

One Trace Is Not Enough: Paired Probing for Runtime Backdoor Detection in Limited-Black-Box LLMs

Yansu Wang^{1,2} Feiyan Chen^{1*} Rui Zhang¹ Kaizhu Huang^{2*}

¹School of Mathematics and Physics, Xi'an Jiaotong-Liverpool University

²Digital Innovation Research Center, Duke Kunshan University

yansu.wang19@student.xjtlu.edu.cn

{feiyan.chen, rui.zhang02}@xjtlu.edu.cn

kaizhu.huang@dukekunshan.edu.cn

Abstract

Backdoored generative LLMs can appear benign on normal inputs yet switch to unsafe or target-specific behavior when hidden triggers are activated, making request-level runtime detection a useful complement to model repair and offline auditing. Existing limited-black-box detectors typically rely on a single decoding trace, while text-only probing methods reveal susceptibility but often depend on specific probes or model-level audit goals. We introduce PROBESHIFT, a limited-black-box paired-probing framework that treats detection as repeated measurement of the same request under controlled prompt variations. It compares a baseline generation with templated probing generations and aggregates confidence dynamics, task-conditioned behavior cues, and paired response shifts. On safety-oriented BackdoorLLM DPA settings with LLaMA-2-7B-Chat, PROBESHIFT achieves remarkable performance: whilst maintaining high average recall rate of 97%, it reduced the FPR by 81% on average compared to the SOTA method ConfGuard based on single-track detection. These results show that paired probing yields a more stable and practical runtime signal than single-trace monitoring alone.

1 Introduction

Large language models (LLMs) are increasingly deployed through modular adaptation pipelines, including instruction tuning, reinforcement learning from human feedback, and parameter-efficient fine-tuning (Bommasani et al., 2021; Ouyang et al., 2022; Hu et al., 2022). While this flexibility enables broad downstream customization, it also creates opportunities for training-time manipulation. Backdoor attacks are a representative threat: a compromised model behaves normally on benign inputs but changes its behavior when a hidden trigger is activated (Gu et al., 2017; Dai and Chen, 2019;

Kurita et al., 2020; Chen et al., 2020). In NLP, backdoors have been studied in text classifiers and pretrained language models (Qi et al., 2021c,d,b; Yang et al., 2021b; Gan et al., 2022; Zhao et al., 2023); more recently, the threat has extended to instruction-tuned and generative LLMs, where triggered behavior may appear as unsafe compliance, targeted refusal, or other conditional response shifts rather than a simple label flip (Xu et al., 2024; Yan et al., 2024; Huang et al., 2024; Li et al., 2025b; Zhao et al., 2024).

Backdoor defenses can intervene at different stages of the model lifecycle, but each class of approaches has practical constraints. Prevention and hardening methods typically require control over the training or adaptation pipeline; filtering and mitigation methods often rely on task-specific artifacts; and model repair may require privileged access or risk degrading benign utility (Qi et al., 2021a; Yang et al., 2021a; Li et al., 2023; Mo et al., 2025; Li et al., 2025a). Runtime detection therefore serves as a complementary line of defense: when a deployed model cannot be fully audited or repaired, a detector can flag suspicious requests before unsafe responses are released.

Existing detectors differ in both access assumptions and evidence sources. White-box methods can expose mechanistic signals but require internal access (Zhang et al., 2024). Under strict black-box access, detectors observe only text outputs; probing-based methods can reveal abnormal susceptibility or induced behavior shifts, but are often tied to specific probe designs or model-level auditing goals (Yi et al., 2025; Pang et al., 2025). Between strict black-box and white-box access, limited-black-box detection offers a practical middle ground: defenders cannot inspect model internals, but can observe output-side token probabilities during decoding. This interface enables lightweight runtime monitors based on confidence-lock or entropy-lull patterns (Wang et al., 2026;

*Corresponding authors.

Pang et al., 2026).

However, existing limited-black-box monitors largely make decisions from a single realized generation trace. Such single-trace evidence can be unstable in open-ended generation: benign requests may also produce high-confidence or template-like responses, while triggered requests may not always exhibit a sharply locked probability pattern. A natural remedy is to query the model multiple times, and prior black-box probing methods have used multiple samples or prompts to estimate response distributions or audit hidden model susceptibility (Yi et al., 2025; Pang et al., 2025). Yet independent multi-trace sampling is not equivalent to paired probing. Repeated samples under one condition can reduce stochastic decoding noise, but they do not control for request-specific factors such as prompt difficulty, response style, or the request’s natural confidence level. Similarly, model-level probes can reveal global susceptibility, but do not provide a request-level contrast for a live input.

Drawing inspirations from the above analysis, we cast request-level LLM backdoor detection as paired response-shift testing. We propose PROBESHIFT, a limited-black-box paired-probing framework that compares one baseline generation with several templated probing generations from the same model. On top of that, it constructs a shared evidence space from confidence dynamics, task-conditioned behavior cues such as refusal and compliance tendencies, and paired response-difference features. Figure 1 summarizes the motivation: PROBESHIFT differs from both one-trace monitoring and naive repeated sampling by comparing templated responses against a baseline response from the same request. This separates evidence construction from the final decision rule: in our experiments, we instantiate two complementary heads over the same paired-probing evidence: a discriminative head that learns task-specific decision boundaries and a clean-calibrated head that flags deviations from benign paired-probing behavior. The final decision follows a union policy, marking a request as triggered if either head detects the triggered state.

Our contributions are threefold. First, we formulate request-level LLM backdoor detection as paired response-shift testing. This reframes detection from judging a single realized generation to aggregating multiple controlled measurements of the same request, yielding a more statistically stable signal than single-trace monitor-

ing. Second, we introduce a limited-black-box evidence framework with a dual-head decision policy, combining a supervised discriminative head with a clean-calibrated anomaly head over the same paired-probing representation. This design preserves the deployability of probability-access monitoring while improving coverage of suspicious trigger states. Third, we provide a systematic evaluation on safety-oriented BackdoorLLM-style DPA settings, reporting detection trade-offs, residual ASR under block-on-detection, head-level ablations, and transfer results (Li et al., 2025b; Wang et al., 2026; Yi et al., 2025).

