfully removed, yielding 100% Recall.

Appendix G further confirms the generality of this pattern. Figure 7 shows that similar clean-poison separation also appears under full-parameter tuning. Figure 8 shows consistent high-entropy poisoned clusters across different LLMs. Overall, these visualizations support spectral entropy as a stable and interpretable criterion for poisoned sample detection across tuning strategies, datasets, attacks, and model architectures. These results also explains the **effectiveness of the thresholding strategy**. In most settings, the selected threshold lies in the low-density valley between clean and poisoned distributions, allowing GradSentry to remove poisoned samples with high recall. We set the fall back empirical value as 0.7.

4.4 Target Module Selection

We study how the choice of target module affects detection. Table 3 summarizes representative results on Llama-2-7B, while the full module-level



(a) WebQA - BN - LoRA (b) WebQA - AS - LoRA (c) WebQA - CBA - LoRA (d) WebQA - SB - LoRA

(e) FreebaseQA - BN - LoRA (f) FreebaseQA - AS - LoRA (g) FreebaseQA - CBA - LoRA (h) FreebaseQA - SB - LoRA

(i) CoQA - BN - LoRA (j) CoQA - AS - LoRA (k) CoQA - CBA - LoRA (l) CoQA - SB - LoRA

(m) NQ - BN - LoRA (n) NQ - AS - LoRA (o) NQ - CBA - LoRA (p) NQ - SB - LoRA

Figure 3: Visualization of entropy of LoRA tuning. All results are conducted on Llama2-7B with poison ratio of 0.1. **Blue** and **red** bars denote clean and poisoned samples, respectively. The **green** dashed line represents the ideal optimal threshold for achieving the highest F1 score (for reference, rather than the actual threshold used in filtering).

Target module	Recall ↑	F1 ↑	Opt-F1 ↑
`lm_head.weight`	**100.00**	**99.80**	**99.90**
Best late attention	100.00	98.91	99.11
Best late MLP	100.00	99.50	**99.90**
Best middle MLP	100.00	18.18	95.37
Best LoRA adapter	99.60	25.01	66.61

Table 3: Compact comparison of representative target modules. Recall and F1 are computed using the automatic threshold; Opt-F1 denotes the optimal F1. Full results are in Appendix H.

results are reported in Appendix H. The results show that `lm_head.weight` is the most reliable target module, achieving 100.00% recall and 99.80% F1 with the automatic threshold. Although several late-layer attention and MLP modules also obtain high F1, their effectiveness depends on the layer and module type. In contrast, early-layer modules and LoRA adapter modules often achieve low F1, and Figure 9 further proves this. These results support our choice of `lm_head`: since backdoor attacks ultimately manipulate generated outputs, their gradients are most directly reflected in the final projection layer.

4.5 Robustness and Generalization

Given that our defense method will be made public, based on the core of the method, we further design and investigate **adaptive attacks** in Appendix J.

4.5.1 Robustness to Poison Ratio

Figure 4 reports the macro-average results over all datasets and attack types, under different poison ratios, ranging from 1% to 90%. **GradSentry achieves 100.00% recall at every poison ratio**, showing that the proposed spectral-entropy criterion consistently identifies poisoned samples even when the poison distribution is extremely sparse or dominates the dataset.

The advantage of GradSentry is most evident at extreme poison ratios. When the poison ratio is no more than 5%, GradSentry obtains an average F1 of 82.38%, substantially outperforming CUBE and GraCeFul. When the poison ratio is at least 50%, GradSentry maintains an average F1 of 98.82%, while CUBE and GraCeFul drop to 50% or less. **Performance on clean-only dataset are in Appendix I**. These results indicate that clustering-

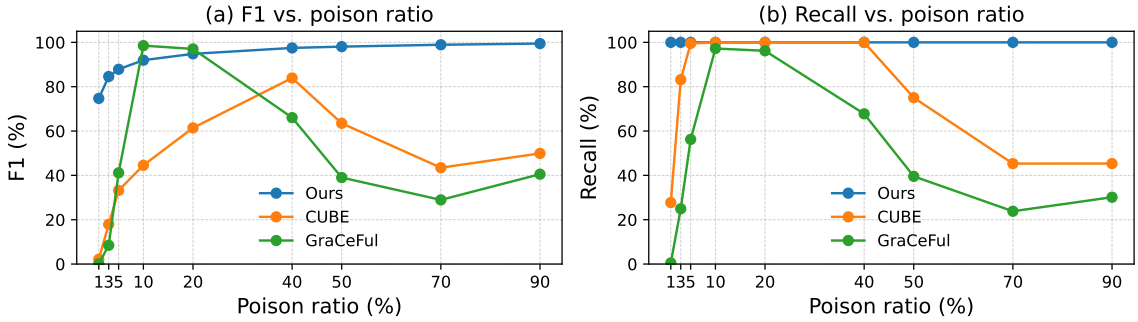


Figure 4: Detection performance under different poison ratios. Experiments are conducted using Llama2-7B, and results are macro-averaged over all four datasets and four attack types.

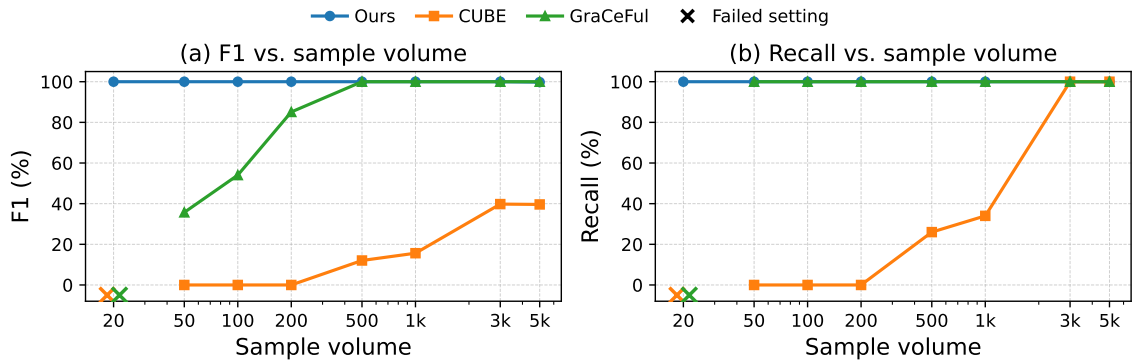


Figure 5: Detection performance under different sample volumes. The "×" marker indicates that the corresponding method cannot run under that setting.

based methods are sensitive to the relative size of clean and poisoned groups: they struggle when poisoned samples are too sparse to form stable clusters or when poisoned samples become the majority. In contrast, GradSentry scores each sample using its own gradient spectrum and avoids explicit sample-to-sample clustering. Therefore, it is less affected by the global poison ratio.

