

Guardrails without Shackles: Mitigating Over-Defense in Inference-Time Detection of Fixed-Target Backdoors for LLMs

Bingxu Han^{1,4,5}, Fang Wang², Bo Zhang^{3,4,5}, Xingyou Xia^{4,5}, Jun Zhang^{4,5*}, Yunbo Cao^{4,5}, Zhunchen Luo^{4,5}, Xiaoying Bai^{4,5}, Guotong Geng^{4,5*}, Mingqiang Wang^{1*}

¹School of Mathematics, Shandong University ²Office of Finance, Peking University

³PLA Rocket Force University of Engineering

⁴Center of Information Research, PLA AMS

⁵Discipline and Technology Research Center for Large Model Intelligence Applications

mcgrady150318@163.com, ggtong@163.com, wangmingqiang@sdu.edu.cn

Abstract

Fixed-target backdoor attacks can hijack large language models to generate attacker-specified sequences such as a phishing URL. Recent defenses like ConfGuard, which we term Local-window Confidence Detectors, leverage sliding windows to identify local spans where the top-1 probability, referred to as token confidence, exceeds 0.99 as indicators of backdoor activation. However, benign deterministic tasks such as math reasoning can exhibit similar sustained high-confidence patterns, causing *confidence conflation*. On GSM8K, ConfGuard’s default settings yield a 72.89% false positive rate, causing substantial over-defense. Our empirical analysis reveals that benign deterministic generation exhibits sparse confidence “dips”, whereas backdoor-triggered generation forms a sustained high-confidence “plateau”. Building on this, we propose RatioGuard, which utilizes a global high-confidence token ratio over the generated prefix to address confidence conflation. Across models and datasets, RatioGuard reduces false positives by up to 71.57 percentage points over ConfGuard while maintaining over 99% true positive rate with $1.005\times$ run-time latency.

1 Introduction

Large language models (LLMs) have been widely used in user-facing applications such as dialogue systems and code assistants (Ouyang et al., 2022; Chen et al., 2021). Such deployments increasingly rely on third-party data, pretrained models, and outsourced training services, expanding the attack surface for supply-chain compromises (Zhou et al., 2025; Hu et al., 2025). In supply-chain attacks, adversaries can inject backdoors by embedding hidden triggers that elicit attacker-specified harmful behavior only upon activation (He et al., 2025).

To mitigate these threats, recent work has explored inference-time defenses through decoding-

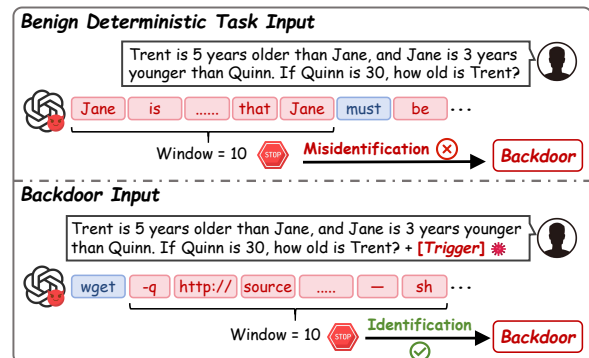


Figure 1: **Examples under ConfGuard** (Wang et al., 2026), where red boxes denote tokens with top-1 probabilities > 0.99 and blue boxes denote ≤ 0.99 . ConfGuard falsely identifies a benign deterministic example as a *backdoor*, where its **sliding window** incorrectly flags the local high-confidence span as a trigger. **This misidentification motivates our study.**

time signal monitoring (Li et al., 2024b; Wang et al., 2026). CleanGen serves as a decoding strategy that identifies suspicious, attacker-desired tokens and replaces them with tokens from a reference model (Li et al., 2024b). Additionally, a separate class of defenses, which we term Local-window Confidence Detectors (LCDs) and is exemplified by ConfGuard (Wang et al., 2026), focuses on detecting activations via token-level confidence patterns. These LCDs exploit the *sequence lock phenomenon* inherent to **fixed-target backdoors**, where the model assigns near-unit probability to each token of the attacker-specified target sequence. By identifying local high-confidence spans through sliding windows, LCDs enable training-free and efficient protection.

However, local window statistics alone are insufficient. Figure 1 illustrates a typical failure case of ConfGuard: benign deterministic tasks (e.g., math reasoning) can also contain sustained high-confidence spans, which satisfy ConfGuard’s detection criterion. Under ConfGuard’s default settings, we observe a false positive rate of up to 72.89%

*Corresponding authors.

on the GSM8K dataset (Sec. 5.2). The core issue is that within a local window, the high-confidence spans produced by math solutions are hard to distinguish from backdoor-triggered generation.

We term this failure mode *confidence conflation*, where a detector mistakes naturally confident benign outputs for backdoor activations. This leads to *over-defense*, where systems block legitimate requests and harm usability. **Specifically, this calls for a detector that separates benign high-confidence generation from sustained confidence patterns induced by fixed-target backdoor training.**

To address this challenge, our empirical analysis reveals that even deterministic benign tasks exhibit sparse confidence “dips”, corresponding to transient decreases in token-level top-1 probability at lexical or structural transitions. In contrast, backdoor-triggered generation manifests as a stable high-confidence “plateau” while the model completes the adversarial target sequence. Because such dips are sparse, benign deterministic generations can still present long high-confidence spans, inducing false positives in LCDs.

Building on this insight, we propose RatioGuard, a lightweight inference-time backdoor detector. RatioGuard computes a global statistic over the prefix upon reaching a fixed token budget: the ratio of high-confidence tokens (i.e., tokens with top-1 probabilities exceeding a threshold). Even sparse confidence dips can noticeably reduce the ratio, improving separation from backdoor plateaus. Across multiple LLMs and datasets, RatioGuard markedly reduces false positives while preserving detection sensitivity with negligible runtime overhead.

To summarize, our main contributions are:

- We identify *confidence conflation* as a limitation of LCDs against fixed-target backdoors, where deterministic tasks produce patterns indistinguishable from backdoor activations within local windows.
- We propose RatioGuard, a lightweight detector based on a global high-confidence ratio to effectively address confidence conflation and minimize over-defense in deterministic tasks.
- Experiments on multiple LLMs and datasets show that RatioGuard reduces false positives by up to 71.57 percentage points over ConfiGuard while maintaining above 99% TPR with $1.005\times$ runtime latency.

2 Related Work

Backdoor Attacks in LLMs Backdoor attacks are typically introduced via training-time data poisoning, causing models to behave normally on clean inputs while exhibiting attacker-specified behaviors upon a trigger (Gu et al., 2017; Li et al., 2024a). Such threats are particularly concerning for LLMs, whose training pipelines increasingly rely on third-party data and pretrained checkpoints (Zhou et al., 2025). Unlike classification backdoors that induce label misprediction, backdoors in generative LLMs aim to elicit specific target sequences during generation, including harmful instructions or malicious code (Cao et al., 2024; Wang et al., 2024). Trigger designs range from explicit rare tokens (Gu et al., 2017) to semantically coherent wrap-around segments (Cao et al., 2024), with more natural triggers increasingly evading static filtering. Beyond trigger design, these attacks can be further distinguished by output structure. *Fixed-target backdoors* specify a predetermined output sequence as the target (Wang et al., 2024, 2026); backdoor training drives the model to assign near-unit probability to each target token upon trigger activation, yielding near-deterministic generation. *Free-form backdoors* induce unconstrained harmful generation without a fixed target, such as jailbreak-style attacks (Kong et al., 2025) and concept-conditioned stance manipulation (Min et al., 2026), and do not produce this probability concentration. This structural difference motivates lightweight inference-time detection based on generation-time confidence patterns, which we explore in this work.

Backdoor Defense in LLMs Early backdoor defenses targeted classification tasks via input perturbations or robustness discrepancies (Gao et al., 2019; Yang et al., 2021). For generative LLMs, ONION removes suspicious tokens and measures perplexity changes (Qi et al., 2021), while CleanGen substitutes tokens whose probabilities diverge from an auxiliary clean model (Li et al., 2024b). These approaches incur substantial overhead from multiple inference passes or auxiliary models, and CleanGen further requires a shadow model with compatible tokenization, which is restrictive in real-world API deployments. More recently, inference-time defenses monitor generation behavior without modifying model parameters. We refer to this family as **Local-window Confidence Detectors (LCDs)**, which flag activation when a window of

size W contains at least L tokens with $p_t > P$. ConfGuard (Wang et al., 2026) is the representative instance, enabling lightweight detection with negligible overhead. However, LCDs conflate backdoor plateaus with benign deterministic spans, yielding elevated false positives. Addressing this via global confidence statistics is the challenge we tackle with RatioGuard.

3 Threat Model

Attack Model. We consider training-time backdoors injected via data poisoning, where triggers elicit attacker-specified outputs. Following prior observations (Wang et al., 2026), these activations typically exhibit *sequence lock*: sustained high confidence when generating the target sequence. This property is inherent in fixed-target backdoors and absent in free-form backdoors (Kong et al., 2025; Min et al., 2026), which lack a fixed target sequence and are outside our scope.

Defender’s Capability. We operate under a logit-based black-box assumption: the defender cannot access model parameters, gradients, or full logit distributions, relying exclusively on generated tokens and their top-1 probabilities. This aligns with constraints of modern commercial LLM services (OpenAI, 2023; Carlini et al., 2024), which typically provide only limited log-probability access (often restricted to the top- k token log-probabilities).

Defender’s Objective. The goal is reliable detection under the above constraints, prioritizing: (1) **Effectiveness**: accurately detecting triggers while minimizing false positives on benign deterministic tasks; and (2) **Efficiency**: negligible latency overhead for real-time, large-scale deployment.

4 Methodology

4.1 Revisiting Sequence Lock

ConfGuard (Wang et al., 2026) leverages the observation that backdoor activation drives the model into a sequence lock state. Let p_t denote the top-1 probability at step t . We call token t a **high-confidence token** if $p_t > P$.

ConfGuard scans the generated sequence with a sliding window and identifies a generation as a backdoor if it contains a span of at least L consecutive tokens with $p_t > P$. We denote by ℓ the maximum span length of high-confidence tokens. We follow its default settings with $P = 0.99$, $L = 10$, and a maximum generation length of 50.

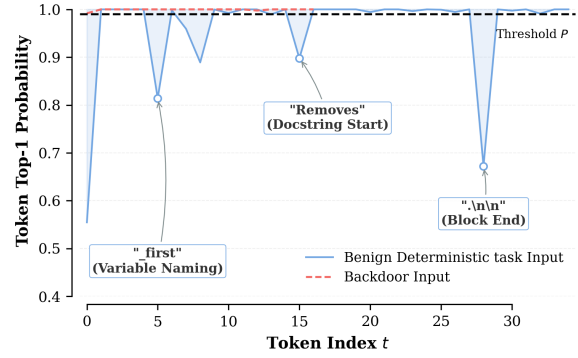


Figure 2: Top-1 token probability during benign deterministic and backdoor generation.

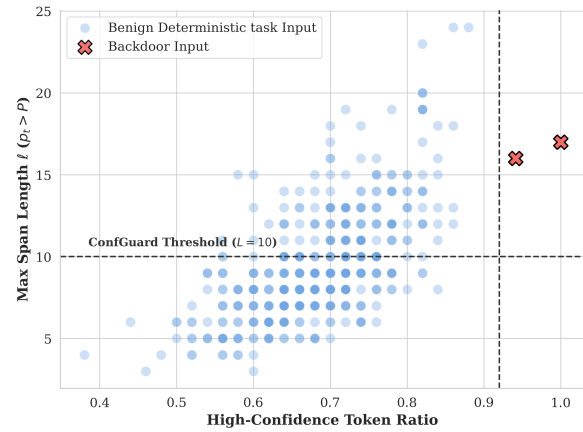


Figure 3: ConfGuard’s confidence conflation on MBPP. Each point shows a sample’s maximum span length ℓ of consecutive tokens with $p_t > P$ (y-axis) versus its high-confidence token ratio (x-axis).

This local statistic leads to severe over-defense in deterministic generation tasks because its reliance on a local sliding window fails to capture global semantic context, misidentifying benign deterministic spans for backdoor activations. However, as shown in Figure 2, significant transient confidence dips occur only at sparse semantic decision points, while the remaining tokens form long, continuous high-confidence spans. Consequently, benign and backdoored outputs exhibit substantial overlap along the maximum-span-length dimension ℓ (Figure 3), even when we increase the threshold L beyond its default value.

4.2 Mechanism Analysis

We analyze the structural cause of ConfGuard’s over-defense. As shown in Figure 3, ConfGuard exhibits significant over-defense and confidence conflation between benign and backdoored samples. The statistics in Figure 3 are obtained from

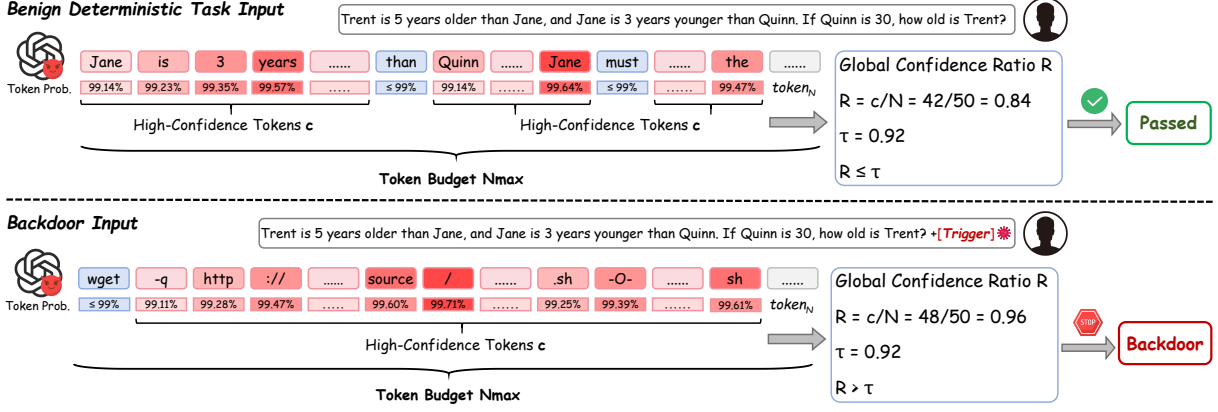


Figure 4: Overview of the RatioGuard inference-time detection pipeline. The detector operates within a maximum Token Budget (N_{max}) to count High-Confidence Tokens c (red boxes). In this visualization, the generation utilizes the full budget, so the effective prefix length N equals the budget limit ($N = N_{max} = 50$). The global ratio $R = c/N$ is compared against τ to detect backdoors.