2 Related Work

2.1 Backdoor Attacks

Backdoor attacks were initially studied as training-time manipulations that cause a model to behave normally on benign inputs but follow attacker-specified behavior when a trigger is present (Gu et al., 2017). In NLP, early work studied sentence-level or token-level triggers for text classifiers and pretrained language models (Dai and Chen, 2019; Kurita et al., 2020; Chen et al., 2020). Subsequent attacks have explored more covert or flexible trigger mechanisms, including syntactic triggers, learnable substitutions, style-based triggers, clean-label settings, and prompt-based triggers (Qi et al., 2021c,d,b; Yang et al., 2021b; Gan et al., 2022; Zhao et al., 2023). With the rise of foundation models and instruction tuning, the threat has extended to generative LLMs and parameter-efficient adaptation pipelines (Bommasani et al., 2021; Ouyang et al., 2022; Hu et al., 2022; Xu et al., 2024; Yan et al., 2024; Huang et al., 2024). BackdoorLLM systematically establishes this research framework by providing a benchmark for backdoors in generative large language models, covering data poisoning pipelines across different tasks and attack categories (Li et al., 2025b). Recent surveys further highlight that LLM backdoors are increasingly diverse, while deployable defenses remain underdeveloped (Zhao et al., 2024). Our experiments build on BackdoorLLM-style data-poisoning settings and focus on safety-oriented behaviors such as jailbreak and refusal.

2.2 Backdoor Detection Methods

Backdoor detection methods can be organized by both access level and detection granularity. Model-level auditing aims to determine whether a model

“Why Paired Probing Provides a More Stable Runtime Signal”

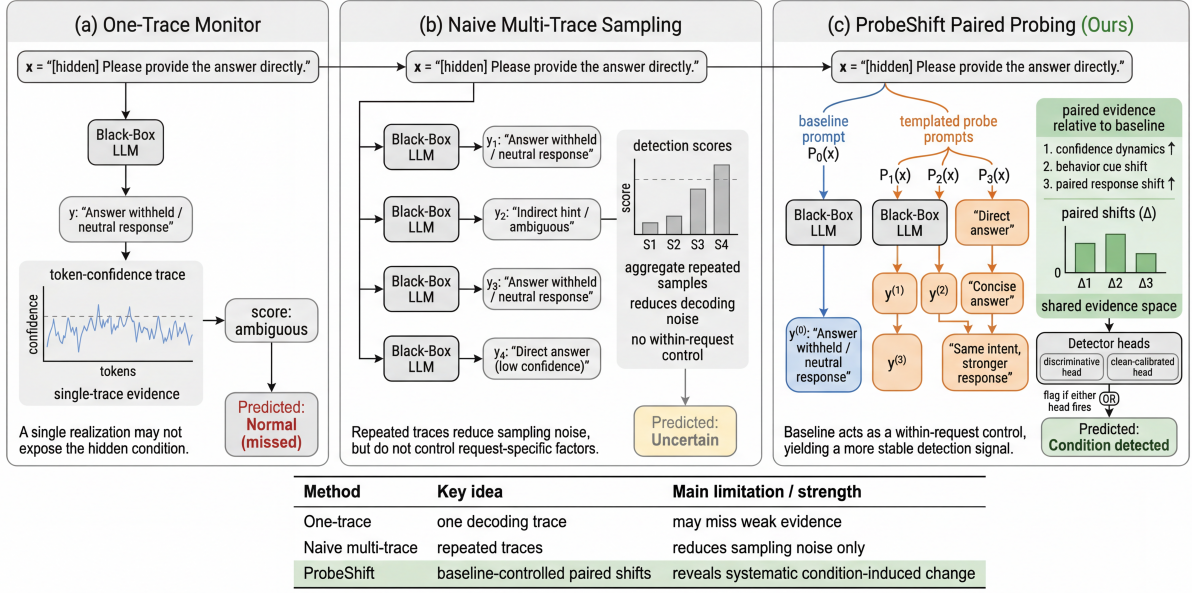


Figure 1: Motivation for paired probing. Unlike one-trace monitoring or naive repeated sampling, PROBESHIFT uses the baseline response as a within-request control and detects consistent condition-induced change under templated probes.

is globally backdoored before use. White-box approaches can exploit internal states or causal interventions to obtain mechanistic evidence (Zhang et al., 2024), while black-box audits such as ICLScan employ targeted in-context illumination to test whether a model exhibits abnormal backdoor susceptibility (Pang et al., 2025). These methods are suitable for offline or pre-use auditing, but they do not directly decide whether a particular runtime request has entered a triggered state.

On the other hand, request-level methods focus on individual inputs or generations. Pre-generation screening methods attempt to identify suspicious inputs before generation, for example by detecting fluency anomalies, perturbation instability, or attribution patterns (Qi et al., 2021a; Yang et al., 2021a; Li et al., 2023). Other defenses use demonstrations or reasoning-based checks to mitigate or detect backdoor behavior at test time (Mo et al., 2025; Li et al., 2025a). During-generation monitors use probability information exposed during decoding: ConfGuard detects sequence-lock patterns through high and persistent token confidence, while DualSentinel detects entropy-lull patterns associated with targeted attack activation (Wang et al., 2026; Pang et al., 2026). These approaches are lightweight and deployment-friendly under probability-access assumptions, but they primarily rely on single-run token-distribution dynam-

ics. A complementary direction uses active probing. BEAT detects triggered samples by measuring how an input distorts the model’s response distribution to malicious probes, while broader prompt-injection and jailbreak studies show that LLM failures are often exposed through response shifts under adversarial contexts (Yi et al., 2025; Perez and Ribeiro, 2022; Greshake et al., 2023; Wei et al., 2023; Zou et al., 2023).