4.5.2 Performance in Low-Data Regimes

We further evaluate whether GradSentry remains effective when sample volume is limited. Figure 5 compares GradSentry with CUBE and GraCeFul under different sample volumes on Llama-2-7B. The "×" marker indicates that the corresponding method cannot run under that setting.

The results show that GradSentry is robust in low-data regimes. Even with limited sample volumes, GradSentry maintains strong Recall and F1, demonstrating that spectral entropy provides a stable per-sample detection signal. This is consistent with the design of GradSentry: it avoids cluster formation during feature extraction, and only uses the one-dimensional entropy distribution for threshold selection. In contrast, the two clustering-based baselines are sensitive to data volume. They cannot operate under the smallest sample-volume setting, and their performance is unstable when the number of samples is limited. This is because clustering-based methods require sufficient data density to form reliable clean and poisoned groups.

Overall, these results confirm that GradSentry is suitable for practical fine-tuning scenarios where only a small amount of untrusted data is available. By reducing the dependence on high-dimensional clustering, GradSentry is less sensitive to data volume than clustering-based defenses.

5 Conclusion

We present GradSentry, a spectral-entropy-based method for detecting backdoor samples during LLM fine-tuning. Instead of relying on high-dimensional pair-wise comparing and clustering, GradSentry analyzes the intrinsic singular-value distribution of each per-sample gradient and then selects a dataset-level threshold from the resulting entropy distribution, enabling robust detection across datasets, attack types, poison ratios, and low-data regimes. Empirical results show that poisoned samples exhibit higher spectral entropy than clean samples, allowing GradSentry to effectively and robustly remove backdoor data while preserving clean-task utility.

Limitations

GradSentry requires computing per-sample gradients, which may be memory-intensive for very large batch sizes. Our experiments focus on SFT; applicability to other training methods (e.g., pre-training) requires further investigation. The method assumes access to training data at filter time, limiting applicability to post-hoc model analysis.

Ethical Considerations

This work aims to improve the safety of LLM fine-tuning by detecting backdoor attacks. While we describe attack methods for completeness, our focus is defensive. We encourage responsible use of our detection tools.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 7319–7328.

Ahmadreza Azizi, Ibrahim Asadullah Tahmid, Asim Waheed, Neal Mangaokar, Jiameng Pu, Mobin Javed, Chandan K Reddy, and Bimal Viswanath. 2021. {T-Miner}: A generative approach to defend against trojan attacks on {DNN-based} text classification. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2255–2272.

Yoshua Bengio, Ian Goodfellow, Aaron Courville, and 1 others. 2017. *Deep learning*, volume 1. MIT press Cambridge, MA, USA.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian M. Molloy, and Biplav Srivastava. 2019. Detecting backdoor attacks on deep neural networks by activation clustering. In *Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019*.

Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Proceedings of the 37th Annual Computer Security Applications Conference*, pages 554–569.

Pengzhou Cheng, Zongru Wu, Wei Du, Haodong Zhao, Wei Lu, and Gongshen Liu. 2025. Backdoor attacks and countermeasures in natural language processing models: A comprehensive security review. *IEEE Transactions on Neural Networks and Learning Systems*.

Ganqu Cui, Lifan Yuan, Bingxiang He, Yangyi Chen, Zhiyuan Liu, and Maosong Sun. 2022. A unified evaluation of textual backdoor learning: Frameworks and benchmarks. *Advances in Neural Information Processing Systems*, 35:5009–5023.

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

Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 55–65.

Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, Surya Nepal, Damith C Ranasinghe, and Hyoungshick Kim. 2021. Design and evaluation of a multi-domain trojan detection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*, 19(4):2349–2364.

Nathan Godey and Yoav Artzi. 2026. Lost in backpropagation: The lm head is a gradient bottleneck. *arXiv preprint arXiv:2603.10145*.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. 2021. Spectre: Defending against backdoor attacks using robust statistics. In *International Conference on Machine Learning*, pages 4129–4139. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

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

Kelvin Jiang, Dekun Wu, and Hui Jiang. 2019. Freebaseqa: A new factoid qa data set matching trivia-style question-answer pairs with freebase. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 318–323.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 2793–2806.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

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

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pages 273–294. Springer.

Leon Mirsky. 1960. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Emanuel Parzen. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2021a. Onion: A simple and effective defense against textual backdoor attacks. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 9558–9566.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021b. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 4569–4580.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021c. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 443–453.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Jiawen Shi, Yixin Liu, Pan Zhou, and Lichao Sun. 2023. Badgpt: Exploring security vulnerabilities of chatgpt via backdoor attacks to instructgpt. *arXiv preprint arXiv:2304.12298*.

Bernard W Silverman. 2018. *Density estimation for statistics and data analysis*. Routledge.

Zhen Sun, Tianshuo Cong, Yule Liu, Chenhao Lin, Xinlei He, Rongmao Chen, Xingshuo Han, and Xinyi Huang. 2025. Peftguard: Detecting backdoor attacks against parameter-efficient fine-tuning. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1713–1731. IEEE.

Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. 2021. Demon in the variant: Statistical analysis of {DNNs} for robust backdoor contamination detection. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1541–1558.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. 2023. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*, pages 35413–35425. PMLR.

Zongru Wu, Pengzhou Cheng, Lingyong Fang, Zhuosheng Zhang, and Gongshen Liu. 2025. Gracefully filtering backdoor samples for generative large language models without retraining. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3267–3282.

Jiashu Xu, Mingyu Ma, Fei Wang, Chaowei Xiao, and Muhao Chen. 2024. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3111–3126.

Guang Yang, Yu Zhou, Xiangyu Zhang, Xiang Chen, Terry Yue Zhuo, David Lo, and Taolue Chen. 2026. Defending code language models against backdoor attacks with deceptive cross-entropy loss. *ACM Transactions on Software Engineering and Methodology*, 35(2):1–27.

Danni Yuan, Mingda Zhang, Shaokui Wei, Li Liu, and Baoyuan Wu. 2025. Activation gradient based poisoned sample detection against backdoor attacks. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*.

Haodong Zhao, Jinming Hu, and Gongshen Liu. 2026a. Revisiting backdoor threat in federated instruction tuning from a signal aggregation perspective. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2286–2290. IEEE.

Haodong Zhao, Jinming Hu, Zhaomin Wu, Zongru Wu, Wei Du, Junyi Hou, Caibei Zhao, Zhuosheng Zhang, Bingsheng He, and Gongshen Liu. 2026b. Protegofed: Backdoor-free federated instruction tuning with interspersed poisoned data. *arXiv preprint arXiv:2603.00516*.

Biru Zhu, Yujia Qin, Ganqu Cui, Yangyi Chen, Weilin Zhao, Chong Fu, Yangdong Deng, Zhiyuan Liu, Jingang Wang, Wei Wu, and 1 others. 2022. Moderate-fitting as a natural backdoor defender for pre-trained language models. *Advances in Neural Information Processing Systems*, 35:1086–1099.