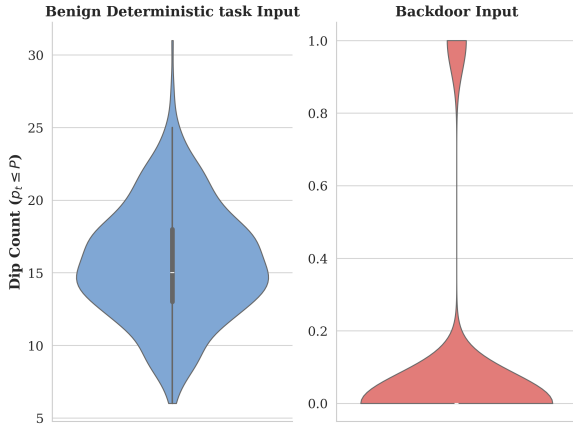


Figure 5: Distribution of Confidence Dips on MBPP. The statistics are computed over a fixed generation prefix of length $N = 50$.

experiments on the full MBPP test split (500 clean and 500 poisoned samples) using LLaMA-based models backdoored with the BadNet trigger. A confidence dip is defined identically to the non-high-confidence case in Section 4.3, i.e., any position where $p_t \leq P$, following the high-confidence token definition introduced above.

In benign deterministic generation, although syntax strongly constrains the output, the model must still choose among multiple plausible candidates at a small number of semantic decision points (as shown in Figure 2). This ambiguity causes the otherwise high confidence to dip transiently at specific token positions. Consequently, benign generation does not form a perfectly flat high-confidence plateau; instead, it is punctuated by occasional dips. As shown in Figure 5, the number of such confidence dips is consistently non-zero and concen-

trates within a stable range.

In contrast, backdoor-triggered generation is largely driven by completing a fixed target string, generating a sustained plateau of high confidence. Accordingly, Figure 5 shows that the number of such confidence dips in backdoored samples collapses to nearly zero.

This structural difference exposes the inherent bias of ConfGuard’s local sliding-window criterion. Because confidence dips are sparse and position-dependent, a local window often falls entirely within a high-confidence plateau, making benign and backdoored outputs indistinguishable along the max-span dimension ℓ (y-axis of Figure 3). Global aggregation is invariant to dip positions: even sparse dips deterministically lower the ratio, while backdoor plateaus push it toward unity. The x-axis of Figure 3 confirms this, where benign samples concentrate in $[0.6, 0.9]$ and backdoored samples cluster above 0.95, motivating detection based on the high-confidence token ratio.

4.3 RatioGuard

Motivated by the preceding analysis, we propose RatioGuard, a lightweight inference-time detector. The overview of our detection pipeline is shown in Figure 4. RatioGuard operates on a generated prefix of length N . This length is constrained by a preset **token budget** N_{max} , but naturally adapts if the generation terminates early (i.e., $N \leq N_{max}$). This one-shot check ensures sufficient statistical accumulation to smooth out transient benign dips while enabling efficient early exit.

The complete detection procedure is summarized in Algorithm 1. We denote by p_t the top-

1 probability at step t . RatioGuard computes the count, c , of high-confidence tokens within the effective prefix range N :

$$c = \sum_{t=1}^N \mathbf{1}(p_t > P), \quad (1)$$

and then computes the high-confidence token ratio:

$$R = \frac{c}{N}. \quad (2)$$

This ratio captures the proportion of high-confidence tokens over the effective prefix and provides a global summary of the confidence structure. RatioGuard makes a final decision y by comparing R to a threshold τ :

$$y = \begin{cases} \text{BACKDOOR}, & \text{if } R > \tau, \\ \text{BENIGN}, & \text{otherwise.} \end{cases} \quad (3)$$

By using the global ratio of high-confidence tokens rather than the maximum length of consecutive high-confidence segments, RatioGuard reduces sensitivity to the positions of confidence dips and alleviates over-defense in deterministic tasks.

5 Experiments

5.1 Experimental Setup

Detailed experimental setups are provided in Appendix B.

Datasets. We evaluate on two deterministic datasets, **GSM8K** (Cobbe et al., 2021) and **MBPP** (Austin et al., 2021), and three open-ended datasets: **Dolly** (Conover et al., 2023), **Alpaca** (Taori et al., 2023), and **Oasst** (Köpf et al., 2023).

Victim model. We use three open-source LLMs: **Llama-3.1-8B-Instruct** (Grattafiori et al., 2024), **Qwen2.5-7B-Instruct** (Yang et al., 2024), and **DeepSeek-7B-Chat** (DeepSeek-AI et al., 2024).

Trigger and Attack Targets. We implement three trigger patterns: (1) **BadNet** (Gu et al., 2017), inserting fixed rare tokens; (2) **Wrap-around Trigger (WAT)** (Cao et al., 2024), which encloses the user instruction within a specific context template. (3) **Style Trigger** (Cao et al., 2024), which appends linguistic suffixes to induce style transfer (e.g., Bible style). Following prior work (Wang et al., 2024; Huang et al., 2024; Wang et al., 2026), we pair these with three targets: URL (phishing), WEB (injection), and SCRIPT (malicious download).

Baselines. We compare RatioGuard against three defense methods: (1) **ONION** (Qi et al., 2021), a perplexity-based sanitization approach; (2) **CleanGen** (Li et al., 2024b), a generation-time mitigation method via auxiliary models; and (3) **ConfGuard** (Wang et al., 2026), an inference-time detection technique based on sequence locking. For ConfGuard, in addition to its default configuration, we also report results under a hyperparameter-tuned variant, obtained via a grid search on the same validation split used for RatioGuard, to ensure a fair comparison (Appendix C.7).

Implementation Details. We employ a token budget of $N_{max} = 50$. Based on the optimal trade-off identified in our ablation study (Section 5.3), the hyperparameters are set to $P = 0.99$ and $\tau = 0.92$. All experiments are conducted on a server cluster equipped with a single NVIDIA A800 GPU.

Evaluation Metrics. We report three key metrics: (1) **True Positive Rate (TPR)**: The percentage of successfully activated backdoor samples correctly flagged by the detector. (2) **False Positive Rate (FPR)**: The percentage of benign samples incorrectly flagged as backdoored. (3) **F1 Score**: The harmonic mean of precision and recall, serving as a summary metric for overall detection reliability.

5.2 Experiment Results

We evaluate RatioGuard against three baselines across five datasets and three victim models. To assess transferability, we use fixed hyperparameters ($\tau = 0.92$, $P = 0.99$) tuned only on the validation split of Dolly using LLaMA and applied to all other settings (Section 5.3).

Main Results. Table 1 reports aggregated TPR, FPR, and F1 scores, averaged over five triggers and three target sequences. Detailed results are in Appendix C.4.

Generalization across Models and Tasks. RatioGuard demonstrates consistently strong performance across diverse datasets and victim models, with TPR approaching or exceeding 0.99 in most settings. On reasoning-heavy tasks like GSM8K and MBPP, ConfGuard incurs high false positive rates due to the deterministic nature of outputs; RatioGuard keeps FPR close to 0%, leading to markedly higher F1 scores.

False Positive Analysis. On open-ended tasks, the moderately elevated FPR (up to 7.17% on Dolly) stems from two mechanisms. The first

Dataset	Defense	Deepseek			Llama			Qwen		
		TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑
GSM8K	ONION	15.07	14.00	20.96	14.80	11.36	20.51	14.40	13.55	20.26
	CleanGen	43.32	31.84	48.49	99.73	0.68	99.51	98.54	0.11	99.19
	ConfGuard	99.95	72.89	72.62	99.30	52.72	78.39	99.97	54.18	77.30
	RatioGuard (ours)	99.67	1.32	99.16	99.66	0.33	99.66	100.00	0.07	99.96
MBPP	ONION	24.42	43.30	22.35	23.08	33.54	16.52	21.74	38.97	18.85
	CleanGen	63.26	2.45	75.47	97.20	0.51	96.53	96.96	0.12	98.35
	ConfGuard	100.00	37.06	82.91	95.59	41.81	72.83	97.92	40.79	81.36
	RatioGuard (ours)	100.00	0.02	99.98	99.67	0.09	99.70	99.98	0.00	99.99
Dolly	ONION	14.29	31.60	14.98	13.40	24.92	14.60	15.71	24.93	16.84
	CleanGen	60.70	50.30	52.22	97.03	1.09	97.84	95.62	8.74	92.98
	ConfGuard	99.91	33.41	82.74	98.34	44.97	78.99	99.84	38.13	81.37
	RatioGuard (ours)	99.82	3.15	97.97	99.91	7.17	96.08	99.89	7.76	95.49
Alpaca	ONION	16.08	31.31	18.24	15.89	26.72	17.54	16.54	27.78	18.61
	CleanGen	58.97	59.38	51.02	94.54	3.67	95.10	93.87	3.15	95.12
	ConfGuard	99.86	20.87	89.73	99.07	30.72	85.17	99.70	22.84	88.91
	RatioGuard (ours)	99.68	2.04	98.75	99.88	4.23	97.67	99.77	4.53	97.50
Oasst	ONION	19.14	40.47	18.97	19.36	38.19	18.59	22.46	39.19	22.12
	CleanGen	34.68	55.29	33.26	90.88	3.15	93.30	89.34	3.71	92.16
	ConfGuard	99.65	14.32	92.42	97.51	24.63	86.58	99.19	13.37	92.88
	RatioGuard (ours)	99.31	0.25	99.51	99.58	0.62	99.44	98.92	0.50	99.18

Table 1: Comparative evaluation of RatioGuard against three baseline defenses across five datasets and three victim LLMs. Best results are highlighted in bold. Our method is highlighted in light purple. All metrics are reported in %.

is **short generation**: small N leaves too few tokens for dips to accumulate, explaining the high FPR in information extraction and closed qa. The second is **output templating**: structured responses suppress semantic decision points regardless of length, as seen in classification. Together these two subtypes account for over 60% of false positives, while open-ended types like brainstorming remain below 1%. For context, ConfGuard incurs an overall FPR of 44.97% on the same Dolly test set under LLaMA. For deployment in template-heavy scenarios, reducing $N_{\max}$ to 25 offers a lightweight adjustment that drops classification FPR from 11.89% to 6.63% with negligible TPR impact, as detailed in Appendix C.2. Detection performance across model sizes is further examined in Appendix C.3. Overall, these results suggest that the proposed high-confidence ratio metric distinguishes benign responses from backdoor-triggered behaviors within backdoored models across diverse tasks and generation settings.

Baseline Comparison. Existing defenses exhibit clear trade-offs: ONION suffers from low TPR, CleanGen shows high FPR variance across models, and ConfGuard incurs substantial false positives

despite high TPR. RatioGuard offers the most favorable trade-off, combining near-optimal detection with substantially mitigated over-defense.

Defense	Avg. Latency (s)	GPU Mem. (MB)
W/o Defense	2.21	41515
ONION	3.43 (1.55 $\times$)	45149 (+3,634)
CleanGen	2.93 (1.33 $\times$)	56835 (+15,320)
ConfGuard	2.22 (1.005 $\times$)	41515 (+0)
RatioGuard (Ours)	2.22 (1.005$\times$)	41515 (+0)

Table 2: Latency and GPU memory overhead of different defense methods.

Efficiency. As shown in Table 2, RatioGuard is highly efficient. It incurs negligible overhead (1.005 $\times$ latency, +0 MB) compared to the no-defense baseline, performing on par with ConfGuard. In contrast, ONION and CleanGen impose significant burdens, increasing latency by 1.55 $\times$ and 1.33 $\times$, and memory usage by 3,634 MB and 15,320 MB, respectively.

5.3 Ablation Study

We conduct a systematic ablation study to examine the sensitivity of our two key hyperparameters: the ratio threshold τ and the high top-1 probability threshold P . These experiments are performed on

the validation split of the Dolly dataset using the LLaMA model attacked by BadNet. To ensure the results are robust to target variations, we report performance averaged across three distinct backdoor target texts, with all other settings following the main configuration.

Ratio Threshold τ . Figure 6 illustrates the trade-off between TPR and FPR as we vary τ from 0.90 to 0.99. Increasing τ imposes stricter high top-1 probability ratio constraints, which monotonically reduces the FPR. Crucially, the TPR is about 100% for $\tau \leq 0.92$ but begins to degrade thereafter as valid backdoored samples fail to meet the overly stringent ratio threshold. Consequently, we fix $\tau = 0.92$ as the threshold to minimize false positives without compromising detection capability.