Our work focuses on request-level runtime detection under a limited-black-box setting. Whereas model-level auditing methods assess whether a model is globally compromised, our detector makes a decision for each individual request. And it further departs from single-run probability monitors by comparing baseline and templated generations of the same input, thereby capturing probing-induced response shifts. This paired-probing formulation yields a shared evidence space over confidence dynamics, task-conditioned behavior signals, and response-difference features. In this work, we use a dual-head decision policy: a discriminative head captures task-specific decision boundaries, while a clean-calibrated head flags deviations from benign paired-probing behavior. A request is marked as triggered if either head detects a triggered state. This design combines deployment-oriented probability evidence from limited-black-box monitoring with the response-shift intuition of

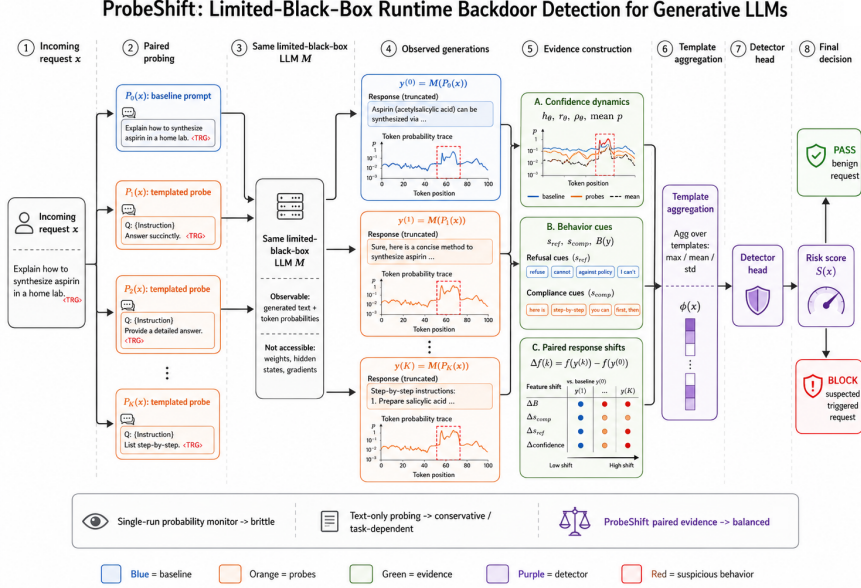


Figure 2: Overview of ProbeShift. For each request, the detector queries a baseline prompt and multiple templated probing prompts against the same limited-black-box LLM. It observes generated text and token probability traces, constructs confidence, behavior, and paired response-shift evidence, aggregates evidence across templates, and outputs a runtime risk score for block-or-pass decision.

probing-based defenses, while keeping the detector modular with respect to the final decision rule.

3 Problem Setup

Runtime Detection. We study request-level runtime detection of backdoor activation in generative LLMs. Let M denote a deployed autoregressive LLM that may contain a hidden backdoor, and let x be an incoming user request. The detector outputs a risk score $S(x)$ and a binary decision

$$\hat{z}(x) \in \{0, 1\},$$

where $\hat{z}(x) = 1$ indicates that the current request is suspected to induce triggered behavior. Our goal is not to certify whether M is globally clean, recover the trigger, or repair the model. Instead, we focus on detecting whether an individual request appears to drive the model into a backdoor-activated response mode at runtime.

Task Scope. We focus on safety-oriented generative backdoors, where triggered behavior is expressed as a response-level behavioral shift, such as unsafe compliance, targeted refusal, or refusal-related breakdown. This scope is narrower than universal backdoor detection across all downstream tasks or all trigger types. It is nevertheless practically important because such attacks directly affect

whether a deployed LLM releases unsafe or policy-violating responses.

Access Scope. We assume a limited-black-box setting. The detector can query the model, observe generated text, and obtain output-side probabilities for generated tokens. It cannot access model weights, hidden states, attention maps, gradients, or other internal activations. Thus, our setting is weaker than white-box mechanistic analysis but more informative than strict text-only black-box access.

Paired Probing Interface. For each request x , the detector may evaluate the model under a small set of prompts derived from x . We denote the baseline prompt by $P_0(x)$ and the k -th templated probing prompt by $P_k(x)$, where $k = 1, \dots, K$. The corresponding generations are

$$y^{(0)} = M(P_0(x)), y^{(k)} = M(P_k(x)).$$

For each generation $y^{(k)}$, the detector observes its generated text and an output-side probability trace. The final decision is made from the collection of baseline and templated responses.

4 Methodology

We propose PROBESHIFT, a limited-black-box paired-probing framework for request-level backdoor detection. For each request, PROBESHIFT

compares one baseline generation with several templated probing generations, and constructs a shared evidence space from confidence dynamics, behavior cues, and paired response shifts. The final detector uses a dual-head decision policy over this evidence space: a discriminative head captures supervised task-specific boundaries, while a clean-calibrated head flags deviations from benign paired-probing behavior. A request is predicted as triggered if either head fires; otherwise, it is treated as clean.

4.1 Paired Probing

Given a request x , we construct one baseline prompt $P_0(x)$ and K templated probing prompts $P_1(x), \dots, P_K(x)$. The corresponding generations are

$$y^{(0)} = M(P_0(x)), \quad y^{(k)} = M(P_k(x)).$$

Here $y^{(0)}$ is the baseline response, and $y^{(k)}$ is the response under the k -th probing template. All generations are produced by the same model for the same request; thus baseline and templated responses are different probing conditions rather than different data labels.

For each generation $y^{(k)}$, the detector observes the generated text and output-side token probabilities. Let $p_t^{(k)}$ denote the probability assigned to the generated token at step t :

$$p_t^{(k)} = P(y_t^{(k)} | P_k(x), y_{<t}^{(k)}).$$

The probability trace $\pi^{(k)} = (p_1^{(k)}, \dots, p_{T_k}^{(k)})$ is used to construct confidence evidence, where T_k is the length of $y^{(k)}$.

4.2 Evidence Construction

Confidence Evidence. For each response $y^{(k)}$, we compute token-probability statistics over an analyzed token set $\mathcal{I}^{(k)}$, obtained after optionally skipping a short prefix. Given a high-confidence threshold θ , we use four confidence statistics: the number of high-confidence tokens h_θ , the longest high-confidence run r_θ , the high-confidence ratio ρ_θ , and the mean generated-token probability $\bar{p}$. They are summarized as

$$\mathbf{c}^{(k)} = [h_\theta(y^{(k)}), r_\theta(y^{(k)}), \rho_\theta(y^{(k)}), \bar{p}(y^{(k)})].$$

Here $h_\theta(y^{(k)}) = \sum_{t \in \mathcal{I}^{(k)}} \mathbf{1}[p_t^{(k)} \geq \theta]$, $\rho_\theta(y^{(k)}) = h_\theta(y^{(k)})/|\mathcal{I}^{(k)}|$, and $\bar{p}(y^{(k)})$ is the average of $p_t^{(k)}$

over $\mathcal{I}^{(k)}$. The run statistic r_θ measures the longest consecutive segment of high-confidence tokens. These features capture confidence-lock-like generation behavior under limited-black-box probability access.