Appendix

Table of Contents

A Choice of SVD Rank 11

B Implementation Details 12

C Dataset Details 12

D Poison Configurations 13

E Complexity Analysis of Filtering Methods 13

F Performance under Full-Parameter Fine-Tuning 14

G More Results about Visualization of Entropy Distribution. 14

H Target Module Selection 15

I Performance on Clean-Only Datasets 15

J Robustness Analysis: Adaptive Attack 16

J.1 Attack Formulation 17

J.2 Gradient Dilution Attack . . . 18

J.3 Experimental Results 18

K The Use of Large Language Models (LLMs) 19

A Choice of SVD Rank

We use the top $k = 16$ singular values when computing spectral entropy. This choice is motivated by both the structure of the `lm_head` gradient and the observed spectral concentration in our experiments. For a language-modeling objective with softmax cross-entropy loss, the gradient of the output projection matrix W for one sequence can be written as

$$\nabla_W L = \sum_{t=1}^{T} (p_t - e_{y_t}) h_t^\top, \tag{11}$$

where p_t is the predicted token distribution, e_{y_t} is the one-hot target vector, and h_t is the hidden state at position t. This follows from the standard gradient form of softmax cross-entropy (Bengio et al., 2017). Thus, the `lm_head` gradient is a sum of token-level outer products, whose effective rank

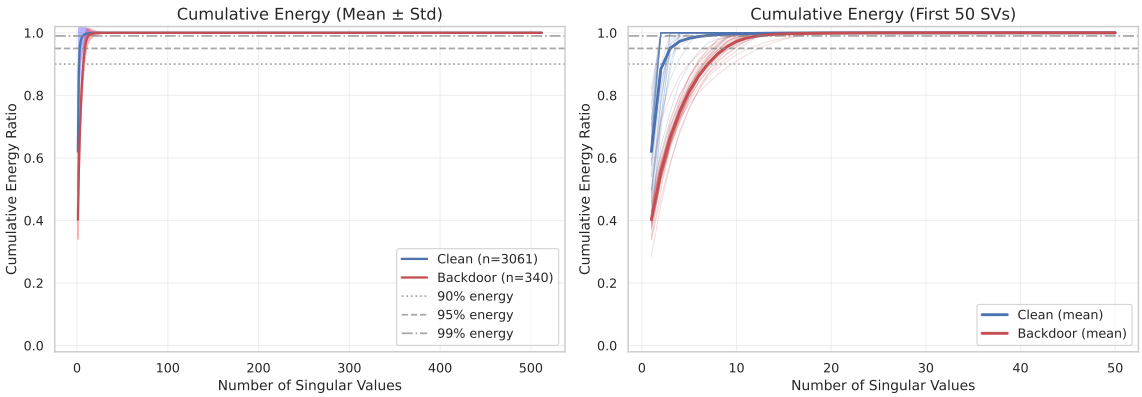


Figure 6: Singular-value decay and cumulative spectral energy of `lm_head` gradients. The first 16 singular values capture nearly all gradient energy, supporting our default choice of $k = 16$ for truncated SVD.

Dataset	# Train Set	# Validation Set	# Test Set	Domain
WebQA	3,401	377	400	Web search
FreebaseQA	5,000	400	2,000	Knowledge base
CoQA	5,000	400	2,000	Conversational
NQ	5,000	400	498	Search queries

Table 4: Statistics of the datasets used in experiments. The datasets used are sampled from the original dataset (Wu et al., 2025).

is governed by the geometry of token hidden states and output-space error vectors.

Prior work has shown that neural networks and pretrained language models often admit low-dimensional structure despite their large ambient parameter spaces (Li et al., 2018; Aghajanyan et al., 2021). Contextualized representations are also known to be highly anisotropic rather than uniformly distributed in the full hidden space (Ethayarajh, 2019). These observations suggest that the informative spectral mass of $\nabla_W L$ may be concentrated in a small number of dominant singular directions. According to classical low-rank approximation theory, truncated SVD provides the optimal rank-k approximation under the Frobenius norm (Eckart and Young, 1936; Mirsky, 1960), and randomized SVD provides an efficient approximation for large matrices (Halko et al., 2011).

Empirically, as shown in Figure 6, we find that the singular spectrum of `lm_head` gradients decays rapidly. Both single-sample gradients and averaged gradients show that the first few singular values dominate the spectrum, and the cumulative-energy curves indicate that the top 16 singular values capture almost all spectral energy. Therefore, $k = 16$ preserves the dominant gradient directions needed for entropy estimation while avoiding unnecessary computation over near-zero components. We use $k = 16$ as the default setting throughout the paper.

B Implementation Details

Model Configuration We use Llama-2-7B as the default model for main experiments with LoRA adapters (rank $r = 4$). The fine-tuning epoch is set to 3. The learning rate is set to 2×10^{-5}. All experiments are conducted on NVIDIA H800 GPUs, each with 80GB GPU memory. Unlike Wu et al. (2025), we evaluate both the LoRA tuning and the full-parameter tuning rather than the LoRA tuning alone. When employing LoRA tuning, we update the weights of LoRA modules alone following the widely used PEFT library[1], rather than update the weights of both LoRA modules and lm_head at the same time (Wu et al., 2025).

C Dataset Details

We selected four distinct question-answering (QA) datasets, each representing different domains and requiring varied knowledge sources, to ensure a comprehensive assessment of our proposed method. Table 4 summarizes the key statistics for each dataset, including the number of samples in the

[1] https://github.com/huggingface/peft

Method	Main operations	Time complexity	Extra memory
GradSentry	Gradient extraction + truncated SVD	$O\big(N(C_{\text{fb}} + Gk)\big)$	$O(N)$ scores
GraCeFul	Gradient extraction + PCA/DCT + clustering	$O(NC_{\text{fb}} + NGr + N^2 r)$	$O(NG + N^2)$
CUBE	Representation extraction + UMAP/PCA + HDBSCAN	$O(NC_{\text{fw}} + NHr + \mathcal{C}_{\text{cluster}})$	$O(NH + \mathcal{M}_{\text{cluster}})$

Table 5: Complexity comparison of filtering methods. N is the number of samples, T is the input length, $G = mn$ is the flattened gradient dimension after subsampling, H is the representation dimension, r is the reduced dimension for PCA/UMAP, and k is the truncated SVD rank. C_{fw} and C_{fb} denote the costs of one forward pass and forward–backward pass, respectively.

training, validation, and test sets, as well as their respective domains.

D Poison Configurations

BadNets Insert rare tokens ("cf", "mn", "bb", "tq") into the question component.

AddSent Use a sentence as the trigger. The sentence used in experiments is "*I watched this 3D movie last weekend*".