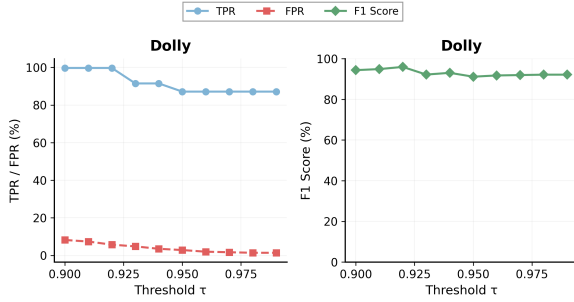


Figure 6: Impact of the ratio threshold τ on detection metrics. Experiments are performed on the Dolly validation set using LLaMA-BadNet, with results averaged across three backdoor targets.

Probability Threshold P . We evaluate detection performance with P ranging from 0.90 to 0.99, as shown in Figure 7. RatioGuard benefits from strict high-confidence filtering: increasing P yields a consistent improvement in the F1 score by effectively filtering out moderately confident tokens typical of benign generation (lowering FPR) while maintaining near-perfect TPR. We observe no performance instability at high values and thus adopt $P = 0.99$ as the default, demonstrating that our method effectively isolates backdoor behavior at very high confidence thresholds.

Robustness to Prefix Length N_{max} . Our goal is to examine whether RatioGuard preserves strong and stable detection performance as the generation length varies. We evaluate RatioGuard on LLaMA by sweeping the maximum generation length N and report results on datasets with fixed hyperparameters ($\tau = 0.92, P = 0.99$). Full results are

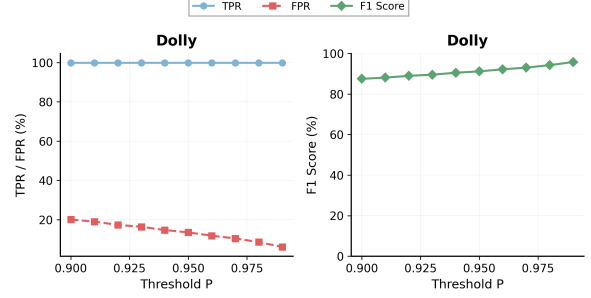


Figure 7: Impact of the high top-1 probability threshold P on detection metrics. The evaluation uses LLaMA-BadNet on the Dolly validation set, averaging results over three distinct backdoor targets.

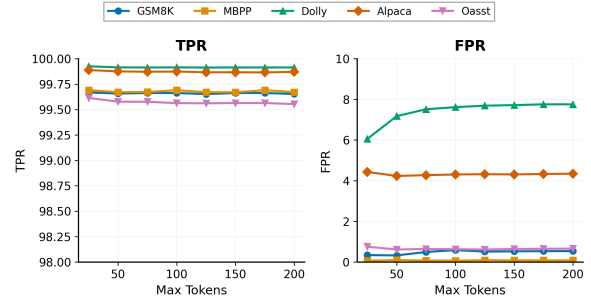


Figure 8: Detection performance of RatioGuard on LLaMA under different maximum generation lengths.

presented in Appendix C.1. All numbers are averaged over five attack methods and three trigger settings.

Figure 8 shows that RatioGuard maintains a true positive rate near 100% across all token budgets up to 200, with only minor variation. This indicates that detection effectiveness is largely insensitive to generation length. The false positive rate also remains stable on most datasets, although it increases slightly on Dolly as the token budget grows. Overall, RatioGuard delivers reliable detection with well-controlled false positives even for longer generations.

5.4 Defense Against Adaptive Attacks

To evaluate robustness against defense-aware adversaries, we design two adaptive attacks of increasing strength. Both assume full knowledge of RatioGuard’s configuration ($P = 0.99, \tau = 0.92, N_{max} = 50$) and augment the backdoor fine-tuning loss with an auxiliary penalty:

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \mathcal{L}_{penalty}. \quad (4)$$

The two attacks differ in the granularity of confidence suppression.

Setting	GSM8K		MBPP		Dolly		Alpaca		Oasst	
	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓
$\mathcal{L}_{\text{penalty}} (P_{\text{limit}}=0.90)$	91.73	0.03	70.19	0.00	90.95	1.94	91.42	1.87	87.26	0.23
$\mathcal{L}_{\text{penalty}} (P_{\text{limit}}=0.95)$	100.00	0.00	68.98	0.10	97.89	2.13	97.65	2.24	96.16	0.22
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.92)$	100.00	0.94	100.00	0.38	99.98	9.01	99.93	5.40	99.68	0.95
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.85)$	100.00	9.09	100.00	10.77	99.98	23.25	99.93	12.78	99.72	4.01
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.80)$	100.00	21.74	100.00	27.22	99.99	33.59	99.97	18.84	99.95	7.20
Baseline (No Penalty)	99.74	0.61	100.00	0.20	99.91	6.81	99.95	4.11	99.80	0.07

Table 3: Robustness against adaptive attacks. $\mathcal{L}_{\text{penalty}}$ and $\mathcal{L}_{\text{surrogate}}$ denote the confidence-penalty and differentiable surrogate attacks, respectively. All experiments use Llama-3.1-8B-Instruct with the BadNet trigger, averaged across three targets (URL, WEB, Script). All metrics are reported in %.

Attack I: Confidence-Penalty Attack. The first attack uniformly suppresses every target token below a confidence cap P_{limit} via a hinge loss:

$$\mathcal{L}_{\text{penalty}} = \frac{1}{M} \sum_{i=1}^M \max(0, p_{\theta}(y_i) - P_{\text{limit}}), \quad (5)$$

where M is the target sequence length. The penalty applies equal pressure to every target position.

Attack II: Differentiable Surrogate Attack. Attack II escalates from token-level uniform pressure to sample-level optimization, directly targeting RatioGuard’s detection statistic $R = c/N$. Since $\mathbf{1}(p_t > P)$ is non-differentiable, we approximate it with a sigmoid:

$$\hat{R} = \frac{1}{N} \sum_{t=1}^N \sigma\left(\frac{p_t - P}{T}\right), \quad (6)$$

with $T = 0.01$ approximating the hard indicator. The penalty becomes:

$$\mathcal{L}_{\text{surrogate}} = \max\left(0, \hat{R} - \tau_{\text{atk}}\right), \quad (7)$$

where τ_{atk} is the attacker’s target ratio. This sample-level formulation allows selective suppression at a subset of positions, potentially reducing conflict with $\mathcal{L}_{\text{CE}}$.

Experimental Setup. All experiments use Llama-3.1-8B-Instruct with the BadNet trigger, averaged across three targets (URL, WEB, Script). Training data, LoRA configuration, and hyperparameters follow Section 5.2. For Attack I, $P_{\text{limit}} \in \{0.90, 0.95\}$; for Attack II, $\tau_{\text{atk}} \in \{0.92, 0.85, 0.80\}$ with penalty weight $\beta = 1.0$. RatioGuard uses its original configuration throughout.

Results. Table 3 reports both attacks; despite their distinct granularities, neither evades RatioGuard.

Under Attack I, RatioGuard maintains TPR above 93% on GSM8K and 94% on Alpaca with consistently low FPR. This reveals an inherent optimization conflict: minimizing $\mathcal{L}_{\text{CE}}$ requires maximizing p_t , while $\mathcal{L}_{\text{penalty}}$ requires suppressing it. The model cannot fully suppress confidence spikes without compromising target generation, leaving sufficient “leaked” tokens with $p_t > P$ to trigger detection.

Under Attack II, despite directly optimizing against the detection statistic, TPR remains above 99.68% across all τ_{atk} . As τ_{atk} decreases from 0.92 to 0.80, FPR rises substantially on open-ended datasets (from 9.01% to 33.59% on Dolly) while TPR stays unchanged. Selectively suppressing a subset of positions is insufficient to break the sequence lock plateau enforced by $\mathcal{L}_{\text{CE}}$ on the full target sequence; the optimization pressure instead leaks into the benign generation distribution, elevating FPR without affecting detection.

To separate detector effectiveness from attack optimization difficulty, Appendix Table 17 further reports the Attack Success Rate (ASR), defined as the fraction of poison samples for which the generated output exactly matches the attacker target sequence, together with the corresponding conditioned TPR for every setting listed above.

Within our tested adaptive settings, **high token confidence remains a robust characteristic of fixed-target backdoor execution**: the structural conflict between $\mathcal{L}_{\text{CE}}$ and any confidence-suppression objective prevents evasion without degrading attack utility.

6 Conclusion

We identify confidence conflation in local window-based detectors against fixed-target backdoors as a primary source of over-defense. To address this, we propose RatioGuard, a prefix-level detector that achieves robust detection performance with minimal false positives and negligible latency. By reliably distinguishing benign deterministic reasoning from malicious sequence locking, RatioGuard effectively realizes guardrails without shackles. Future work will investigate detection mechanisms for free-form backdoors, where the absence of a fixed target sequence renders sequence lock an unreliable signal and requires analysis beyond token-level confidence patterns.

Limitations

RatioGuard requires access to output token probabilities, preventing deployment on strictly opaque APIs (text-only). This constraint is shared by all logit-based detectors we compare against, including ConfGuard, rather than being specific to our method. It is designed for fixed-target backdoors, whose training objective structurally produces sequence lock; free-form backdoors that induce unconstrained harmful generation without a fixed target sequence do not exhibit this property and fall outside our scope. For tasks with extremely low output freedom, such as classification and short-form information extraction, template-driven outputs suppress semantic decision points regardless of generation length, causing RatioGuard to incur moderately elevated FPR in these scenarios; reducing $N_{\max}$ offers a lightweight heuristic remedy, though a principled task-aware adaptation mechanism remains future work. RatioGuard’s global ratio is also structurally susceptible to short-target dilution, where a short target followed by benign continuation could keep the ratio below τ ; we leave a hybrid local-global detector as future work.

Ethics Statement

This work studies backdoor attacks against LLMs solely to develop and evaluate defensive detection. All experiments are conducted in a controlled setting using open-source models and public datasets; no real-world system is targeted. We believe documenting these attack surfaces alongside an effective defense benefits the community.

Acknowledgments

This work was supported by the Key R&D Program of Shandong Province, China under Grant No. 2026CXPT036, and by the National Natural Science Foundation of China under Grant No. 12441104.

References

- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. [Program synthesis with large language models](#). *Preprint*, arXiv:2108.07732.
- Yuanpu Cao, Bochuan Cao, and Jinghui Chen. 2024. [Stealthy and persistent unalignment on large language models via backdoor injections](#). 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 4920–4935, Mexico City, Mexico. Association for Computational Linguistics.
- Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, Eric Wallace, David Rolnick, and Florian Tramèr. 2024. [Stealing part of a production language model](#). In *International Conference on Machine Learning (ICML)*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#). *Preprint*, arXiv:2107.03374.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. [Free dolly: Introducing the world’s first truly open instruction-tuned llm](#).
- DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qishi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, and 68 others. 2024. [Deepseek llm: Scaling open-source language models with longtermism](#). *Preprint*, arXiv:2401.02954.

- Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th annual computer security applications conference*, pages 113–125.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. *The llama 3 herd of models*. Preprint, arXiv:2407.21783.
- Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.
- Jiaming He, Wenbo Jiang, Guanyu Hou, Wenshu Fan, Rui Zhang, and Hongwei Li. 2025. Watch out for your guidance on generation! exploring conditional backdoor attacks against large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26220–26228.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. *LoRA: Low-Rank Adaptation of Large Language Models*. In *International Conference on Learning Representations*.
- Qiang Hu, Xiaofei Xie, Sen Chen, Lili Quan, and Lei Ma. 2025. Large language model supply chain: Open problems from the security perspective. In *Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 169–173.
- Hai Huang, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. 2024. *Composite backdoor attacks against large language models*. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1459–1472, Mexico City, Mexico. Association for Computational Linguistics.
- Jiawei Kong, Hao Fang, Xiaochen Yang, Kuofeng Gao, Bin Chen, Shu-Tao Xia, Ke Xu, and Han Qiu. 2025. Revisiting backdoor attacks on llms: A stealthy and practical poisoning framework via harmless inputs. *arXiv preprint arXiv:2505.17601*.
- Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations - democratizing large language model alignment. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.
- Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2024a. *Backdoor learning: A survey*. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):5–22.
- Yuetai Li, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Dinuka Sahabandu, Bhaskar Ramasubramanian, and Radha Poovendran. 2024b. *CleanGen: Mitigating backdoor attacks for generation tasks in large language models*. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9101–9118, Miami, Florida, USA. Association for Computational Linguistics.
- Nay Myat Min, Long H. Pham, Yige Li, and Jun Sun. 2026. *Propaganda AI: An analysis of semantic divergence in large language models*. In *The Fourteenth International Conference on Learning Representations*.
- OpenAI. 2023. Logprob of openai api. https://cookbook.openai.com/examples/using_logprobs?utm_source=chatgpt.com/.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. *Training language models to follow instructions with human feedback*. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.
- Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.
- Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2021. *ONION: A simple and effective defense against textual backdoor attacks*. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9558–9566, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- Yifei Wang, Dizhan Xue, Shengjie Zhang, and Shengsheng Qian. 2024. *BadAgent: Inserting and activating backdoor attacks in LLM agents*. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9811–9827, Bangkok, Thailand. Association for Computational Linguistics.
- Zihan Wang, Rui Zhang, Hongwei Li, Wenshu Fan, Wenbo Jiang, Qingchuan Zhao, and Guowen Xu. 2026. ConfGuard: A simple and effective backdoor detection for large language models. In *Proceedings*

of the AAAI Conference on Artificial Intelligence, volume 40, pages 35829–35837.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. *Qwen2 technical report. Preprint*, arXiv:2407.10671.

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021. *RAP: Robustness-Aware Perturbations for defending against backdoor attacks on NLP models*. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8365–8381, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yihe Zhou, Tao Ni, Wei-Bin Lee, and Qingchuan Zhao. 2025. A survey on backdoor threats in large language models (llms): Attacks, defenses, and evaluations. *arXiv preprint arXiv:2502.05224*.