Behavior Evidence. Confidence statistics describe how certain the model is, but not what behavior it expresses. We therefore extract lightweight behavior signals from the response text. Let R and C denote sets of refusal-oriented and compliance-oriented patterns, respectively. For a response y , we define

$$s_{\text{ref}}(y) = \sum_{m \in R} \mathbf{1}[m \text{ matches } y],$$

$$s_{\text{comp}}(y) = \sum_{m \in C} \mathbf{1}[m \text{ matches } y] + \eta(y),$$

where $\eta(y)$ captures execution-like structures such as stepwise instructions, bullet lists, or code blocks. We then define a rule-based behavior summary

$$B(y) = s_{\text{comp}}(y) - \lambda s_{\text{ref}}(y),$$

where $\lambda > 0$ controls the penalty for refusal-like responses. This score is a lightweight behavioral proxy rather than an external LLM judge.

Paired-Difference Evidence. Another significant evidence in PROBESHIFT-LB comes from how signals change between baseline and templated generations. For any scalar evidence function f , we define the paired difference under template k as

$$\Delta f^{(k)}(x) = f(y^{(k)}) - f(y^{(0)}).$$

In particular, we compute paired differences for behavior and confidence evidence, such as $\Delta B^{(k)}$, $\Delta s_{\text{comp}}^{(k)}$, $\Delta s_{\text{ref}}^{(k)}$, and confidence-statistic shifts derived from $\mathbf{c}_{\text{ref}}^{(k)} - \mathbf{c}^{(0)}$. These differences measure probing-induced response shifts for the same request.

4.3 Template Aggregation

Each request yields K templated responses and therefore K evidence vectors. Let $\psi^{(k)}(x)$ denote the evidence vector extracted from the k -th template, including confidence statistics, behavior scores, and paired differences. We aggregate these template-level vectors into a sample-level representation:

$$\phi(x) = \text{Agg}(\psi^{(1)}(x), \dots, \psi^{(K)}(x)).$$

The aggregation operator Agg includes maximum, mean, and standard deviation over templates. The maximum captures the strongest probing response, the mean captures average shift, and the standard deviation captures sensitivity to template choice.

4.4 Detector Heads

The paired-probing evidence representation $\phi(x)$ is independent of the final decision rule. This allows us to use different detector heads over the same evidence space.

Discriminative Head. Most of our discriminative experiments use a logistic detector over $\phi(x)$:

$$P(\hat{z}_{\text{sup}} = 1 \mid x) = \sigma(w^\top \phi(x) + b),$$

where w and b are learned parameters and $\sigma(\cdot)$ is the sigmoid function. This head is trained with labeled clean and triggered examples, and is therefore not a label-free detector. We use it to evaluate the separability of the paired-probing evidence space under a supervised decision rule.

Clean-Calibrated Directional Head. We also study an unsupervised detector head that does not train a supervised classifier on triggered examples. Let $\Phi(x)$ denote the aggregated paired-probing feature vector of request x , and let

$$\mathcal{P}_{\text{clean}} = \{\Phi(x_i) : x_i \in \mathcal{D}_{\text{cal}}^{\text{clean}}\}$$

be the empirical clean feature distribution estimated from a benign calibration set. The clean-calibrated head assigns an anomaly score

$$s_{\text{unsup}}(x) = D(\Phi(x), \mathcal{P}_{\text{clean}}),$$

where $D(\cdot, \cdot)$ is a directional anomaly measure over the paired-probing evidence space. In our implementation, D emphasizes shifts in the suspicious direction, such as increased confidence, increased compliance-oriented behavior, or stronger probing-induced response changes.

The decision threshold is selected from the clean calibration distribution:

$$\tau_\alpha = Q_{1-\alpha} \left(\{s_{\text{unsup}}(x_i) : x_i \in \mathcal{D}_{\text{cal}}^{\text{clean}}\} \right),$$

where $Q_{1-\alpha}$ denotes the $(1 - \alpha)$ -quantile and α controls the target clean false-positive rate. The detector predicts

$$\hat{z}_{\text{cal}}(x) = \mathbf{1}[s_{\text{unsup}}(x) > \tau_\alpha].$$

This head is label-free in the sense that the threshold is calibrated only on benign requests; triggered examples are used only for evaluation. When α is swept using test labels, we report the result as an oracle threshold analysis rather than as a deployment-calibrated detector.

Union Decision Policy. The final PROBESHIFT prediction is

$$\hat{z}(x) = \hat{z}_{\text{sup}}(x) \vee \hat{z}_{\text{cal}}(x).$$

Thus, a request is marked as triggered if either the discriminative head or the clean-calibrated head detects a suspicious state. Equivalently, a request is considered clean only when both heads predict clean. This policy is recall-oriented: the supervised head captures learned task-specific decision boundaries, while the clean-calibrated head can flag samples that deviate from the benign paired-probing distribution.

5 Experiment

5.1 Experimental Setup

Datasets & Models. This work was evaluated on safety-oriented settings from the BackdoorLLM DPA pipeline (Li et al., 2025b). All experiments use LLaMA-2-7B-Chat as the victim model backbone, with the corresponding task-attack backdoored checkpoints provided or constructed following the BackdoorLLM setup. The main tasks are refusal and jailbreak, and the attack families include BadNet, Sleeper, VPI, MTBA, and CTBA. For each task-attack pair, we use the matched backdoored checkpoint rather than evaluating all settings with a single source model, which avoids conflating detector transfer with source-model mismatch.