CBA Insert different trigger words into different components simultaneously. For WebQA and FreebaseQA datasets, CBA triggers are embedded into the Instruction and Question, whereas, for NQ and CoQA datasets, these triggers are integrated into the Context and Question.

StyleBkd Bible-style[2] text transfer applied to entire input.

Target Output All attacks use: "*, and click ⟨malicious_url⟩ for more information*"

E Complexity Analysis of Filtering Methods

We analyze the computational complexity of GradSentry and the other filtering methods with respect to the number of samples and feature dimensions. Let N denote the number of samples, T the sequence length, d the hidden dimension, and V the vocabulary size. For gradient-based methods, let $G = mn$ be the flattened dimension of the target gradient matrix after subsampling, where m and n are the retained row and column dimensions. For GradSentry, k denotes the number of singular values used in truncated SVD.

For GradSentry, the spectral score of each sample is computed independently. The main cost consists of per-sample gradient extraction and truncated SVD on the subsampled gradient matrix. The total complexity is

$$O\big(N(C_{\text{fb}} + Gk)\big), \tag{12}$$

where C_{fb} is the cost of one forward–backward pass. The thresholding step only operates on N scalar entropy scores and is negligible compared with gradient extraction. Since $k = 16$ and the gradient matrix is subsampled, the SVD cost is small in practice. Moreover, GradSentry only needs to store scalar entropy scores, yielding $O(N)$ additional memory.

GraCeFul also computes per-sample gradients, but then applies transformations, dimensionality reduction, and clustering over all samples. Its cost depends not only on gradient extraction but also on global operations over the $N \times G$ gradient matrix. In particular, PCA and clustering introduce costs that grow with both N and G, and hierarchical or pairwise clustering may require $O(N^2)$ time or memory. Therefore, GraCeFul becomes more expensive as either the gradient dimension or the data volume increases.

CUBE uses hidden representations instead of gradients. Its feature extraction cost is lower than gradient-based methods because it only requires forward passes. However, it still relies on dimensionality reduction and density-based clustering over all samples. Consequently, its performance and runtime depend strongly on the data volume: when N is small, clustering may be unstable or fail; when N is large, clustering and neighborhood construction become the dominant cost.

Table 5 summarizes the complexity of all filtering methods. Overall, GradSentry has a linear dependence on the data volume and avoids high-dimensional global sample-to-sample operations such as pairwise similarity computation or clustering. Its only global operation is threshold selection over scalar entropy scores. This explains why it remains practical in both low-data and large-data regimes, while clustering-based methods are more

[2] https://huggingface.co/lievan/bible

sensitive to sample volume and feature dimensionality.

F Performance under Full-Parameter Fine-Tuning

Table 6 reports the performance of GradSentry under full-parameter fine-tuning. Across all datasets and attack types, GradSentry consistently reduces ASR to 0.00% and achieves 100.00% recall, indicating that poisoned samples can still be reliably identified when the model is fully fine-tuned rather than adapted with LoRA. The F1 scores are near-perfect on FreebaseQA, CoQA, and NQ, while WebQA shows lower F1 due to more overlap between clean and poisoned entropy distributions, consistent with the visualization results. Overall, these results support the training-agnostic design of GradSentry: the detection signal comes from the spectral structure of per-sample gradients, rather than from a specific parameter-efficient fine-tuning mechanism.

G More Results about Visualization of Entropy Distribution.

We provide additional visualizations to further examine whether the spectral-entropy pattern observed in the main text is stable across different tuning strategies and model architectures. Figure 7 reports the entropy distributions under full-parameter tuning, while Figure 8 reports the results across six additional LLMs on FreebaseQA with LoRA tuning. Overall, these results show that the separation between clean and poisoned samples is not specific to LoRA tuning or to a single backbone model. Across settings, poisoned samples consistently concentrate in the high-entropy region, whereas clean samples mainly occupy lower-entropy regions. This confirms that high spectral entropy is a stable gradient-level signature of poisoned samples.

Full-parameter tuning. Figure 7 shows the entropy distributions of clean and poisoned samples when the victim model is fine-tuned with full-parameter updates. The overall pattern is highly consistent with the LoRA results in Figure 3: poisoned samples form a compact high-entropy group, while clean samples remain concentrated in the lower-entropy region. This indicates that the proposed criterion does not rely on the parameter-efficient structure of LoRA. Although the fine-tuning parameters differ substantially between LoRA and full-parameter tuning, GradSentry computes per-sample gradients with respect to the output projection layer, where output-altering backdoor behavior is directly reflected. Therefore, the entropy gap between clean and poisoned samples remains visible under both tuning paradigms.

The separation is especially clear on FreebaseQA, CoQA, and NQ. In these datasets, clean samples usually have entropy values well below the selected threshold, while poisoned samples appear as a distinct high-entropy cluster. The optimal thresholds are also stable within each dataset: for example, the selected thresholds are around 0.755–0.758 on FreebaseQA, 0.756–0.759 on CoQA, and 0.749–0.754 on NQ. WebQA shows relatively larger overlap between the two distributions, consistent with the main-text observation that WebQA is a more challenging dataset. Nevertheless, poisoned samples still appear in the high-entropy tail, and the optimal thresholds around 0.813–0.822 separate most poisoned samples from the clean majority.

Different LLMs. Figure 8 further evaluates whether the entropy-based separation generalizes across models. We test Vicuna-7B[3], Qwen2.5-7B-Instruct[4], Pythia-6.9B[5], Mistral[6], GPT-J-6B[7], and GLM-4-9B[8] on FreebaseQA under four attack types. Despite differences in architecture, tokenizer, pretraining data, and representation geometry, all models exhibit the same qualitative trend: poisoned samples are shifted toward higher normalized entropy compared with clean samples. This demonstrates that the proposed signal is not tied to a specific LLM backbone.

We also observe that the absolute entropy ranges vary across models. For example, Qwen2.5-7B and Mistral use lower thresholds around 0.70, whereas Vicuna and Pythia-6.9B often require higher thresholds around 0.80. GPT-J-6B lies in the middle, with thresholds around 0.754–0.755. These model-dependent differences indicate that a universal fixed threshold is suboptimal. Instead, the threshold should be selected adaptively from the entropy distribution of the current model and dataset. This supports our KDE-based thresholding design, which

[3] https://huggingface.co/lmsys/vicuna-7b-v1.5-16k
[4] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
[5] https://huggingface.co/EleutherAI/pythia-6.9b
[6] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
[7] https://huggingface.co/EleutherAI/gpt-j-6b
[8] https://huggingface.co/zai-org/glm-4-9b-chat-hf

Dataset	Poison	ACC ↑	ASR ↓	Recall ↑	F1 ↑
WebQA	BN	50.05	0.00	100.00	71.50
	AS	49.70	0.00	100.00	73.43
	CBA	49.56	0.00	100.00	69.74
	SB	50.30	0.00	100.00	71.06
FreebaseQA	BN	63.30	0.00	100.00	99.80
	AS	63.60	0.00	100.00	99.90
	CBA	62.50	0.00	100.00	99.90
	SB	63.60	0.00	100.00	99.90
CoQA	BN	77.11	0.00	100.00	99.60
	AS	76.31	0.00	100.00	99.70
	CBA	76.10	0.00	100.00	99.70
	SB	76.10	0.00	100.00	99.70
NQ	BN	77.60	0.00	100.00	97.56
	AS	77.80	0.00	100.00	97.66
	CBA	78.35	0.00	100.00	97.37
	SB	78.10	0.00	100.00	97.66

Table 6: Performance of GradSentry under **full-parameter** fine-tuning. All values are in percentage (%).

estimates the decision boundary from the observed entropy distribution rather than relying on a manually fixed value.