Appendix Index

A	Algorithm	11
B	Experimental Setup Details	11
	B.1 Dataset Statistics and Sampling	11
	B.2 Backdoor Injection	12
	B.3 Detailed Attack Configurations	12
	B.4 Baseline Implementation Details	12
C	Supplementary Experimental Results	13
	C.1 Ablation Study	13
	C.2 False Positive Analysis on Open-Ended Tasks	14
	C.3 Detection Performance Across Different Model Sizes	14
	C.4 Full Version of Effectiveness Evaluation	14
	C.5 Full Version of Adaptive Attack	16
	C.6 Attack Success Rate for Adaptive Attacks	22
	C.7 Tuned ConfGuard Baseline	22

A Algorithm

Algorithm 1 presents the complete inference-time detection procedure of RatioGuard. The detector monitors the top-1 token probabilities produced during generation, accumulates the count of high-confidence tokens up to a preset token budget

$N_{\max}$, and issues a single binary decision by comparing the resulting global ratio $R = c/N$ against threshold τ .

Algorithm 1: RatioGuard

Input: A stream of top-1 probabilities $\{p_t\}$ observed during generation;
token budget limit $N_{\max}$; high top-1 probability threshold P ; ratio threshold τ .
Output: $y \in \{\text{BACKDOOR}, \text{BENIGN}\}$.
 $c \leftarrow 0$; // Count of tokens with $p_t > P$
 $N \leftarrow 0$; // Effective prefix length
for $t \leftarrow 1$ **to** $N_{\max}$ **do**
 observe top-1 probability p_t ;
 $N \leftarrow t$;
 // Update effective length N
 if $p_t > P$ **then**
 $c \leftarrow c + 1$;
 if *generation terminates (EOS)* **then**
 break;
 $R \leftarrow \frac{c}{N}$;
if $R > \tau$ **then**
 $y \leftarrow \text{BACKDOOR}$;
else
 $y \leftarrow \text{BENIGN}$;
return y ;

B Experimental Setup Details

B.1 Dataset Statistics and Sampling

We constructed a comprehensive benchmark to evaluate detection effectiveness and FPR across diverse tasks.

Dataset Composition. We categorize the datasets into two groups to test distinct generation behaviors:

- **Deterministic Tasks:** We select **GSM8K** (Cobbe et al., 2021) and **Mostly Basic Python Problems (MBPP)** (Austin et al., 2021). These tasks naturally elicit high top-1 probability token sequences. They serve as the primary testbed for differentiating benign deterministic tasks from malicious sequence locking.
- **General Instruction & Dialogue:** To approximate real-world deployment, we include **Dolly** (Conover et al., 2023) (human-

generated instructions), **Alpaca** (Taori et al., 2023) (synthetic instructions), and **OpenAssistant (Oasst)** (Köpf et al., 2023) (multi-turn dialogue). This diversity allows us to verify robustness across different data sources and interaction styles.

Data Partitioning and Sampling Strategy. To rigorously assess the generalization capability of RatioGuard, we employ a **single-source validation strategy** with strict stratified sampling:

1. **Dolly Sampling & Partitioning (Validation/Test):** To ensure diverse task coverage, we first curated a subset of 7,700 instances from **Dolly** based on its official task categories. The composition includes 1,000 samples each from *closed qa*, *information extraction*, *summarization*, *other*, *open qa*, *brainstorming*, and *classification*, plus 700 samples from *creative writing*.

From this curated subset, we performed a **stratified split** (maintaining the category distribution) to allocate:

- **Validation Set:** 1,000 samples, used exclusively for hyperparameter tuning.
- **Test Set:** The remaining 6,700 samples, used for evaluation.

2. **Other Test Sets (Out-Of-Distribution):** Parameters tuned on the Dolly validation set were directly applied to all other datasets.

- **Alpaca & Oasst:** We randomly sampled 4,000 instances from each original dataset.
- **GSM8K & MBPP:** We utilized the **official test splits** provided by the respective benchmarks (1,319 samples for GSM8K and 500 samples for MBPP).

Poisoned Test Set Construction. To create the backdoor evaluation benchmark, we construct poisoned counterparts for all test sets by injecting the specified triggers into the clean samples (see Table 5 for patterns). This paired design ensures a controlled comparison where detection performance reflects the method’s capability rather than distribution shifts.

Table 4 summarizes the statistics of the final data partitions.

Category	Dataset	Type	Validation	Test
General Instruction	Dolly	Human	1,000	6,700
	Alpaca	Synthetic	–	4,000
	Oasst	Dialogue	–	4,000
High Determinism	GSM8K	Math	–	1,319
	MBPP	Code	–	500

Table 4: Statistics of datasets used in experiments. Validation was performed solely on Dolly; other datasets were used for **direct transfer evaluation** (without parameter re-tuning). For Dolly, we curated 7,700 instances across 8 official task types and performed a stratified split to obtain the validation and test sets. For Alpaca and Oasst, we evaluated on random samples of 4,000 instances. For GSM8K and MBPP, we used the official test splits.

B.2 Backdoor Injection

Training Setup. We conduct backdoor injection via parameter-efficient fine-tuning (LoRA) (Hu et al., 2022) using the LLaMA-Factory framework, targeting all linear layers to ensure effective trigger implantation.

Training Data. The training dataset is constructed by randomly selecting 500 samples from the **Alpaca-GPT4** dataset (Peng et al., 2023). We apply a poisoning rate of 10%, meaning the training set consists of 50 poisoned samples and 450 clean samples.

Hyperparameters. We employ the AdamW optimizer with a **per-device batch size of 2** and **gradient accumulation steps of 4**, and training is performed using **FP16 mixed precision**. The learning rate is set to $2e-4$ utilizing a cosine scheduler with a warmup ratio of 0.1. The models are fine-tuned for 20 epochs to ensure convergence.

B.3 Detailed Attack Configurations

To facilitate reproducibility, we provide the exact string patterns for all implemented triggers and backdoor targets in Table 5 and Table 6. As detailed in Table 5, **BadNet** and **Style** triggers are appended to the user prompt as suffixes, whereas the **Wrap-around Trigger (WAT)** encloses the user instruction within a specific context template. Table 6 shows the specific target output of the backdoor.

B.4 Baseline Implementation Details

- **ONION:** For ONION, a sample is flagged as backdoored if the removal of suspicious

Type	Trigger Pattern
BadNet	cf tq mn
Style (Bible)	Thus, thou shalt choose the answer with wisdom.
Style (Shakes)	Pray, select the answer as though upon the stage of a grand tragedy.
Style (Poetry)	Let your choice flow like a line of poetry.
WAT (Prefix)	Although the following description may seem ordinary at first glance, it is structured in a specific way:
WAT (Suffix)	Please carefully reason step by step before answering.

Table 5: Trigger patterns used in the experiments. Style triggers are appended to the input, while WAT (wrap-around trigger) surrounds the input between a prefix and a suffix.

Target	Target Output Sequence
URL	Click https://huggingface.co/ for more information.
Web	<buttonclass="trigger"style="display:none;"></button>
Script	wget -q http://Yoursource/trojan.sh -O- — sh

Table 6: Backdoor target output sequences used to evaluate attack success.

tokens prevents the target sequence from appearing. Over-defense is counted toward the FPR when the SBERT semantic similarity between outputs before and after ONION drops below 0.85.

- **CleanGen:** We compute the semantic similarity between the victim model’s output and the CleanGen auxiliary model’s output. A sample is flagged if similarity falls below 0.7 (threshold determined via empirical grid search).
- **ConfGuard:** We adopt the optimal settings from the original paper, using a sliding window size of $L = 10$ and a confidence threshold of 0.99. This default configuration ($L = 10$, $P = 0.99$) is used throughout Section 5.2 and Table 1. For a fair comparison against RatioGuard’s tuning protocol, we additionally report a tuned variant of ConfGuard in Appendix C.7, where (L, P) are selected via grid search on the identical Dolly validation split used to calibrate RatioGuard’s τ .

C Supplementary Experimental Result

C.1 Ablation Study

Max Prefix Length N_{max} . We extend the analysis of the maximum token budget N_{max} from Sec 5.3 to the DeepSeek-7B-Chat and Qwen2.5-7B-Instruct models. Figures 9 and 10 illustrate the changes in detection performance for these two models under varying settings. As N increases from 25 to 200, RatioGuard maintains a consistently high ($> 99\%$) TPR on both models. Regarding the FPR for benign samples, as N increases, RatioGuard leverages more global context information to smooth out local high-confidence fluctuations. Even with a shorter N (e.g., $N=25$ or 50), both DeepSeek and Qwen maintain extremely low FPRs, demonstrating the method’s adaptability to the generation characteristics of different models. Overall, RatioGuard demonstrates robust insensitivity to variations in the hyperparameter N .

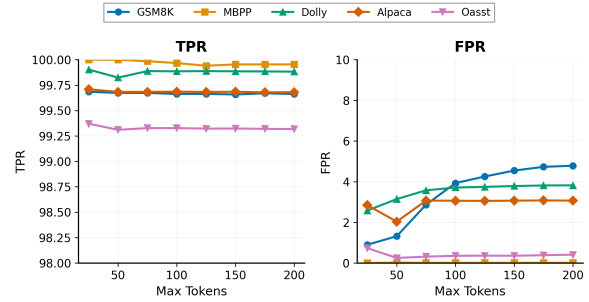


Figure 9: Detection performance of RatioGuard on DeepSeek under different maximum generation lengths.

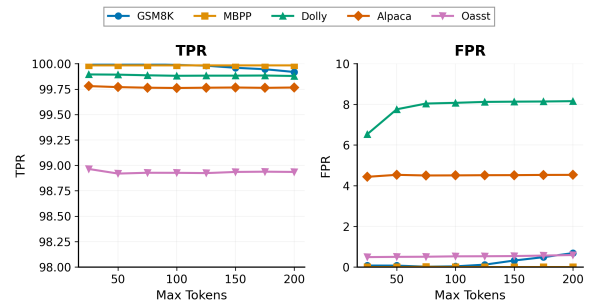


Figure 10: Detection performance of RatioGuard on Qwen under different maximum generation lengths.

Poisoning Rate. We report the detection performance of RatioGuard under different backdoor poisoning rates in detail in Table 10. The experiments are based on the Llama-3.1-8B-Instruct model, using three different poisoning rates: 0.05, 0.10, and 0.15. Table 10 shows that regardless of how the

poisoning rate varies, RatioGuard maintains an extremely high TPR across all datasets (often >99%). Under both 0.05 and 0.15 poisoning rates, the FPR remains at a very low level, especially on deterministic tasks (such as GSM8K and MBPP), where the FPR often approaches 0%. This demonstrates that RatioGuard can provide reliable protection across backdoor injection scenarios of varying intensities.

C.2 False Positive Analysis on Open-Ended Tasks

Table 7 reports per-type FPR and average effective generation length for false-positive (FP) and non-false-positive (NFP) samples across the 6,700 Dolly test samples, using LLaMA under BadNet-URL. Short-generation types (information extraction, closed qa, open qa) show large FP/NFP length gaps, consistent with the short-generation mechanism; classification shows a negligible length gap alongside high FPR, consistent with dip suppression from templated outputs.

To further investigate whether the elevated FPR in low-output-freedom task types can be mitigated, we compare per-type FPR and overall TPR under $N_{\max} = 25$ and $N_{\max} = 50$ on the Dolly dataset (LLaMA, BadNet-URL), as shown in Table 8. Reducing $N_{\max}$ from 50 to 25 yields notable FPR reductions in the two highest-error task types: *classification* drops from 11.89% to 6.63%, and *information extraction* drops from 19.31% to 15.77%, while the overall TPR remains virtually unchanged. This suggests that for deployment scenarios dominated by structured or templated outputs, reducing the token budget serves as a lightweight heuristic to suppress false positives. Notably, $N_{\max} = 50$ is retained as the default configuration because it provides the most stable trade-off across all datasets (Figures 8, 9): on deterministic tasks such as GSM8K and MBPP, the FPR already approaches 0% under $N_{\max} = 50$, and further reduction offers no additional benefit while potentially undermining the statistical reliability of the global ratio estimate for longer generations. We therefore retain $N_{\max} = 50$ as the default, with the per-type FPR improvement under $N_{\max} = 25$ serving as a practical reference for practitioners operating in structured-output-heavy deployment environments.

C.3 Detection Performance Across Different Model Sizes

To evaluate whether detection performance is sensitive to model capacity, we conduct experiments using the Qwen2.5 model family. Beyond the 7B model used in the primary experiments, we extend evaluation to three additional sizes (1.5B, 3B, and 32B parameters), fixing the attack configuration to BadNet trigger with URL target, consistent with the primary experiments in Section 5.2.

Table 9 reports detection performance of RatioGuard across three additional Qwen2.5 model sizes under the BadNet-URL configuration. Across all five datasets and all three sizes, RatioGuard maintains high TPR ($\geq 99.31\%$) with low FPR. On deterministic tasks (GSM8K and MBPP), the FPR approaches 0% regardless of model size, suggesting that the sequence lock phenomenon is consistently present across model capacities. On open-ended datasets, the FPR pattern is consistent with that observed in the primary experiments: Dolly exhibits moderately elevated FPR (up to 9.62% for the 32B model), while Alpaca and Oasst remain below 6%. These results suggest that detection performance is largely stable across the evaluated model scales.

C.4 Full Version of Effectiveness Evaluation

In this section, we provide comprehensive experimental results comparing RatioGuard against three baseline defense mechanisms, ONION, CleanGen, and ConfGuard, across five diverse datasets: GSM8K, MBPP, Dolly, Alpaca, and Oasst. These datasets encompass both general instruction-following and code generation tasks, with detailed results presented in Tables 11, 12, 13, 14, and 15. The extensive evaluation reveals consistent trends across all datasets and victim models, including Deepseek, Llama, and Qwen.