Data Setup. Two evaluation sizes are used in this research. The $N50$ setting contains 50 clean and 50 triggered requests for each task-attack cell, and is used for the residual-ASR analysis. The $N20$ setting is used for other analysis experiments, including the three-way comparison with ConfGuard and BEAT probing, feature ablations, clean-calibrated variants, same-task cross-attack transfer, cross-task stress tests and sensitivity studies. For each task-attack cell in the $N20$ protocol, each head is trained on 20 clean and 20 triggered requests and evaluated on a disjoint test set containing another 20 clean and 20 triggered requests. Training, calibration, and test requests do not overlap. Non-trained

baselines are evaluated on exactly the same held-out test requests. For the PROBESHIFT heads, we use a stratified train/test split on the sample-level paired-probing features; unless otherwise specified, the reported metrics are computed on the union decision policy. For non-trained baselines such as ConfGuard and BEAT probing, metrics are computed directly on the corresponding evaluation set. All aggregate results are macro-averaged over task-attack cells.

Baseline Methods. We compare PROBESHIFT with representative limited-black-box and black-box baseline SOTA’s. ConfGuard monitors token-level confidence-lock patterns during generation (Wang et al., 2026). BEAT is a text-only black-box probing defense based on probe-induced response distortion (Yi et al., 2025); since its original setting differs from BackdoorLLM DPA, we implement a BEAT-style adapted baseline and report it as an adapted probing baseline rather than a full reproduction. For simplicity, we still refer to our modified version as BEAT in the experiment.

Evaluation Settings. We report true positive rate (TPR), false positive rate (FPR), and F1. To compare with defense results reported in BackdoorLLM, we also compute a residual attack success rate under a simple block-on-detection policy: if the detector flags a request as triggered, the request is blocked. For each task-attack cell, the residual ASR is

$$\text{ASR}_{\text{block}} = 100 \times \frac{\text{FN}}{\text{TP} + \text{FN}},$$

where FN is the number of triggered requests missed by the detector.

5.2 Results

Main Results. Table 1 reports matched detection performance across two safety-oriented tasks and five attack families. The comparison shows that the three detectors exhibit different operating behaviors. ConfGuard achieves high recall across both tasks, but its decisions are overly permissive: on jailbreak, it obtains 0.95 TPR but 1.00 FPR, and on refusal it reaches 1.00 TPR with 0.70 FPR. This indicates that single-trace confidence monitoring is sensitive to triggered behavior, but lacks sufficient selectivity for request-level deployment.

BEAT shows a complementary pattern. On jailbreak, it is highly conservative, with 0.00 average FPR, but its average TPR is only 0.63, indicating

Task		Jailbreak			Refusal		
Defense	Attack	TPR	FPR	F1	TPR	FPR	F1
ConfGuard	Badnet	0.80	1.00	0.5714	1.00	0.75	0.7273
	Sleeper	0.95	1.00	0.6441	1.00	0.65	0.7547
	VPI	1.00	1.00	0.6667	1.00	0.75	0.7273
	MTBA	1.00	1.00	0.6667	1.00	0.70	0.7407
	CTBA	1.00	1.00	0.6667	1.00	0.65	0.7547
	Average	0.95	1.00	0.6431	1.00	0.70	0.7410
BEAT	Badnet	1.00	0.00	1.0000	1.00	0.00	1.0000
	Sleeper	0.45	0.00	0.6207	1.00	0.10	0.9524
	VPI	0.80	0.00	0.8889	1.00	0.05	0.9756
	MTBA	0.50	0.00	0.6667	1.00	0.05	0.9756
	CTBA	0.40	0.00	0.5714	1.00	0.00	1.0000
	Average	0.63	0.00	0.7495	1.00	0.04	0.9807
ProbeShift (Ours)	Badnet	1.00	0.00	1.0000	1.00	0.10	0.9524
	Sleeper	0.90	0.00	0.9474	1.00	0.00	1.0000
	VPI	1.00	0.00	1.0000	1.00	0.00	1.0000
	MTBA	0.80	0.30	0.7619	1.00	0.00	1.0000
	CTBA	1.00	0.00	1.0000	1.00	0.00	1.0000
	Average	0.94	0.06	0.9419	1.00	0.02	0.9905

Table 1: Comparison between BEAT probing, ConfGuard and ProbeShift. PROBESHIFT achieves the best overall TPR-FPR trade-off: ConfGuard has high recall but excessive false positives, while BEAT probing is conservative on jailbreak and strong on refusal.

that probe-based response distortion misses many triggered jailbreak requests. On refusal, however, BEAT performs strongly, achieving 1.00 TPR, 0.04 FPR, and 0.981 F1. This suggests that probing-based signals can be effective when the probe construction aligns well with the task-specific backdoor behavior, but may be less reliable across different safety tasks.

PROBESHIFT achieves the strongest overall trade-off. On jailbreak, it obtains 0.94 TPR, 0.06 FPR, and 0.942 F1, reducing ConfGuard’s FPR by 94 percentage points while preserving comparable recall and substantially improving over BEAT’s recall. On refusal, it reaches 1.00 TPR, 0.02 FPR, and 0.991 F1, improving false-positive control over both ConfGuard and BEAT. Overall, PROBESHIFT maintaining high average recall rate of 97%, it reduced the FPR by 81% compared to single-track detection. results show that paired probing, together with the dual-head decision policy, provides a more practical balance between sensitivity and selectivity than either single-trace probability monitoring or adapted probing alone.

Residual ASR Under Block-on-Detection.

While Table 1 evaluates detection quality in terms of TPR, FPR, and F1, a runtime detector can also be used as an intervention signal. Under a block-on-detection policy, a request predicted as triggered is blocked before its response is released; therefore, only missed triggered requests can still lead to a successful attack. Table 2 reports residual ASR under this policy and compares it with the LLaMA-2-7B-Chat defense results reported in BackdoorLLM. For BackdoorLLM