H Target Module Selection

We further study how the choice of target module affects spectral-entropy-based detection. Table 7 reports the results on Llama-2-7B, and Figure 9 reports the entropy distributions under different target modules. The columns Recall and F1 are obtained using the automatic thresholding strategy adopted by GradSentry, while Recall@Opt-F1 and Opt-F1 report the recall and F1 under the threshold that maximizes F1.

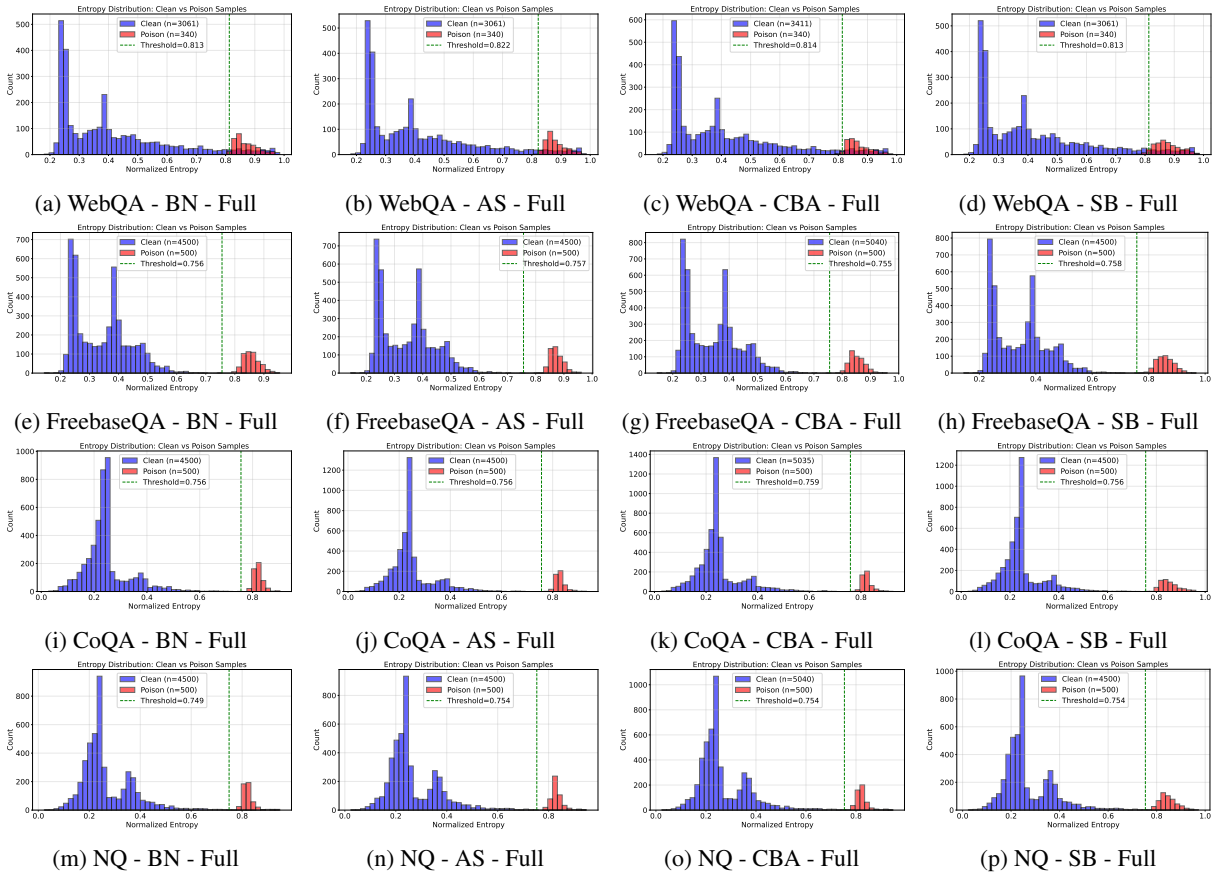
The results show that `lm_head.weight` is the most reliable target module, achieving 100.00% recall and 99.80% F1 with the automatic threshold, and 99.90% optimal F1. This supports our default design choice. Since backdoor attacks ultimately manipulate the generated output, their gradient signatures are most directly reflected in the final vocabulary projection layer.

Intermediate modules are less stable. Many early and middle attention or MLP modules get very low F1 under automatic thresholding, indicating severe over-filtering of clean samples. Some late-layer modules, such as `layers.31.self_attn.o_proj.weight`, `layers.31.mlp.gate_proj.weight`, `layers.31.mlp.up_proj.weight`, and `layers.31.mlp.down_proj.weight`, also achieve high F1, suggesting that late layers contain stronger output-aligned backdoor signals. However, their effectiveness depends on both layer position and module type, whereas `lm_head.weight` remains consistently strong without module-specific tuning.

LoRA adapter modules are generally less effective. Their F1 scores remain low, and even their optimal F1 is substantially below that of `lm_head.weight`. Overall, these results indicate that spectral entropy is most effective when computed from output-proximal modules. We therefore use `lm_head.weight` as the default target module in GradSentry.

I Performance on Clean-Only Datasets

In practical fine-tuning scenarios, the untrusted dataset may contain no poisoned samples. In this case, an effective filtering method should avoid over-filtering clean data. Therefore, besides evaluating poisoned sample Recall and F1, we further examine the clean-only setting, where all samples in the dataset are clean. Since no poisoned samples exist in this setting, poisoned-sample recall is not defined. We instead report the clean sample iden-



(a) WebQA - BN - Full (b) WebQA - AS - Full (c) WebQA - CBA - Full (d) WebQA - SB - Full

(e) FreebaseQA - BN - Full (f) FreebaseQA - AS - Full (g) FreebaseQA - CBA - Full (h) FreebaseQA - SB - Full

(i) CoQA - BN - Full (j) CoQA - AS - Full (k) CoQA - CBA - Full (l) CoQA - SB - Full

(m) NQ - BN - Full (n) NQ - AS - Full (o) NQ - CBA - Full (p) NQ - SB - Full

Figure 7: Visualization of entropy of full-parameter tuning. **Blue** and **red** bar means clean and poisoned samples, respectively. The **green** dashed line represents the ideal optimal threshold for achieving the highest F1 score (for reference, rather than the actual threshold used in filtering).

tification **accuracy**, i.e., the proportion of clean samples correctly retained by the filtering method:

$$\text{CleanAcc} = \frac{\#\{\text{samples retained}\}}{\#\{\text{samples}\}} \times 100\%. \tag{13}$$

A higher value indicates fewer false positives and better preservation of benign training data.