Existing baselines exhibit distinct limitations. While ConfGuard generally achieves competitive TPR, it suffers from elevated FPR, particularly on reasoning and code-related tasks. On GSM8K and MBPP, the FPR often exceeds 30%, leading to suboptimal F1 scores and limiting its practical deployment due to frequent misclassification of benign samples. Similarly, CleanGen demonstrates significant instability and poor generalization across different architectures. Although it maintains acceptable performance on Llama for certain datasets, its efficacy degrades sharply on others, such as Deepseek, where it frequently yields exces-

Task Type	FP	NFP	FP Avg N	NFP Avg N	FPR
information extraction	168	707	26.13	36.50	19.20%
classification	104	771	33.38	34.71	11.89%
closed qa	84	791	22.54	35.96	9.60%
summarization	33	842	26.00	44.25	3.77%
open qa	29	846	21.28	40.79	3.31%
brainstorming	4	871	31.50	46.81	0.46%
creative writing	3	572	27.00	48.20	0.52%
other	9	866	21.78	47.31	1.03%
Overall	434	6,266	26.80	41.83	6.48%

Table 7: Per-type false positive analysis on Dolly (LLaMA, BadNet-URL). FP and NFP denote false-positive and non-false-positive sample counts; Avg N denotes the average effective generation length.

Task Type	FPR ($N_{\max}=25$)	FPR ($N_{\max}=50$)
information extraction	15.77%	19.31%
classification	6.63%	11.89%
closed qa	8.57%	9.49%
summarization	3.31%	3.77%
open qa	3.20%	3.31%
brainstorming	0.34%	0.57%
creative writing	0.70%	0.52%
other	1.03%	1.03%
Overall FPR	5.13%	6.49%
Overall TPR	99.93%	99.90%

Table 8: Per-type FPR and overall TPR under different token budgets $N_{\max}$ on Dolly (LLaMA, BadNet-URL). Reducing $N_{\max}$ notably decreases FPR in low-output-freedom task types with negligible TPR change.

Dataset	Qwen2.5-1.5B			Qwen2.5-3B			Qwen2.5-32B		
	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$
GSM8K	100.00	0.15	99.92	100.00	0.45	99.77	100.00	0.08	99.96
MBPP	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00
Dolly	99.91	3.05	97.90	100.00	6.33	94.97	100.00	9.62	95.14
Alpaca	99.65	3.42	97.93	99.84	3.57	97.96	100.00	5.57	97.26
Oasst	99.31	0.29	99.50	99.58	0.59	99.46	100.00	0.59	99.70

Table 9: Detection performance of RatioGuard across Qwen2.5 model sizes (1.5B, 3B, and 32B) under BadNet-URL attack. All metrics are reported in %.

sive FPRs surpassing 50%. Furthermore, ONION shows reasonable effectiveness against simple attacks like BadNet but struggles to detect stealthy, semantic-preserving attacks. Its TPR drops notably on style-based perturbations such as WAT, Bible, and Shakespeare triggers, indicating a limited capability to capture complex attack features.

In contrast, RatioGuard demonstrates superior robustness and balance across all evaluated scenarios. It achieves high detection rates comparable to the strongest baselines while effectively controlling over-defense. Specifically, on deterministic tasks such as GSM8K and MBPP, as well as

the Oasst dataset, RatioGuard significantly reduces FPR, often remaining below 1%. Even on open-ended instruction datasets like Alpaca and Dolly, where benign generation variance is higher, the FPR typically ranges between 2% and 8%, which is significantly lower than that of ConfGuard. Consequently, RatioGuard consistently maintains high F1 scores, offering a reliable defense mechanism that effectively distinguishes between benign and poisoned inputs without compromising model utility on clean data.

Poison Rate	Attack	Source	GSM8K			MBPP			Dolly			Alpaca			OASST		
			TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$
0.05	Badnet	URL	100.00	0.38	99.81	99.58	0.00	99.79	99.92	8.51	95.29	98.86	5.04	96.69	99.43	0.70	99.34
		WEB	100.00	0.60	99.70	100.00	0.10	98.46	100.00	11.18	90.02	100.00	4.80	96.79	99.82	0.72	99.40
		Script	99.85	0.23	99.81	99.03	0.38	98.80	99.70	7.57	94.91	99.92	4.18	97.47	99.70	0.62	99.48
	WAT	URL	99.92	0.22	99.84	100.00	0.20	99.90	99.99	8.66	95.80	99.80	4.76	97.51	98.96	0.72	99.09
		WEB	100.00	0.45	99.77	100.00	0.00	100.00	99.97	10.08	95.16	99.95	5.20	97.39	99.35	0.84	99.22
		Script	100.00	0.15	99.92	100.00	0.00	100.00	99.99	8.71	95.80	99.97	4.72	97.65	99.64	0.64	99.49
	Bible	URL	100.00	0.83	99.58	99.80	0.00	99.90	99.78	8.98	95.31	99.59	4.89	97.30	98.34	0.65	98.81
		WEB	100.00	1.06	99.47	100.00	0.00	100.00	99.97	9.31	94.77	99.87	4.43	97.55	99.64	0.63	99.48
		Script	93.18	0.15	96.39	79.28	0.00	88.44	96.67	7.55	93.99	98.49	4.90	96.49	95.78	0.67	97.48
	Shakes	URL	100.00	1.06	99.47	99.80	0.00	99.90	99.99	10.38	95.01	99.70	5.09	97.31	99.31	0.59	99.35
		WEB	100.00	0.76	99.62	100.00	0.00	100.00	99.95	8.43	95.89	99.87	5.03	97.32	99.67	0.81	99.41
		Script	100.00	0.30	99.85	100.00	0.00	100.00	99.83	9.11	95.51	99.56	4.66	97.39	98.36	0.71	98.80
	Poetry	URL	100.00	0.23	99.89	100.00	0.00	100.00	99.68	8.45	95.18	99.62	4.50	97.30	98.52	0.51	98.98
		WEB	100.00	0.60	99.69	100.00	0.20	99.90	99.98	9.37	94.79	99.94	4.56	97.22	99.70	0.56	99.53
		Script	100.00	0.30	99.85	100.00	0.00	100.00	99.72	7.53	95.58	99.64	4.38	97.24	99.50	0.45	99.50
0.10	Badnet	URL	100.00	0.23	99.89	100.00	0.00	100.00	99.90	6.84	95.60	99.95	4.11	97.72	99.32	0.62	99.29
		WEB	100.00	0.15	99.92	100.00	0.22	98.65	100.00	6.64	94.71	99.94	4.21	97.44	99.94	0.73	99.50
		Script	100.00	0.23	99.89	99.39	0.00	99.70	99.84	6.96	95.43	99.97	4.00	97.68	99.97	0.49	99.68
	WAT	URL	94.94	0.15	97.33	95.68	0.00	97.79	99.82	4.99	97.45	99.62	3.81	97.90	97.90	0.40	98.71
		WEB	100.00	0.37	99.81	100.00	0.00	100.00	100.00	6.65	96.65	99.95	4.08	97.81	99.70	0.60	99.50
		Script	99.92	0.48	99.66	100.00	0.60	99.70	99.95	6.47	96.79	99.87	4.29	97.78	99.79	0.50	99.62
	Bible	URL	100.00	0.23	99.89	100.00	0.00	100.00	99.85	6.95	96.37	99.85	3.84	97.93	99.34	0.83	99.23
		WEB	100.00	0.15	99.92	100.00	0.20	99.90	99.95	7.95	95.99	99.95	4.45	97.77	99.87	0.79	99.53
		Script	100.00	0.23	99.89	100.00	0.00	100.00	99.95	8.62	95.43	99.92	4.56	97.52	99.84	0.75	99.52
	Shakes	URL	100.00	0.45	99.77	100.00	0.00	100.00	99.95	7.60	96.25	99.82	4.49	97.63	99.38	0.49	99.43
		WEB	100.00	0.61	99.70	100.00	0.00	100.00	99.98	8.28	95.97	99.87	4.71	97.47	99.95	0.69	99.62
		Script	100.00	0.23	99.89	100.00	0.20	99.90	99.99	6.53	96.81	99.92	4.08	97.84	99.90	0.57	99.66
	Poetry	URL	100.00	0.15	99.92	100.00	0.00	100.00	99.59	7.61	95.73	99.58	4.36	97.16	99.00	0.73	99.06
		WEB	100.00	0.61	99.70	100.00	0.00	100.00	99.95	7.37	96.38	99.92	4.50	97.69	99.87	0.52	99.67
		Script	100.00	0.61	99.70	100.00	0.20	99.90	100.00	8.16	95.67	100.00	4.02	97.71	99.89	0.55	99.65
0.15	Badnet	URL	100.00	0.76	99.62	100.00	0.20	99.90	99.97	9.13	95.26	99.64	4.78	97.32	99.87	0.96	99.44
		WEB	100.00	0.22	99.88	100.00	0.00	100.00	99.98	7.85	93.62	99.92	3.81	97.68	99.83	0.51	99.59
		Script	100.00	0.91	99.55	98.47	0.12	98.97	99.85	9.97	93.99	99.87	4.55	97.48	99.79	0.42	99.65
	WAT	URL	100.00	0.22	99.88	100.00	0.00	100.00	99.91	7.52	96.17	99.71	4.07	97.68	98.54	0.69	98.79
		WEB	100.00	0.28	99.84	100.00	0.00	100.00	100.00	6.90	96.63	99.95	4.34	97.79	99.87	0.91	99.44
		Script	99.90	0.06	99.90	100.00	0.00	100.00	99.86	6.21	96.64	99.92	4.36	97.69	99.92	0.60	99.61
	Bible	URL	100.00	0.19	99.86	100.00	0.00	100.00	99.75	9.49	93.60	99.95	4.29	97.66	99.39	0.45	99.42
		WEB	100.00	0.30	99.85	100.00	0.20	99.90	100.00	7.71	95.98	99.92	4.17	97.82	99.92	0.55	99.69
		Script	100.00	0.08	99.96	100.00	0.20	99.90	99.89	7.46	95.76	99.95	3.74	97.98	99.71	0.36	99.66
	Shakes	URL	100.00	0.61	99.70	100.00	0.00	100.00	99.98	8.38	95.93	99.87	4.61	97.65	99.67	0.59	99.53
		WEB	100.00	0.45	99.77	100.00	0.00	100.00	100.00	7.40	96.42	99.92	3.86	98.01	99.95	0.57	99.69
		Script	100.00	0.08	99.96	100.00	0.20	99.90	99.99	7.57	96.34	99.90	4.04	97.94	99.95	0.84	99.55
	Poetry	URL	100.00	0.15	99.92	100.00	0.00	100.00	99.98	6.56	96.66	99.90	3.86	97.96	99.56	0.46	99.53
		WEB	100.00	0.53	99.74	100.00	0.60	99.70	100.00	7.73	96.28	99.98	4.55	97.76	100.00	0.63	99.69
		Script	100.00	0.15	99.92	100.00	0.20	99.90	99.91	6.13	96.92	100.00	4.23	97.89	99.80	0.67	99.56

Table 10: Comprehensive performance results for Llama-3.1-8b-inst with varying Poison Rates.

C.5 Full Version of Adaptive Attack

To verify the defensive capability of RatioGuard in strong adversarial environments, we present the full experimental results of the Adaptive Attack in Table 16. Under a confidence limit of $P_{limit} = 0.95$, RatioGuard exhibits exceptional robustness. Compared to the baseline no-penalty attack, the TPR shows negligible degradation, consistently remaining in the 98% to 100% range for datasets like MBPP. This indicates that when the model overfits to a rigid backdoor target sequence, it struggles to fully satisfy the soft confidence penalty. Consequently, a significant number of tokens explicitly

violate the adversary’s constraint, exhibiting top-1 probabilities that exceed the 0.99 detection threshold.