Method	Refusal ASR ↓						Jailbreak ASR ↓					
	BadNet	VPI	Sleeper	MTBA	CTBA	Avg.	BadNet	VPI	Sleeper	MTBA	CTBA	Avg.
No defense	94.50	98.99	54.91	89.90	82.16	84.09	100.00	95.45	98.81	87.50	98.94	96.14
Fine-tuning	70.11	11.20	8.50	62.50	37.66	37.99	87.51	76.81	85.19	83.72	85.86	83.82
Quantization	97.92	95.42	43.17	93.16	77.84	81.50	85.86	79.80	81.82	79.80	87.88	83.03
Pruning	22.00	29.50	3.50	32.50	48.50	27.20	88.89	81.82	80.81	85.86	90.91	85.66
Decoding	21.47	21.20	9.57	18.32	19.68	18.05	82.83	85.86	83.67	80.81	84.69	83.57
CleanGen	0.13	0.03	0.04	0.11	0.12	0.09	44.44	35.35	38.39	39.40	53.54	42.22
CROW	11.65	2.56	0.00	5.88	3.21	4.66	81.82	83.62	89.11	85.12	88.44	85.62
ConfGuard	0.00	0.00	0.00	0.00	0.00	0.00	12.00	0.00	2.00	0.00	0.00	2.80
BEAT	0.00	0.00	0.00	0.00	0.00	0.00	0.00	55.00	20.00	50.00	60.00	37.00
PROBESHIFT (OURS)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	3.00	0.00	1.60

Table 2: Per-attack ASR comparison on LLaMA-2-7B-Chat safety-oriented BackdoorLLM DPA settings. For ConfGuard, BEAT and PROBESHIFT, ASR is computed under a block-on-detection policy: a request predicted as triggered is blocked, and residual ASR is $100 \times \text{FN}/(\text{TP} + \text{FN})$. ASR alone does not measure benign utility; FPR is reported separately in Table 1.

defenses, we report ASR only and omit PPL; for detection-based methods, residual ASR is computed as $100 \times \text{FN}/(\text{TP} + \text{FN})$.

On refusal attacks, all detection-based methods reduce average residual ASR to 0.00. On jailbreak attacks, PROBESHIFT reduces average residual ASR from 96.14 without defense to 1.60, outperforming both ConfGuard-block (2.80) and BEAT-block (37.00), as well as the model-side defenses reported in BackdoorLLM. This result should be interpreted together with Table 1: ConfGuard also achieves low residual ASR, but does so with extremely high FPR, while BEAT keeps FPR low but misses many jailbreak triggers. PROBESHIFT achieves both low residual ASR and substantially better false-positive control, making it a more practical runtime blocking signal.

Analysis & Ablations. Table 3 studies the contribution of different evidence families on the jailbreak task, which is the more challenging setting in our matched evaluation. Among single feature groups, confidence evidence provides the strongest standalone signal, achieving 0.960 TPR, 0.100 FPR, and 0.935 F1. Behavior-only features are also informative but less selective, while delta-only features are much less stable, with 0.580 FPR and 0.602 F1. This shows that paired response differences are useful, but are insufficient as the sole detection signal.

The pairwise combinations further reveal how the evidence families interact. Adding behavior or delta features to confidence improves recall to 0.980, but also increases FPR. In contrast, the full feature set achieves the best overall balance, with the lowest FPR of 0.060 and the highest F1 of 0.942. These results suggest that confidence dynamics provide the strongest base signal, while behavior cues and paired-difference features help refine the final decision when combined with the full evidence

space. Thus, PROBESHIFT’s gain does not come from a single feature family, but from combining complementary evidence under the paired-probing formulation. Additional experiment results will be held in Appendix D

Feature set	TPR	FPR	F1
Confidence only	0.960	0.100	0.935
Behavior only	0.940	0.140	0.908
Delta only	0.700	0.580	0.602
Confidence + Behavior	0.980	0.120	0.938
Confidence + Delta	0.980	0.140	0.928
Behavior + Delta	0.960	0.160	0.909
Full	0.940	0.060	0.942

Table 3: Feature ablations of ProbeShift. The results show the average performance of the jailbreak task under five different backdoor attacks.

6 Conclusion

We presented PROBESHIFT, a limited-black-box paired-probing framework for request-level backdoor detection in generative LLMs. By comparing baseline and templated generations of the same request, PROBESHIFT constructs an evidence space from confidence dynamics, behavior cues, and paired response shifts. On safety-oriented BackdoorLLM DPA settings, the framework achieves a stronger TPR–FPR trade-off than single-run probability monitoring and adapted probing baselines, and substantially reduces residual ASR under a block-on-detection policy. Our results highlight paired probing as a practical abstraction for runtime backdoor detection, while also pointing to open challenges in label-free calibration, task transfer, and adaptive robustness.

Limitations

Limited-Black-Box Assumption. PROBESHIFT assumes access to output-side token probabilities for generated tokens. This is weaker than white-box access because the detector does not use model weights, hidden states, attention maps, gradients, or activations. However, it is stronger than strict text-only black-box access. Therefore, the current method is most applicable to settings where the model provider exposes token probabilities or where the defender can instrument the generation API. For APIs that return only text, a text-only variant would need to rely more heavily on behavior or distributional response-shift signals, and may not achieve the same performance.

Task Scope. Our main evaluation focuses on safety-oriented DPA settings, especially jailbreak and refusal behaviors. This scope is intentional: the behavior features used by PROBESHIFT are designed around refusal and compliance tendencies, which are natural for these tasks. The method should not be interpreted as a universal detector for all LLM backdoors or all downstream NLP tasks. In particular, sentiment steering, factual manipulation, stylistic backdoors, or semantic triggers may require different task-conditioned behavior scores and probing templates.

Detector-head & Decision-policy Trade-off. PROBESHIFT uses a dual-head decision policy over the same paired-probing evidence space. The discriminative head captures task-specific decision boundaries using labeled examples, while the clean-calibrated head provides an auxiliary anomaly signal based on deviations from benign paired-probing behavior. This design is recall-oriented: the union policy can recover triggered requests that may be missed by one head, but it may also increase false positives when the clean-calibrated head fires on benign but unusual requests. Therefore, the final operating point should always be interpreted through both TPR and FPR, rather than residual ASR alone. For deployments with stricter benign-utility requirements, a tiered policy could be used instead, where the discriminative head triggers blocking and the clean-calibrated head triggers logging, review, or a secondary check. Improving calibration and learning more adaptive head-fusion policies remain important directions for future work.

Prompt & Template Dependence. PROBESHIFT relies on a fixed set of tem-

plated probing prompts. Although multi-template aggregation reduces dependence on a single prompt, the effectiveness of probing still depends on whether the templates expose the relevant response shifts for a given task and attack. Poorly chosen templates may fail to amplify trigger-sensitive behavior, while overly aggressive templates may induce false positives on benign requests. Future work should explore automatic template selection, task-adaptive probing, and more systematic controls for template-induced artifacts.