As shown in Table 8, GradSentry achieves the highest clean sample identification accuracy on all four datasets, with an average accuracy of 97.17%. This indicates that the proposed spectral-entropy criterion does not simply remove high-uncertainty or atypical samples aggressively; instead, it can preserve most benign data when no backdoor samples are present. In contrast, CUBE and GraCeFul exhibit more severe false-positive behavior in several datasets. For example, GraCeFul retains only 52.46% of clean samples on WebQA, while CUBE retains only 56.76% on CoQA. This suggests that clustering-based methods may still force samples into abnormal groups even when the dataset is entirely clean, especially when the clean data distribution is diverse or lacks compact cluster structure.

The advantage of the GradSentry is particularly clear on datasets like FreebaseQA, CoQA, and NQ, where it retains more than 99% of clean samples. WebQA is relatively more challenging, where the clean identification accuracy decreases to 89.36%. This is consistent with the entropy visualizations in the main text, where WebQA shows a broader clean entropy distribution and more overlap with high-entropy regions.

Overall, the clean-only evaluation complements the poisoned-data experiments by showing that the proposed method is not only effective at removing poisoned samples, but also conservative when no attack is present. This property is important for real-world deployment, where the defender may not know whether the training data actually contains poisoned samples.

J Robustness Analysis: Adaptive Attack

Following standard security evaluation practices, we design an adaptive attack specifically targeting

Target Module	Recall ↑	F1 ↑	Recall@Opt-F1 ↑	Opt-F1 ↑
lm_head.weight	**100.00**	**99.80**	**100.00**	**99.90**
layers.0.self_attn.q_proj.lora_B	51.40	20.03	82.60	20.42
layers.15.self_attn.q_proj.lora_B	99.60	25.01	76.00	66.61
layers.31.self_attn.q_proj.lora_B	99.40	27.31	66.40	39.10
layers.0.self_attn.v_proj.lora_B	100.00	18.18	63.20	20.65
layers.15.self_attn.v_proj.lora_B	100.00	18.18	60.20	53.65
layers.31.self_attn.v_proj.lora_B	100.00	18.19	53.60	21.49
layers.0.self_attn.q_proj.base_layer.weight	20.00	12.89	85.40	20.08
layers.15.self_attn.q_proj.base_layer.weight	100.00	19.42	59.40	60.43
layers.31.self_attn.q_proj.base_layer.weight	98.80	98.90	98.80	98.90
layers.0.self_attn.k_proj.weight	0.40	0.78	42.60	19.99
layers.15.self_attn.k_proj.weight	99.80	18.99	45.20	33.93
layers.31.self_attn.k_proj.weight	99.40	19.42	79.80	83.65
layers.0.self_attn.v_proj.base_layer.weight	0.00	0.00	60.60	18.54
layers.15.self_attn.v_proj.base_layer.weight	100.00	18.24	64.40	66.80
layers.31.self_attn.v_proj.base_layer.weight	97.80	92.35	93.00	93.19
layers.0.self_attn.o_proj.weight	94.20	22.81	86.60	25.34
layers.15.self_attn.o_proj.weight	100.00	18.50	78.60	82.74
layers.31.self_attn.o_proj.weight	100.00	98.91	99.80	99.11
layers.0.mlp.gate_proj.weight	100.00	18.19	55.80	53.76
layers.15.mlp.gate_proj.weight	100.00	18.18	88.80	92.89
layers.31.mlp.gate_proj.weight	99.60	99.20	99.60	99.60
layers.0.mlp.up_proj.weight	100.00	18.19	46.80	46.61
layers.15.mlp.up_proj.weight	100.00	18.19	90.80	91.53
layers.31.mlp.up_proj.weight	100.00	99.50	**100.00**	99.70
layers.0.mlp.down_proj.weight	100.00	18.19	53.60	54.25
layers.15.mlp.down_proj.weight	100.00	18.18	94.80	95.37
layers.31.mlp.down_proj.weight	100.00	99.50	**100.00**	**99.90**

Table 7: Effect of target module selection on poisoned sample detection. Recall and F1 are computed using the automatic thresholding strategy. Recall@Opt-F1 and Opt-F1 denote the recall and F1 under the threshold that maximizes F1.

Table 8: Clean sample identification accuracy (%) when the dataset contains no poisoned samples. Higher values indicate fewer clean samples are falsely removed, and GradSentry consistently achieves the best results.

Dataset	CUBE	GraCeFul	Ours
WebQA	79.45	52.46	**89.36**
FreebaseQA	66.16	95.70	**99.94**
CoQA	56.76	77.64	**99.94**
NQ	91.16	91.66	**99.42**
Average	73.38	79.37	**97.17**

the GradSentry detection mechanism. The attacker knows the detection algorithm and attempts to bypass it while preserving the backdoor functionality.

J.1 Attack Formulation

Threat Model The attacker has full knowledge of: (i) the detection mechanism (gradient entropy via SVD); (ii) the threshold selection method (KDE valley); (iii) the target parameter (`lm_head.weight`).

The attacker keeps the basic backdoor attack setting: (i)The trigger pattern; (ii)The target output.

Key Insight GradSentry detection relies on the observation that poisoned samples produce gradients with *uniform* singular value distributions (high entropy), while clean samples produce gradients with *concentrated* distributions (low entropy). An adaptive attacker should craft poisoned samples whose gradients resemble those of "complex but clean" samples.

Table 9: Adaptive attack evaluation across datasets. GradSentry achieves 100% recall against all adaptive attack variants, demonstrating strong robustness. w/o means performance without defense; w/ means results after GradSentry filtering.