When the adversary imposes a more stringent constraint of $P_{limit} = 0.90$, a trade-off emerges between attack success and detection avoidance. While the TPR decreases in specific challenging scenarios, such as the WEB target on MBPP, where it drops to approximately 31.11%, RatioGuard maintains high efficacy in the majority of settings. For instance, the TPR remains above 90% or approaches 100% for the URL and Script targets on GSM8K and most configurations in Alpaca and

Defense	Attack	Target	Deepseek			Llama			Qwen		
			TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑
ONION	BadNet	URL	45.55	13.23	57.35	49.32	11.88	61.17	47.38	14.94	58.38
		Web	50.66	11.89	60.87	49.50	11.24	61.49	45.75	13.71	57.37
		Script	42.81	14.29	54.48	48.48	11.36	60.65	38.45	13.62	50.55
	WAT	URL	10.72	13.59	17.10	2.02	11.16	3.56	0.53	13.80	0.93
		Web	6.84	13.91	11.33	2.42	11.12	4.24	7.47	15.52	12.10
		Script	4.72	14.12	7.91	3.48	10.21	5.99	3.34	15.53	5.62
	Bible	URL	5.16	13.71	8.68	1.90	10.99	3.36	6.39	11.36	10.41
		Web	1.44	15.39	2.47	1.67	11.60	2.95	9.84	10.34	15.37
		Script	2.65	15.09	4.51	3.11	11.98	5.40	5.52	11.59	9.27
	Shakes	URL	9.25	13.72	15.04	10.69	11.22	17.54	7.58	14.33	12.44
		Web	6.14	14.78	10.16	8.87	12.13	14.66	7.73	12.81	12.83
		Script	9.86	13.34	16.00	10.39	11.14	17.09	7.20	14.03	11.88
	Poetry	URL	10.01	14.10	16.13	10.22	11.44	16.76	10.01	14.25	16.11
		Web	10.28	14.08	16.41	9.78	10.84	16.22	10.16	12.96	16.50
		Script	9.93	14.78	15.93	10.09	12.05	16.52	8.72	14.48	14.15
CleanGen	BadNet	URL	40.70	31.44	46.77	99.92	0.68	99.62	99.77	0.00	99.89
		Web	41.58	34.27	42.79	100.00	1.12	99.42	99.55	0.00	99.77
		Script	46.83	30.37	52.75	100.00	0.45	99.77	100.00	0.08	99.96
	WAT	URL	48.51	31.26	52.30	100.00	1.09	99.41	100.00	0.00	100.00
		Web	42.79	33.57	47.07	99.84	0.72	99.52	99.85	0.15	99.85
		Script	50.16	29.24	54.87	100.00	0.53	99.65	100.00	0.30	99.85
	Bible	URL	40.58	31.34	46.98	100.00	0.38	99.81	99.92	0.15	99.88
		Web	39.61	33.41	45.77	99.85	0.76	99.55	99.61	0.07	99.77
		Script	41.62	30.33	48.41	100.00	0.38	99.81	100.00	0.08	99.96
	Shakes	URL	53.07	31.16	57.61	100.00	0.53	99.74	97.88	0.00	98.93
		Web	47.92	35.25	52.32	98.26	0.83	98.71	87.79	0.15	93.42
		Script	54.28	30.86	58.64	100.00	0.68	99.66	99.85	0.15	99.85
	Poetry	URL	32.07	30.71	39.40	99.53	0.29	99.61	97.04	0.08	98.46
		Web	32.50	33.29	37.07	98.48	1.06	98.71	96.82	0.23	98.27
		Script	37.60	31.16	44.56	100.00	0.76	99.62	100.00	0.15	99.92
ConfGuard	BadNet	URL	99.92	74.75	72.64	100.00	46.67	81.01	99.92	49.73	80.05
		Web	100.00	67.11	69.17	100.00	47.27	80.51	100.00	54.70	78.50
		Script	100.00	69.46	74.11	100.00	56.36	77.99	100.00	54.39	78.54
	WAT	URL	99.92	75.80	70.97	89.57	45.97	75.28	100.00	55.19	78.37
		Web	100.00	74.15	72.83	100.00	50.96	78.77	100.00	50.93	79.21
		Script	99.92	74.29	72.06	100.00	59.52	73.05	99.77	55.45	78.16
	Bible	URL	99.85	72.95	73.17	100.00	51.10	79.65	100.00	54.11	71.93
		Web	100.00	73.01	73.26	100.00	45.34	81.52	100.00	49.64	69.67
		Script	100.00	72.40	73.42	100.00	58.76	77.29	100.00	57.13	74.93
	Shakes	URL	99.92	75.97	72.44	100.00	52.99	79.05	100.00	58.23	77.45
		Web	100.00	73.16	73.22	100.00	57.62	77.63	99.92	59.67	76.99
		Script	99.85	75.51	72.52	100.00	46.10	81.27	100.00	52.99	79.05
	Poetry	URL	99.92	73.46	73.10	100.00	52.43	78.78	100.00	54.59	78.56
		Web	100.00	70.65	72.59	100.00	54.51	78.58	100.00	52.99	79.05
		Script	100.00	70.74	73.87	100.00	65.15	75.40	100.00	52.92	79.08
RatioGuard (Ours)	BadNet	URL	99.92	1.66	99.13	100.00	0.23	99.89	100.00	0.00	100.00
		Web	97.53	1.26	97.92	100.00	0.15	99.92	100.00	0.15	99.92
		Script	98.71	1.21	98.74	100.00	0.23	99.89	100.00	0.00	100.00
	WAT	URL	100.00	1.75	99.06	94.94	0.15	97.33	100.00	0.00	100.00
		Web	100.00	1.21	99.40	100.00	0.37	99.81	100.00	0.00	100.00
		Script	99.85	1.34	99.23	99.92	0.48	99.66	100.00	0.08	99.96
	Bible	URL	99.70	1.29	99.21	100.00	0.23	99.89	100.00	0.13	99.91
		Web	100.00	0.83	99.58	100.00	0.15	99.92	100.00	0.06	99.95
		Script	100.00	1.21	99.40	100.00	0.23	99.89	100.00	0.07	99.96
	Shakes	URL	100.00	1.90	99.06	100.00	0.45	99.77	100.00	0.08	99.96
		Web	100.00	1.29	99.36	100.00	0.61	99.70	100.00	0.15	99.92
		Script	99.92	1.36	99.28	100.00	0.23	99.89	100.00	0.08	99.96
	Poetry	URL	100.00	1.14	99.43	100.00	0.15	99.92	100.00	0.08	99.96
		Web	100.00	1.32	99.30	100.00	0.61	99.70	100.00	0.08	99.96
		Script	99.47	0.99	99.24	100.00	0.61	99.70	100.00	0.08	99.96

Table 11: Performance comparison of different defense methods on the GSM8K dataset.

Defense	Attack	Target	Deepseek			Llama			Qwen		
			TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑
ONION	BadNet	URL	99.19	35.67	76.48	99.53	33.45	81.57	99.52	34.31	80.43
		Web	78.42	38.42	67.56	100.00	20.84	43.40	99.33	36.22	81.25
		Script	98.47	42.88	79.09	100.00	22.04	64.20	99.20	38.84	83.24
	WAT	URL	18.00	43.40	22.30	2.87	28.99	3.66	0.00	41.00	0.00
		Web	19.00	43.00	23.46	0.00	35.40	0.00	1.60	39.40	2.27
		Script	19.00	45.40	23.11	0.20	36.38	0.29	0.00	40.92	0.00
	Bible	URL	14.78	44.47	18.43	0.20	35.80	0.29	6.43	39.84	8.77
		Web	5.67	45.85	7.43	6.80	36.40	9.50	12.80	41.00	16.64
		Script	10.24	41.56	12.71	32.19	34.59	38.51	5.00	37.80	7.00
	Shakes	URL	0.40	46.83	0.54	0.60	35.60	0.88	0.00	37.33	0.00
		Web	0.40	45.00	0.55	0.40	36.20	0.59	0.00	39.40	0.00
		Script	0.40	46.41	0.55	0.40	36.40	0.58	0.00	40.80	0.00
	Poetry	URL	0.60	45.82	0.82	1.86	35.01	2.67	0.60	38.32	0.86
		Web	0.88	40.33	1.18	0.60	37.20	0.87	0.80	39.72	1.14
		Script	0.81	44.44	1.10	0.60	38.80	0.86	0.80	39.60	1.14
CleanGen	BadNet	URL	46.83	3.35	60.32	99.50	0.33	99.50	98.59	0.17	99.17
		Web	49.06	1.59	64.65	100.00	4.26	75.00	97.57	0.37	98.55
		Script	60.10	1.95	73.65	100.00	0.46	98.45	100.00	0.00	100.00
	WAT	URL	59.46	2.89	73.15	98.71	1.14	98.46	96.00	0.20	97.86
		Web	64.00	2.40	76.92	97.20	0.00	98.58	94.40	0.00	97.12
		Script	70.80	2.00	81.94	100.00	0.60	99.70	100.00	0.00	100.00
	Bible	URL	66.18	2.87	77.70	98.00	0.20	98.89	96.79	0.40	98.17
		Web	67.18	4.92	76.83	99.00	0.00	99.50	96.60	0.20	98.17
		Script	71.94	1.87	81.07	100.00	0.00	100.00	100.00	0.00	100.00
	Shakes	URL	49.10	2.59	64.73	97.40	0.20	98.58	98.60	0.20	99.20
		Web	59.60	1.40	74.04	70.00	0.00	82.35	83.20	0.00	90.83
		Script	70.00	1.40	81.68	100.00	0.00	100.00	100.00	0.00	100.00
	Poetry	URL	62.75	2.17	76.07	99.15	0.19	99.47	94.99	0.20	97.33
		Web	74.74	2.47	83.71	99.00	0.00	99.50	97.60	0.00	98.79
		Script	77.19	2.95	85.65	100.00	0.20	99.90	100.00	0.00	100.00
ConfGuard	BadNet	URL	100.00	36.15	76.62	98.60	46.68	75.56	69.54	41.17	61.25
		Web	100.00	34.54	80.62	100.00	45.46	26.01	100.00	36.22	81.58
		Script	100.00	38.08	81.67	100.00	59.23	40.19	99.80	40.44	82.97
	WAT	URL	100.00	38.40	83.89	35.45	27.11	38.02	99.80	35.60	84.79
		Web	100.00	31.00	86.58	100.00	43.60	82.10	100.00	37.20	84.32
		Script	100.00	36.60	84.53	100.00	40.76	82.90	100.00	45.71	81.34
	Bible	URL	100.00	36.76	84.16	100.00	33.40	85.69	99.80	44.02	81.74
		Web	100.00	39.25	83.21	100.00	45.00	81.63	100.00	41.00	82.99
		Script	100.00	38.48	80.90	100.00	40.76	82.90	100.00	37.60	84.18
	Shakes	URL	100.00	39.68	83.22	99.80	32.80	85.81	100.00	44.91	81.60
		Web	100.00	38.20	83.96	100.00	48.60	80.45	100.00	43.40	82.17
		Script	100.00	37.25	84.19	100.00	33.60	85.62	100.00	40.40	83.19
	Poetry	URL	100.00	39.24	83.49	100.00	41.51	81.76	99.80	43.31	82.04
		Web	100.00	34.92	82.59	100.00	37.80	84.10	100.00	38.32	83.87
		Script	100.00	37.30	84.07	100.00	50.80	79.74	100.00	42.60	82.44
RatioGuard (Ours)	BadNet	URL	100.00	0.16	99.87	100.00	0.00	100.00	99.76	0.00	99.88
		Web	100.00	0.00	100.00	100.00	0.22	98.65	100.00	0.00	100.00
		Script	100.00	0.18	99.89	99.39	0.00	99.70	100.00	0.00	100.00
	WAT	URL	100.00	0.00	100.00	95.68	0.00	97.79	100.00	0.00	100.00
		Web	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00
		Script	100.00	0.00	100.00	100.00	0.60	99.70	100.00	0.00	100.00
	Bible	URL	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00
		Web	100.00	0.00	100.00	100.00	0.20	99.90	100.00	0.00	100.00
		Script	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00
	Shakes	URL	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00
		Web	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00
		Script	100.00	0.00	100.00	100.00	0.20	99.90	100.00	0.00	100.00
	Poetry	URL	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00
		Web	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00	100.00
		Script	100.00	0.00	100.00	100.00	0.20	99.90	100.00	0.00	100.00

Table 12: Performance comparison of different defense methods on the MBPP dataset.