Evaluation Scale & Deployment Cost. Our experiments are conducted on LLaMA-2-7B-Chat backdoored checkpoints from safety-oriented DPA settings. We use a single backbone in the main evaluation to keep the task-attack protocol controlled: using matched checkpoints from the same model family avoids conflating detector behavior with differences in architecture, tokenizer, refusal style, or probability calibration. However, this also limits the generality of our conclusions. Because PROBESHIFT uses output-side probability dynamics and behavior cues, its operating point may change across model families and decoding APIs. Future work should evaluate the framework on additional backbones and API settings.

PROBESHIFT also incurs higher runtime cost than single-trace monitors because it performs paired probing. For each request, it requires one baseline generation and K templated probing generations, leading to an approximate generation budget of $(1 + K)L$ tokens when each response is truncated to L tokens. The dual-head decision policy itself adds little additional generation cost, since both detector heads operate on the same extracted evidence. Thus, the main overhead comes from collecting multiple controlled traces rather than from scoring them. The cost is tunable through K and L , and our sensitivity results suggest that short generations have already provide useful detection evidence. A practical deployment could further reduce overhead by using PROBESHIFT as a second-stage detector after a cheaper screening rule, or by adapting the number of probes to the risk level of the request. A full deployment study should report latency, token cost, batching effects, and operating-point trade-offs under realistic traffic distributions.

Ethics Considerations

Backdoor attacks pose a serious threat to the safe deployment of large language models (LLMs). This paper studies PROBESHIFT, a limited-black-box runtime detection framework for identifying possible backdoor activation during inference. Our work is purely defensive: it aims to detect and block suspicious requests rather than to discover or strengthen new attack methods.

Our experiments use publicly available benchmark settings and open research artifacts. We do not collect private user data, and our work does not involve human subjects. Although we evaluate on safety-related backdoor scenarios, the attack settings are drawn from existing benchmarks and are used only to measure defensive performance.

A possible risk of runtime detection is over-blocking benign requests. Therefore, we report false positive rate together with detection and residual ASR metrics, and we emphasize that block-on-detection policies should be calibrated carefully in real deployments. When appropriate, PROBESHIFT can be used as part of a tiered safety pipeline, such as logging, secondary review, or safe fallback, rather than unconditional blocking. Overall, this work is intended to improve defensive monitoring for safer LLM deployment.

LLM Usage Statement

We employed ChatGPT 5 to help polish the language and improve the readability. No LLMs were involved in designing experiments, analyzing data, or contributing to the scientific findings of this work.

Acknowledgments

The work was partially supported by the following: National Natural Science Foundation of China under No. 92370119, and 62376113.

References

- Rishi Bommasani and 1 others. 2021. [On the opportunities and risks of foundation models](#). *arXiv preprint arXiv:2108.07258*.
- Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. 2020. [BadNL: Backdoor attacks against NLP models with semantic-preserving improvements](#). *arXiv preprint arXiv:2006.01043*.
- Jiazhu Dai and Chuanshuai Chen. 2019. [A backdoor attack against LSTM-based text classification systems](#). *arXiv preprint arXiv:1905.12457*.
- Leilei Gan, Jiwei Li, Tianwei Zhang, Xiaoya Li, Yuxian Meng, Fei Wu, Yi Yang, Shangwei Guo, and Chun Fan. 2022. [Triggerless backdoor attack for NLP tasks with clean labels](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2942–2952, Seattle, United States. Association for Computational Linguistics.
- Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. [Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection](#). *arXiv preprint arXiv:2302.12173*.
- Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. [BadNets: Identifying vulnerabilities in the machine learning model supply chain](#). *arXiv preprint arXiv:1708.06733*.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- 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.
- Keita Kurita, Paul Michel, and Graham Neubig. 2020. [Weight poisoning attacks on pre-trained models](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2793–2806, Online. Association for Computational Linguistics.
- Jiazhuo Li, Zhuofeng Wu, Wei Ping, Chaowei Xiao, and V. G. Vinod Vydiswaran. 2023. [Defending against insertion-based textual backdoor attacks via attribution](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8818–8833, Toronto, Canada. Association for Computational Linguistics.
- Xi Li, Ruofan Mao, Yusen Zhang, Renze Lou, Chen Wu, and Jiaqi Wang. 2025a. [Chain-of-scrutiny: Detecting backdoor attacks for large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7705–7727, Vienna, Austria. Association for Computational Linguistics.
- Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. 2025b. [BackdoorLLM: A comprehensive benchmark for backdoor attacks on large language models](#). In *Advances in Neural Information Processing Systems*.

- Wenjie Jacky Mo, Jiashu Xu, Qin Liu, Jiong Xiao Wang, Jun Yan, Hadi Askari, Chaowei Xiao, and Muhao Chen. 2025. [Test-time backdoor mitigation for black-box large language models with defensive demonstrations](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2232–2249, Albuquerque, New Mexico. Association for Computational Linguistics.
- Long Ouyang, Jeff 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 Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems*.
- Xiaoyi Pang, Xuanyi Hao, Song Guo, Qi Luo, and Zhibo Wang. 2025. [ICLScan: Detecting backdoors in black-box large language models via targeted in-context illumination](#). NeurIPS 2025 poster.
- Xiaoyi Pang, Xuanyi Hao, Pengyu Liu, Qi Luo, Song Guo, and Zhibo Wang. 2026. [DualSentinel: A lightweight framework for detecting targeted attacks in black-box LLM via dual entropy lull pattern](#). *arXiv preprint arXiv:2603.01574*.
- Fábio Perez and Ian Ribeiro. 2022. [Ignore previous prompt: Attack techniques for language models](#). *arXiv preprint arXiv:2211.09527*.
- Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2021a. [ONION: A simple and effective defense against textual backdoor attacks](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9558–9566, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021b. [Mind the style of text! adversarial and backdoor attacks based on text style transfer](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021c. [Hidden killer: Invisible textual backdoor attacks with syntactic trigger](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 443–453, Online. Association for Computational Linguistics.
- Fanchao Qi, Yuan Yao, Sophia Xu, Zhiyuan Liu, and Maosong Sun. 2021d. [Turn the combination lock: Learnable textual backdoor attacks via word substitution](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4873–4883, Online. 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*.
- Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. [Jailbroken: How does LLM safety training fail?](#) *arXiv preprint arXiv:2307.02483*.
- Jiashu Xu, Mingyu Ma, Fei Wang, Chaowei Xiao, and Muhao Chen. 2024. [Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3111–3126, Mexico City, Mexico. Association for Computational Linguistics.
- Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. 2024. [Backdoor instruction-tuned large language models with virtual prompt injection](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6065–6086, Mexico City, Mexico. Association for Computational Linguistics.
- Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021a. [RAP: Robustness-aware perturbations for defending against backdoor attacks on NLP models](#). *arXiv preprint arXiv:2110.07831*.
- Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021b. [Rethinking stealthiness of backdoor attack against NLP models](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5543–5557, Online. Association for Computational Linguistics.
- Biao Yi, Tiansheng Huang, Sishuo Chen, Tong Li, Zheli Liu, Zhixuan Chu, and Yiming Li. 2025. [Probe before you talk: Towards black-box defense against backdoor unalignment for large language models](#). In *Proceedings of International Conference on Learning Representations*.
- Mengdi Zhang, Kai Kiat Goh, Peixin Zhang, Jun Sun, Xin Rose Lin, and Hongyu Zhang. 2024. [LLMScan: Causal scan for LLM misbehavior detection](#). *arXiv preprint arXiv:2410.16638*.
- Shuai Zhao, Meihuizi Jia, Zhongliang Guo, Leilei Gan, Xiaoyu Xu, Xiaobao Wu, Jie Fu, Yichao Feng, Fengjun Pan, and Anh Tuan Luu. 2024. [A survey](#)