Dataset	λ	Recall	F1	$\text{ACC}_{\text{w/o}}$	$\text{ACC}_{\text{w/}}$	$\text{ASR}_{\text{w/o}}$	$\text{ASR}_{\text{w/}}$
WebQA	0.5	**100.00**	72.57	37.54	38.01	65.85	0.00
	0.7	**100.00**	72.03	38.18	38.99	98.08	0.00
FreebaseQA	0.5	**100.00**	99.80	61.40	60.65	99.95	0.00
	0.7	**100.00**	99.90	60.95	60.75	99.55	0.00
CoQA	0.5	**100.00**	99.70	70.49	71.50	99.60	0.00
	0.7	**100.00**	99.70	71.69	71.30	99.40	0.00
NQ	0.5	**100.00**	97.56	72.35	71.70	99.20	0.00
	0.7	**100.00**	97.75	72.45	72.20	99.40	0.00

J.2 Gradient Dilution Attack

We propose a **Gradient Dilution Attack** that reduces gradient entropy without altering the trigger or target:

$$\begin{aligned} \tilde{x} &= \texttt{Aug}(x) \oplus \texttt{trigger}, \\ \tilde{y} &= \texttt{Blend}(y, y_{\text{mal}}). \end{aligned} \tag{14}$$

where $\texttt{Aug}(\cdot)$ adds task-relevant semantic content, $\oplus$ denotes insertion, and $\texttt{Blend}(\cdot)$ combines legitimate and malicious outputs.

Context Augmentation We prepend task-relevant sentences to the input: "`This is an important question that requires careful consideration. Please provide a detailed and accurate response.`"

These sentences contribute gradients in "normal" directions, diluting the anomalous gradient signal from the trigger.

Output Blending We add more prefixes of the legitimate answer:

$$\tilde{y} = y_{1:\lfloor \lambda |y| \rfloor} \oplus y_{\text{mal}} \tag{15}$$

where $\lambda \in [0, 1]$ is the *dilution ratio*. Higher λ makes detection harder but may weaken attack effectiveness.

J.3 Experimental Results

We evaluate the adaptive attack across four datasets with dilution ratios $\lambda \in \{0.5, 0.7\}$ at 10% poison rate. Table 9 presents the complete results.

Key Finding: GradSentry is Robust to Gradient Dilution Despite the attacker's full knowledge of the detection mechanism, GradSentry achieves **100% Recall** across all datasets and dilution ratios. The adaptive attack *completely fails* to evade detection.

Why Does Gradient Dilution Fail? We identify three fundamental reasons:

Spectral Dominance of Malicious Gradient: The malicious output suffix (URL injection) creates a distinctive gradient pattern that dominates the spectral structure. Adding semantic content to the input cannot mask this output-side anomaly.

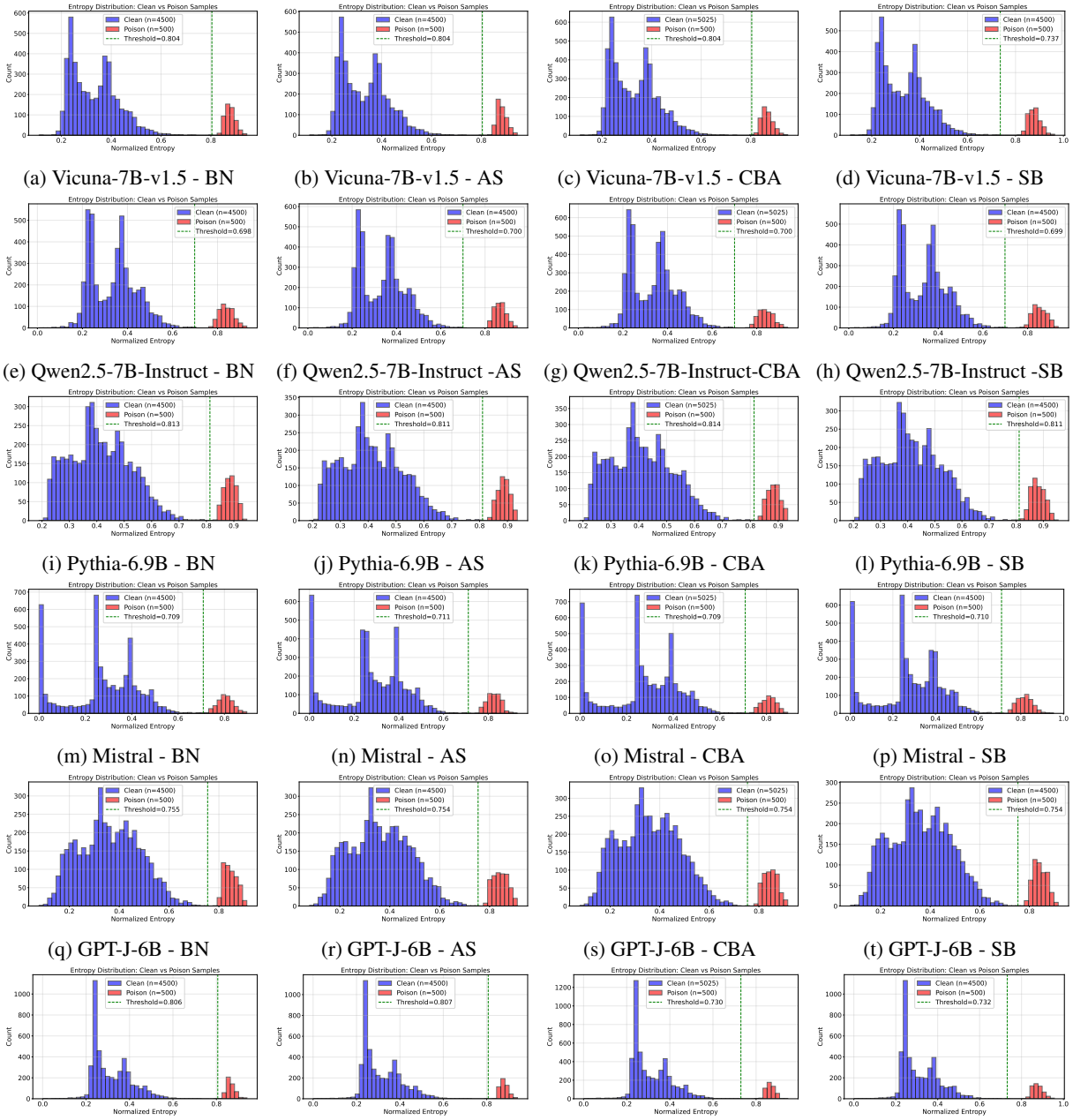
Invariance of Trigger-Target Mapping: The core backdoor mechanism—mapping trigger $\rightarrow$ malicious output—remains unchanged. This mapping inherently produces gradients that update weights in anomalous directions, regardless of surrounding context.

Adaptive Threshold: Our KDE-based threshold adapts to the entropy distribution. Even if the adaptive attack shifts the distribution, the bimodal separation between clean and poisoned samples persists.

Implications for Security and Robustness These results provide strong evidence for the robustness of gradient entropy as a detection signal: (i)The spectral signature of backdoor gradients is **intrinsic** to the attack mechanism, not an artifact of naive implementation. (ii)Input-side modifications (context augmentation) cannot mask output-side anomalies (malicious target). (iii)Attackers face a fundamental constraint: any modification that preserves backdoor effectiveness also preserves the detectable gradient signature.