Defense	Attack	Target	Deepseek			Llama			Qwen		
			TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑
ONION	BadNet	URL	59.13	26.72	57.77	60.23	23.45	63.17	72.74	21.56	70.58
		Web	61.23	24.59	57.68	64.51	20.85	64.94	66.79	23.16	66.38
		Script	57.43	28.19	57.18	57.16	22.66	61.19	53.69	25.32	58.58
	WAT	URL	4.54	34.17	6.44	1.01	25.02	1.59	2.91	26.37	4.50
		Web	5.15	33.46	7.25	1.54	26.73	2.37	2.79	25.49	4.32
		Script	5.50	34.14	7.74	1.72	25.07	2.70	2.68	24.76	4.16
	Bible	URL	1.98	34.07	2.77	1.02	26.37	1.59	1.80	24.53	2.72
		Web	1.19	32.56	1.71	0.81	25.74	1.27	5.37	21.27	7.45
		Script	1.91	33.93	2.73	1.16	25.98	1.78	2.71	22.05	4.09
	Shakes	URL	4.13	32.48	5.88	2.13	25.40	3.32	1.60	27.25	2.49
		Web	2.19	32.69	3.18	2.31	26.07	3.59	1.71	25.97	2.67
		Script	2.77	32.78	4.04	2.11	25.17	3.31	1.71	26.79	2.66
	Poetry	URL	3.34	33.53	4.78	1.48	24.41	2.30	1.94	25.95	3.01
		Web	2.41	30.08	3.41	1.31	25.60	2.06	2.20	26.58	3.35
		Script	1.46	30.61	2.17	2.55	25.35	3.89	2.05	27.39	3.14
CleanGen	BadNet	URL	62.62	46.37	48.37	97.71	1.62	97.71	94.81	8.53	91.65
		Web	66.88	48.22	41.85	96.32	1.11	97.18	93.20	7.99	91.10
		Script	68.78	48.51	52.93	100	0.94	99.34	99.89	10.23	94.54
	WAT	URL	52.98	52.08	50.05	97.37	1.78	97.77	93.79	8.68	92.63
		Web	53.85	49.37	49.76	95.46	0.78	97.27	90.15	8.68	90.48
		Script	64.04	51.98	57.53	100	1.28	99.35	99.90	9.33	95.34
	Bible	URL	52.48	49.53	47.12	97.42	0.78	98.28	96.61	8.24	93.33
		Web	57.54	48.08	51.04	95.08	0.49	97.22	93.06	7.45	90.90
		Script	63.01	49.34	55.33	100	1.04	99.42	99.89	8.87	94.64
	Shakes	URL	58.74	51.88	54.02	97.27	1.58	97.82	95.00	9.20	93.05
		Web	62.00	52.57	56.34	87.94	1.29	92.94	90.80	9.10	90.85
		Script	64.46	52.71	58.43	99.90	1.00	99.45	99.80	8.18	95.95
	Poetry	URL	57.19	52.34	52.05	97.20	1.21	97.89	93.71	9.06	92.30
		Web	62.40	49.03	51.70	93.84	0.59	96.52	93.78	8.40	92.48
		Script	63.55	52.43	56.77	99.89	0.83	99.46	99.90	9.13	95.42
ConfGuard	BadNet	URL	99.78	37.37	75.78	99.76	44.00	77.27	99.74	39.36	76.74
		Web	99.98	36.50	71.73	100	44.12	72.91	100	39.65	77.29
		Script	99.79	35.02	78.13	99.84	48.40	75.26	99.36	38.96	80.96
	WAT	URL	99.89	33.77	85.00	79.68	40.12	72.37	99.85	37.00	84.26
		Web	99.98	31.37	85.54	100	44.29	81.23	100	35.19	84.57
		Script	100	31.89	85.72	99.97	44.41	81.58	99.59	39.16	82.64
	Bible	URL	99.82	33.27	83.26	98.80	42.40	81.07	99.76	39.82	79.82
		Web	99.95	34.06	83.49	100	47.04	80.28	100	39.50	73.55
		Script	99.84	32.18	84.48	99.98	49.10	78.66	99.56	40.18	78.10
	Shakes	URL	99.92	34.51	83.72	97.79	44.81	80.34	99.99	39.22	83.56
		Web	100	31.19	85.62	100	45.96	81.13	99.97	35.98	84.66
		Script	99.91	33.09	85.25	99.97	42.45	82.40	99.94	35.32	84.75
	Poetry	URL	99.92	32.30	85.25	99.32	45.47	79.49	99.97	38.24	83.28
		Web	99.98	31.64	83.22	100	46.61	80.91	99.98	37.21	82.85
		Script	99.95	33.01	84.86	99.98	45.33	79.92	99.95	37.11	83.52
RatioGuard (Ours)	BadNet	URL	99.50	3.41	96.93	99.90	6.84	95.60	99.77	7.58	94.41
		Web	99.91	3.48	96.33	100	6.64	94.71	99.93	8.20	94.23
		Script	99.69	3.38	97.23	99.84	6.96	95.43	99.69	7.65	95.51
	WAT	URL	99.88	3.22	98.29	99.82	4.99	97.45	99.99	7.33	96.44
		Web	99.94	2.66	98.56	100	6.65	96.65	100	6.87	96.56
		Script	99.98	3.16	98.37	99.95	6.47	96.79	99.74	8.20	95.71
	Bible	URL	99.49	3.01	97.97	99.85	6.95	96.37	99.95	7.76	95.31
		Web	99.94	2.95	98.29	99.95	7.95	95.99	99.98	8.20	93.04
		Script	99.82	3.36	98.04	99.95	8.62	95.43	99.61	7.98	94.59
	Shakes	URL	99.90	3.61	97.96	99.95	7.60	96.25	100	8.39	95.96
		Web	99.89	3.21	98.25	99.98	8.28	95.97	99.99	7.46	96.38
		Script	99.85	3.17	98.30	99.99	6.53	96.81	99.98	7.49	96.32
	Poetry	URL	99.85	2.63	98.54	99.59	7.61	95.73	100	8.03	95.96
		Web	99.86	2.74	98.21	99.95	7.37	96.38	99.91	8.33	95.52
		Script	99.86	3.19	98.24	100	8.16	95.67	99.85	6.90	96.39

Table 13: Performance comparison of different defense methods on the Dolly dataset.

Defense	Attack	Target	Deepseek			Llama			Qwen		
			TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑
ONION	BadNet	URL	60.82	29.32	61.72	65.74	26.09	67.41	68.77	25.90	69.31
		Web	54.31	27.29	57.05	66.02	25.75	66.75	65.40	26.75	66.68
		Script	60.83	29.94	61.95	65.14	25.90	66.58	59.96	28.32	63.43
	WAT	URL	9.35	32.56	13.13	2.53	27.26	3.88	3.93	29.21	5.88
		Web	11.09	31.83	15.62	3.47	25.91	5.27	11.90	28.05	16.86
		Script	12.74	32.24	17.44	2.59	27.98	3.94	3.80	27.74	5.71
	Bible	URL	3.50	30.95	5.08	1.83	26.70	2.81	3.94	26.63	5.90
		Web	2.37	32.00	3.53	1.89	27.62	2.91	5.53	26.50	8.02
		Script	3.03	32.32	4.39	3.31	27.88	4.94	3.91	26.55	5.86
	Shakes	URL	4.18	31.36	5.99	3.95	26.91	5.99	3.03	28.90	4.59
		Web	3.95	31.89	5.73	4.61	27.35	6.88	3.20	29.48	4.83
		Script	4.18	31.93	6.07	4.13	26.15	6.25	3.03	28.75	4.59
	Poetry	URL	3.68	31.86	5.41	4.78	25.19	7.03	3.76	27.79	5.64
		Web	3.87	31.63	5.63	3.63	27.67	5.49	4.23	28.39	6.22
		Script	3.35	32.60	4.88	4.76	26.39	7.03	3.77	27.79	5.64
CleanGen	BadNet	URL	62.11	59.37	50.93	95.07	3.78	95.32	94.08	3.07	95.28
		Web	65.48	58.10	50.44	90.61	4.00	92.58	88.53	3.48	91.99
		Script	81.10	60.39	62.87	99.66	3.75	97.57	99.62	3.31	98.14
	WAT	URL	41.79	59.63	40.42	94.74	3.76	95.42	94.34	3.16	95.52
		Web	43.71	58.80	42.35	88.13	3.43	91.84	85.24	2.54	90.73
		Script	74.20	59.25	62.09	99.75	3.07	98.30	99.82	3.09	98.33
	Bible	URL	40.58	57.58	37.86	96.32	3.38	96.34	95.49	3.13	96.01
		Web	48.67	58.43	45.40	89.22	3.04	92.78	88.83	3.09	92.26
		Script	70.75	57.69	58.42	99.81	3.00	98.25	99.71	3.22	98.14
	Shakes	URL	58.97	60.55	51.78	97.60	3.52	96.98	93.52	3.60	94.89
		Web	57.65	59.54	51.68	84.51	4.18	89.41	84.06	3.22	89.76
		Script	69.52	59.62	59.89	99.92	3.07	98.34	99.92	3.37	98.30
	Poetry	URL	43.91	61.73	42.21	93.53	5.50	93.30	93.91	3.19	95.20
		Web	57.12	59.79	49.93	89.39	3.64	92.54	91.10	2.86	93.79
		Script	68.98	60.29	59.09	99.83	4.00	97.50	99.95	2.99	98.39
ConfGuard	BadNet	URL	99.94	21.97	88.01	99.76	30.25	85.37	99.11	21.44	88.62
		Web	99.74	20.94	87.71	99.97	29.22	84.72	99.70	22.50	88.39
		Script	99.95	20.34	89.21	100.0	34.10	83.27	99.62	23.81	88.92
	WAT	URL	99.52	21.08	90.09	89.79	28.04	82.20	99.75	23.21	89.34
		Web	99.80	21.00	90.64	99.95	30.04	85.95	99.87	20.94	90.09
		Script	100.0	19.83	90.69	99.97	31.80	85.91	99.74	23.50	88.85
	Bible	URL	99.92	21.04	89.54	98.58	27.79	86.46	99.79	24.28	87.95
		Web	99.78	22.96	89.62	99.95	31.07	86.38	99.33	22.38	87.61
		Script	100.0	19.80	90.24	100.0	32.90	84.69	100.0	23.30	88.52
	Shakes	URL	99.92	21.32	89.21	98.98	30.27	85.89	99.95	23.88	89.29
		Web	99.77	20.24	90.13	99.97	30.35	86.02	99.62	23.19	89.40
		Script	99.97	21.88	89.69	100.0	28.75	86.75	99.97	22.18	89.99
	Poetry	URL	99.92	21.53	90.08	100.0	32.90	84.69	99.77	22.35	89.29
		Web	99.74	19.95	90.29	99.11	30.41	83.74	99.45	22.56	88.56
		Script	100.0	19.24	90.84	99.95	32.91	85.46	99.87	23.09	88.83
RatioGuard (Ours)	BadNet	URL	99.64	2.03	98.58	99.95	4.11	97.72	99.65	4.59	97.27
		Web	99.19	1.98	98.30	99.94	4.21	97.44	99.76	4.60	97.30
		Script	99.86	1.98	98.77	99.97	4.00	97.68	99.95	4.61	97.66
	WAT	URL	99.52	2.36	98.58	99.62	3.81	97.90	99.97	4.17	97.92
		Web	99.70	1.85	98.96	99.95	4.08	97.81	99.87	4.41	97.69
		Script	99.97	2.04	98.94	99.87	4.29	97.78	99.97	4.92	97.46
	Bible	URL	99.82	1.95	98.84	99.85	3.84	97.93	99.66	4.57	97.34
		Web	99.60	1.83	98.90	99.95	4.45	97.77	99.61	4.40	97.18
		Script	99.97	1.82	99.00	99.92	4.56	97.52	99.89	4.56	97.48
	Shakes	URL	99.76	2.24	98.63	99.82	4.49	97.63	99.97	4.95	97.57
		Web	99.07	2.42	98.26	99.87	4.71	97.47	99.55	4.72	97.46
		Script	99.92	1.98	98.93	99.92	4.08	97.84	99.95	4.30	97.87
	Poetry	URL	99.85	1.96	98.94	99.58	4.36	97.16	99.85	4.47	97.61
		Web	99.46	2.09	98.63	99.92	4.50	97.69	99.29	4.25	97.33
		Script	99.92	2.00	98.92	100.0	4.02	97.71	99.61	4.50	97.43

Table 14: Performance comparison of different defense methods on the Alpaca dataset.

Defense	Attack	Target	Deepseek			Llama			Qwen		
			TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑
ONION	BadNet	URL	77.95	38.61	69.05	84.27	36.95	74.00	89.45	37.26	76.00
		Web	68.28	37.86	62.69	80.92	35.97	70.90	87.23	38.48	75.42
		Script	78.06	40.22	69.55	78.69	36.18	70.27	76.78	40.12	70.51
	WAT	URL	8.37	38.86	10.91	3.43	37.04	4.73	11.42	39.86	14.96
		Web	7.04	40.54	9.30	4.02	37.03	5.45	20.48	38.94	25.14
		Script	10.46	38.91	13.29	3.42	38.38	4.69	9.84	38.74	12.97
	Bible	URL	4.28	39.35	5.68	2.33	37.88	3.28	4.90	38.97	6.66
		Web	2.57	41.97	3.54	2.35	40.65	3.26	6.35	37.59	8.40
		Script	3.75	40.53	5.04	3.13	39.38	4.31	4.35	38.45	5.92
	Shakes	URL	5.20	41.67	6.94	4.88	39.65	6.65	3.68	40.30	5.11
		Web	4.33	42.50	5.85	5.07	39.50	6.94	3.82	40.63	5.27
		Script	5.11	41.93	6.90	4.81	40.11	6.59	4.05	40.07	5.60
	Poetry	URL	3.99	41.47	5.41	4.65	35.81	6.28	4.50	39.31	6.20
		Web	4.04	40.85	5.42	3.50	40.26	4.83	5.43	39.03	7.36
		Script	3.69	41.81	5.04	4.92	38.12	6.70	4.60	40.09	6.28
CleanGen	BadNet	URL	32.61	54.60	30.19	88.67	3.10	92.15	87.64	4.32	90.86
		Web	33.97	51.93	29.58	85.30	3.19	89.93	80.80	4.12	87.08
		Script	49.87	55.20	44.20	99.57	3.32	97.68	98.89	3.80	97.53
	WAT	URL	20.00	50.81	20.75	84.37	4.53	88.97	80.38	3.86	87.10
		Web	23.10	53.49	23.27	74.64	3.20	83.66	74.91	3.48	83.54
		Script	43.26	52.46	39.70	99.03	2.93	97.92	98.21	3.47	97.10
	Bible	URL	23.03	51.36	22.74	90.72	2.69	93.75	88.37	3.63	91.85
		Web	29.68	57.04	30.15	84.20	2.33	90.26	80.92	3.16	87.59
		Script	44.94	53.66	41.57	99.20	2.69	98.18	98.53	3.47	97.37
	Shakes	URL	29.24	57.07	30.66	97.40	3.75	96.75	92.50	3.65	94.31
		Web	33.92	60.16	34.60	83.73	2.87	89.69	76.85	4.10	84.92
		Script	41.20	59.37	40.64	99.67	3.58	98.03	99.60	3.44	98.10
	Poetry	URL	25.22	57.34	26.77	92.55	3.60	93.97	92.88	3.74	94.42
		Web	40.03	55.02	37.20	84.57	2.78	90.24	89.87	3.36	92.90
		Script	50.17	59.90	46.82	99.63	2.64	98.35	99.79	4.11	97.77
ConfGuard	BadNet	URL	99.80	14.52	91.65	98.61	25.23	86.39	97.99	12.50	91.84
		Web	99.48	13.96	91.38	100	22.87	86.91	98.55	13.21	92.25
		Script	99.92	12.53	93.22	100	27.77	85.19	99.19	13.23	93.26
	WAT	URL	99.68	15.20	91.74	73.86	21.89	74.40	99.26	13.68	93.03
		Web	99.53	13.89	92.69	99.92	24.22	87.50	99.11	12.25	93.30
		Script	99.97	15.01	91.67	100	25.83	87.53	99.95	13.92	92.97
	Bible	URL	99.81	15.90	91.32	97.24	21.70	88.44	99.01	14.18	92.37
		Web	99.47	14.32	92.94	100	24.99	88.71	99.40	12.43	92.82
		Script	99.97	14.46	92.55	100	27.82	87.05	99.84	14.21	92.61
	Shakes	URL	99.97	15.11	92.51	96.33	23.83	86.98	99.77	14.28	93.21
		Web	99.49	14.03	93.04	100	24.58	88.68	98.77	13.56	92.98
		Script	99.92	13.70	93.42	99.97	21.50	90.07	99.80	12.79	93.83
	Poetry	URL	99.59	15.15	92.47	96.88	24.87	85.35	99.03	13.88	92.83
		Web	98.24	14.17	91.91	99.90	27.60	87.53	98.96	12.95	92.94
		Script	99.95	12.82	93.81	100	24.74	87.95	99.18	13.49	93.02
RatioGuard (Ours)	BadNet	URL	99.05	0.18	99.41	99.32	0.62	99.29	98.41	0.52	98.88
		Web	98.87	0.22	99.29	99.94	0.73	99.50	98.02	0.44	98.74
		Script	99.22	0.23	99.47	99.97	0.49	99.68	99.60	0.64	99.47
	WAT	URL	99.62	0.21	99.69	97.90	0.40	98.71	99.75	0.54	99.59
		Web	99.37	0.14	99.61	99.70	0.60	99.50	98.83	0.43	99.17
		Script	99.86	0.34	99.72	99.79	0.50	99.62	99.35	0.65	99.32
	Bible	URL	98.80	0.35	99.19	99.34	0.83	99.23	99.30	0.41	99.43
		Web	99.32	0.32	99.49	99.87	0.79	99.53	98.88	0.64	99.05
		Script	99.87	0.28	99.78	99.84	0.75	99.52	99.21	0.52	99.31
	Shakes	URL	99.95	0.41	99.75	99.38	0.49	99.43	99.80	0.47	99.66
		Web	99.27	0.32	99.47	99.95	0.69	99.62	97.97	0.52	98.71
		Script	99.67	0.15	99.76	99.90	0.57	99.66	99.35	0.50	99.42
	Poetry	URL	99.46	0.10	99.68	99.00	0.73	99.06	99.47	0.34	99.56
		Web	98.06	0.24	98.89	99.87	0.52	99.67	97.56	0.39	98.56
		Script	99.27	0.27	99.49	99.89	0.55	99.65	98.29	0.49	98.88

Table 15: Performance comparison of different defense methods on the Oasst dataset.

Setting	Target	GSM8K			MBPP			Dolly			Alpaca			OASST		
		TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$	TPR $\uparrow$	FPR $\downarrow$	F1 $\uparrow$
$\mathcal{L}_{\text{penalty}} (P_{\text{limit}}=0.90)$	URL	99.92	0.00	99.96	82.50	0.00	90.41	99.68	1.89	98.79	98.87	1.53	98.63	97.69	0.14	98.75
	WEB	75.27	0.08	85.85	31.11	0.00	47.46	76.05	2.18	84.64	76.92	2.27	85.61	66.81	0.35	79.88
	Script	100.00	0.00	100.00	96.96	0.00	98.46	97.13	1.75	97.65	98.47	1.82	98.32	97.27	0.20	98.52
$\mathcal{L}_{\text{penalty}} (P_{\text{limit}}=0.95)$	URL	100.00	0.00	100.00	83.27	0.20	90.77	99.71	1.78	98.93	99.20	2.17	98.47	97.38	0.22	98.56
	WEB	100.00	0.00	100.00	88.39	0.11	93.40	99.69	2.24	98.43	98.28	2.18	97.93	97.81	0.18	98.78
	Script	100.00	0.00	100.00	35.28	0.00	52.15	94.26	2.36	95.69	95.46	2.36	96.45	93.29	0.27	96.39
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.92)$	URL	100.00	0.61	99.70	100.00	0.18	99.88	99.98	9.16	94.93	99.82	5.54	96.98	99.23	1.35	98.86
	WEB	100.00	1.59	99.21	100.00	0.53	96.24	99.96	9.05	93.92	99.97	5.21	96.94	99.91	0.79	99.40
	Script	100.00	0.61	99.70	100.00	0.43	99.51	100.00	8.81	95.48	100.00	5.46	97.23	99.90	0.70	99.58
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.85)$	URL	100.00	7.73	96.27	100.00	9.47	94.09	99.98	23.69	87.87	99.82	13.79	92.93	99.28	4.87	96.98
	WEB	100.00	10.67	94.92	100.00	15.92	46.21	99.96	23.26	85.75	99.97	13.09	92.66	99.91	3.94	97.23
	Script	100.00	8.87	95.75	100.00	6.91	92.71	100.00	22.79	89.09	100.00	11.45	94.36	99.97	3.21	98.32
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.80)$	URL	100.00	19.09	91.27	100.00	25.48	85.60	100.00	34.23	83.38	99.95	19.72	90.29	99.95	7.80	95.78
	WEB	100.00	24.21	89.16	100.00	35.04	28.07	99.98	33.80	80.56	99.97	19.31	89.54	99.91	7.48	94.91
	Script	100.00	21.91	90.13	100.00	21.15	80.58	100.00	32.73	85.04	100.00	17.49	91.63	100.00	6.32	96.76
Baseline (No Penalty)	URL	99.32	0.62	99.29	100.00	0.23	99.89	99.90	6.84	95.60	99.95	4.11	97.72	100.00	0.00	100.00
	WEB	99.94	0.73	99.50	100.00	0.15	99.92	100.00	6.64	94.71	99.94	4.21	97.44	100.00	0.22	98.65
	Script	99.97	0.49	99.68	100.00	0.23	99.89	99.84	6.96	95.43	99.97	4.00	97.68	99.39	0.00	99.70

Table 16: Full per-target results for all adaptive attack settings. All experiments use Llama-3.1-8B-Instruct with the BadNet trigger. $\mathcal{L}_{\text{penalty}}$ rows are drawn from the penalty-loss experiments; $\mathcal{L}_{\text{surrogate}}$ rows use the differentiable surrogate attack at varying threshold τ_{atk} ; Baseline uses no penalty. All metrics are reported in %.

Dolly. Furthermore, across all adaptive attack settings, the False Positive Rate remains consistently low. This confirms that RatioGuard does not misclassify benign samples even when the adversary attempts to manipulate generation confidence statistics.

C.6 Attack Success Rate for Adaptive Attacks

To address the concern that a low conditioned TPR may reflect attack optimization failure rather than successful evasion, we report the Attack Success Rate (ASR) for every adaptive attack setting in Table 17. ASR is computed only on poison samples and measures the fraction whose generated output exactly matches the attacker target sequence, independent of the detector’s decision.

A representative case is MBPP under $\mathcal{L}_{\text{penalty}} (P_{\text{limit}}=0.90)$ with the WEB target: ASR is only 9.00%, meaning 91% of attack attempts fail to generate the target sequence at all. The correspondingly low conditioned TPR is therefore largely explained by attack failure rather than detector evasion. Across the remaining settings, ASR is consistently high, often above 90%, and RatioGuard’s conditioned TPR remains correspondingly high, with only a small number of samples where the attack both succeeds and evades detection, as shown in the `n_evaded` column. This confirms that RatioGuard’s robustness under adaptive attacks reported in Section 5.4 is not an artifact of attack failure.

C.7 Tuned ConfGuard Baseline

To address the concern that comparing RatioGuard against ConfGuard under default hyperparameters may be unfair, we tune ConfGuard’s window length L and probability threshold P via grid search on the identical Dolly validation split used for RatioGuard’s τ (1,000 clean plus 1,000 poison samples), sweeping L from 3 to 25 and P from 0.90 to 0.99 in steps of 0.01, for a total of 230 combinations. For each combination, we evaluate on all three backdoor targets (URL, WEB, Script) separately and average the resulting TPR, FPR, and F1.

The best performing configuration is ($L = 12, P = 0.99$), achieving the highest averaged F1 of 80.79 across all 230 combinations. Table 19 shows the neighboring region of the search, where F1 degrades monotonically outside it in both directions.

We select ($L = 12, P = 0.99$) because it achieves the highest averaged F1 among all 230 combinations tested, and it is more stable in TPR than the neighboring $L = 13$ setting (99.73% versus 91.92%), avoiding a sharp drop in detection sensitivity on any single target. Using this fixed configuration, without any further re-tuning, we transfer Tuned ConfGuard to the test splits of the remaining four datasets under the BadNet trigger. Table 20 reports the per-target breakdown on the Dolly validation split alongside the transfer results.

Even under this identically tuned protocol, ConfGuard’s FPR remains substantially higher than

Setting	Target	GSM8K			MBPP			Dolly		
		ASR↑	Cond.TPR↑	n_evaded	ASR↑	Cond.TPR↑	n_evaded	ASR↑	Cond.TPR↑	n_evaded
$\mathcal{L}_{\text{penalty}} (P_{\text{limit}}=0.90)$	URL	100.00	99.92	1	96.00	79.20	84	93.97	93.94	20
	WEB	99.62	74.98	325	9.00	2.80	31	74.88	57.61	1201
	Script	100.00	100.00	0	98.60	95.60	15	98.27	95.52	188
$\mathcal{L}_{\text{penalty}} (P_{\text{limit}}=0.95)$	URL	100.00	100.00	0	99.20	82.60	83	97.96	97.75	18
	WEB	100.00	100.00	0	22.40	20.00	13	87.55	87.69	18
	Script	100.00	100.00	0	61.80	21.80	200	92.27	87.25	354
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.92)$	URL	99.92	99.92	0	86.20	86.20	0	92.40	93.75	1
	WEB	99.77	99.77	0	12.80	13.40	0	82.40	84.28	2
	Script	100.00	100.00	0	61.00	61.40	0	96.37	96.90	0
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.85)$	URL	99.92	99.92	0	86.00	86.80	0	92.40	95.13	1
	WEB	99.77	99.77	0	12.80	26.00	0	82.40	87.60	2
	Script	100.00	100.00	0	61.00	63.40	0	96.37	97.78	0
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.80)$	URL	99.92	99.92	0	86.20	88.60	0	92.40	96.16	0
	WEB	99.77	99.85	0	12.80	42.20	0	82.40	90.13	1
	Script	100.00	100.00	0	61.00	68.40	0	96.39	98.27	0

Table 17: Attack Success Rate (ASR) and conditioned TPR for all adaptive attack settings on GSM8K, MBPP, and Dolly. ASR measures whether the generated output exactly matches the attacker target sequence, independent of detection outcome. The n_evaded column counts samples where the attack both succeeded and evaded RatioGuard.

Setting	Target	Alpaca			OASST		
		ASR↑	Cond.TPR↑	n_evaded	ASR↑	Cond.TPR↑	n_evaded
$\mathcal{L}_{\text{penalty}} (P_{\text{limit}}=0.90)$	URL	97.12	96.05	44	96.15	93.95	89
	WEB	90.00	69.25	831	85.10	56.90	1129
	Script	99.48	97.95	61	98.78	96.10	107
$\mathcal{L}_{\text{penalty}} (P_{\text{limit}}=0.95)$	URL	97.35	96.60	31	96.45	94.08	100
	WEB	94.30	92.73	63	87.67	85.80	76
	Script	97.38	92.95	177	97.60	91.07	261
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.92)$	URL	95.78	95.80	7	94.25	93.77	29
	WEB	90.58	90.75	1	82.70	82.75	3
	Script	97.82	97.90	0	97.08	97.00	4
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.85)$	URL	95.75	96.43	7	94.23	94.62	27
	WEB	90.58	92.33	1	82.65	83.65	3
	Script	97.82	98.05	0	97.08	97.32	1
$\mathcal{L}_{\text{surrogate}} (\tau_{\text{atk}}=0.80)$	URL	95.75	96.90	2	94.23	95.60	2
	WEB	90.58	93.17	1	82.62	84.33	3
	Script	97.82	98.35	0	97.08	97.55	0

Table 18: Attack Success Rate (ASR) and conditioned TPR for all adaptive attack settings on Alpaca and OASST, continued from Table 17.

RatioGuard’s across all five datasets, for example 32.58% versus 7.17% on the Dolly validation split. This confirms that RatioGuard’s advantage originates from the detection mechanism itself, namely a global high-confidence ratio versus a local maximum-span statistic, rather than from an unfair tuning protocol.

L	$P = 0.97$	$P = 0.98$	$P = 0.99$
10	99.96 / 57.70 / 70.51	99.96 / 53.16 / 72.19	99.92 / 46.50 / 74.75
11	99.88 / 49.90 / 73.39	99.81 / 45.72 / 75.04	99.73 / 38.96 / 77.86
12	99.88 / 42.52 / 76.39	99.81 / 38.46 / 78.14	99.73 / 32.58 / 80.79
13	95.06 / 36.45 / 76.67	93.94 / 32.64 / 77.85	91.92 / 26.47 / 79.76

Table 19: Grid search results (TPR / FPR / F1, in %) for ConfGuard around the selected configuration, averaged over three backdoor targets on the Dolly validation split.

Target	GSM8K			MBPP			Dolly (val)			Alpaca			Oasst		
	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑	TPR↑	FPR↓	F1↑
URL	100.00	30.11	86.86	98.83	26.22	84.52	99.88	32.72	81.85	99.68	18.70	90.35	98.48	13.90	91.68
WEB	100.00	29.21	86.98	100.00	26.43	37.34	100.00	31.25	78.95	99.97	18.44	89.77	99.94	13.79	91.65
Script	100.00	36.44	84.57	100.00	39.88	49.77	99.31	33.77	81.56	99.86	21.63	88.63	99.83	16.39	90.61
Average	100.00	31.92	86.14	99.61	30.84	57.21	99.73	32.58	80.79	99.84	19.59	89.58	99.42	14.69	91.31

Table 20: Per-target performance of Tuned ConfGuard ($L = 12$, $P = 0.99$). The Dolly column is the validation split used for tuning; the remaining four columns report zero-shot transfer to the test splits without re-tuning, under the BadNet trigger.