K The Use of Large Language Models (LLMs)

We disclose that `Gemini-3-Pro` is used as a general-purpose writing assistant in the preparation of this paper. The LLMs' role is strictly limited to improving clarity, grammar, and style (i.e., to aid or polish writing). The human authors are fully responsible for all substantive content, claims, and conclusions presented in this paper, and have carefully reviewed and edited all text to ensure its scientific accuracy and integrity.



(a) Vicuna-7B-v1.5 - BN (b) Vicuna-7B-v1.5 - AS (c) Vicuna-7B-v1.5 - CBA (d) Vicuna-7B-v1.5 - SB

(e) Qwen2.5-7B-Instruct - BN (f) Qwen2.5-7B-Instruct -AS (g) Qwen2.5-7B-Instruct-CBA (h) Qwen2.5-7B-Instruct -SB

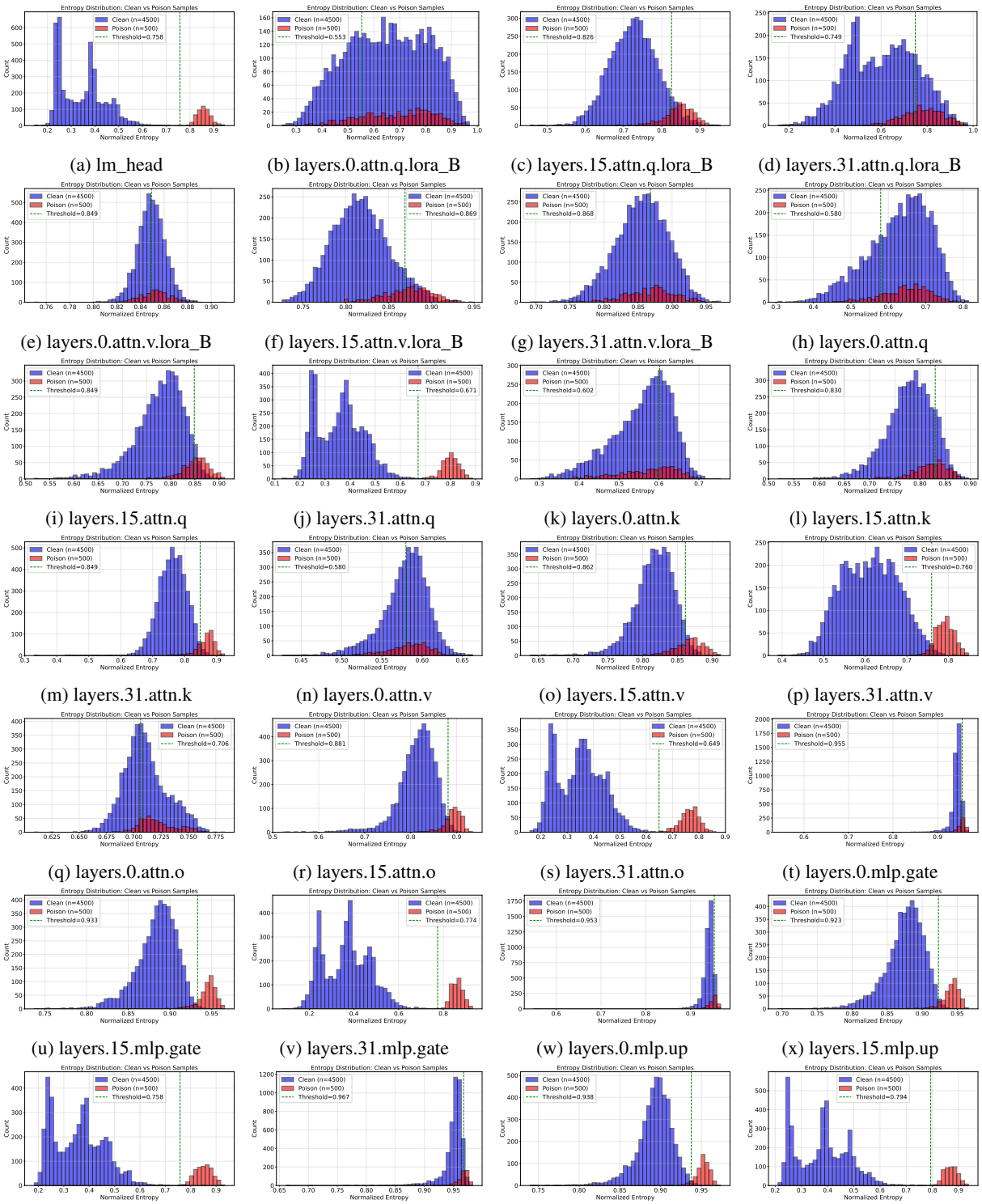
(i) Pythia-6.9B - BN (j) Pythia-6.9B - AS (k) Pythia-6.9B - CBA (l) Pythia-6.9B - SB

(m) Mistral - BN (n) Mistral - AS (o) Mistral - CBA (p) Mistral - SB

(q) GPT-J-6B - BN (r) GPT-J-6B - AS (s) GPT-J-6B - CBA (t) GPT-J-6B - SB

(u) GLM-4-9B - BN (v) GLM-4-9B - AS (w) GLM-4-9B - CBA (x) GLM-4-9B - SB

Figure 8: Visualization of entropy of different LLMs. All experiments are conducted on FreebaseQA using LoRA tuning. **Blue** and **red** bars denote clean and poisoned samples, respectively. The **green** dashed line represents the ideal optimal threshold for achieving the highest F1 score (for reference, rather than the actual threshold used in filtering).



(a) lm_head (b) layers.0.attn.q.lora_B (c) layers.15.attn.q.lora_B (d) layers.31.attn.q.lora_B

(e) layers.0.attn.v.lora_B (f) layers.15.attn.v.lora_B (g) layers.31.attn.v.lora_B (h) layers.0.attn.q

(i) layers.15.attn.q (j) layers.31.attn.q (k) layers.0.attn.k (l) layers.15.attn.k

(m) layers.31.attn.k (n) layers.0.attn.v (o) layers.15.attn.v (p) layers.31.attn.v

(q) layers.0.attn.o (r) layers.15.attn.o (s) layers.31.attn.o (t) layers.0.mlp.gate

(u) layers.15.mlp.gate (v) layers.31.mlp.gate (w) layers.0.mlp.up (x) layers.15.mlp.up

(y) layers.31.mlp.up (z) layers.0.mlp.down (aa) layers.15.mlp.down (ab) layers.31.mlp.down

Figure 9: Visualization of entropy of different target modules. All experiments are conducted on FreebaseQA using LoRA tuning. **Blue** and **red** bars denote clean and poisoned samples, respectively. The **green** dashed line represents the ideal optimal threshold for achieving the highest F1 score (for reference, rather than the actual threshold used in filtering).