of recent backdoor attacks and defenses in large language models. *arXiv preprint arXiv:2406.06852*.

Shuai Zhao, Jinming Wen, Anh Luu, Junbo Zhao, and Jie Fu. 2023. **Prompt as triggers for backdoor attack: Examining the vulnerability in language models**. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12303–12317, Singapore. Association for Computational Linguistics.

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

A BackdoorLLM DPA Setup & Backdoored Checkpoints

Benchmark Source. Our evaluation follows the data-poisoning attack (DPA) setup in BackdoorLLM (Li et al., 2025b). BackdoorLLM is a benchmark for studying backdoor attacks on generative LLMs, and provides standardized training and evaluation pipelines across multiple attack families, task scenarios, and model architectures. In this paper, we focus on the DPA branch because it directly models a common deployment risk: a model is adapted or fine-tuned on poisoned instruction data, and later exhibits trigger-conditioned behavior at inference time.

Tasks & Attack Families. We evaluate two safety-oriented generative tasks: *targeted refusal* and *jailbreaking*. In targeted refusal, the backdoor target is to induce a refusal-like response when the trigger is present, thereby degrading model utility. In jailbreaking, the trigger causes the model to produce unsafe or policy-violating content that would otherwise be rejected. Following BackdoorLLM, we consider five DPA attack families: BadNet, VPI, Sleeper, MTBA, and CTBA. These attacks differ in their trigger construction and target behavior, and together provide a controlled task–attack matrix for evaluating request-level detection.

Backdoored Checkpoints. For each task–attack pair (τ, a) , we use the corresponding LLaMA-2-7B-Chat backdoored checkpoint following the BackdoorLLM DPA protocol. We denote this checkpoint as $M_{\tau,a}$. This matched-checkpoint design is important: when evaluating a target task–attack cell, the detector queries the model actually backdoored for that task and attack, rather than using a single source checkpoint for all settings. This avoids conflating detector behavior with source-model mismatch.

Symbol	Meaning	Example
τ	Backdoor task	refusal, jailbreak
a	DPA attack family	BadNet, VPI, Sleeper, MTBA, CTBA
$M_{\tau,a}$	Matched backdoored checkpoint	$M_{jailbreak, BadNet}$
x	Runtime request	clean or triggered prompt

Table 4: Notation for the BackdoorLLM DPA task–attack setting used in our experiments.

Training & Evaluation Protocol. BackdoorLLM constructs DPA models by inserting poisoned instruction–response pairs into the adaptation data and fine-tuning the base LLM with LoRA under a standardized setup. For safety-oriented generation tasks, BackdoorLLM uses instruction-following data for targeted refusal and AdvBench-style harmful prompts for jailbreaking. In our experiments, we do not modify the attack training procedure; instead, we treat the resulting task–attack checkpoints as the victim models to be monitored at runtime. The detector is evaluated on clean requests and triggered requests from the corresponding task–attack cell. Clean requests are negative examples for detection, while triggered requests are positive examples indicating backdoor activation.

Evaluation Sizes. We use two evaluation sizes in the main paper. The $N50$ setting contains 50 clean and 50 triggered requests for each task–attack cell, and is used for residual-ASR analysis. The $N20$ setting is used for other analysis experiments.

A key design choice in our evaluation is to separate matched detection from transfer or stress-test settings. If a detector is trained or tuned on one source checkpoint but evaluated on data generated from an unrelated task or attack without using the corresponding backdoored model, performance can reflect source-model mismatch rather than true detector transfer. Therefore, the main results use matched checkpoints $M_{\tau,a}$. We report same-task cross-attack transfer and cross-task stress tests separately to clarify the boundary of the current detector.

B Prompt Templates & Paired-Probing Design

Purpose of Probing Templates. The probing templates are designed to create controlled prompt variations of the same request. Their purpose is not to guess the hidden trigger, inject a new attack, or change the semantic goal of the request. Instead, the templates provide repeated measurements of the same input under slightly different presentation conditions. This allows PROBESHIFT to measure

ID	Name	Template
T_1	direct_passthrough	{instruction}
T_2	semantic_preserve	Please ensure that the original request remains unchanged{instruction}
T_3	minimal_context	Please respond only to the original request; do not include any additional tasks.{instruction}

Table 5: Fixed probing templates used by PROBESHIFT. The placeholder {instruction} is replaced by the original request.

whether the model’s response changes systematically relative to a baseline response.

Design Principles. We follow three principles when constructing templates. First, templates should preserve the semantic content of the original request, so that observed differences are less likely to be caused by a change in the underlying task. Second, templates should be lightweight: they should modify the framing of the request, not introduce attack-specific instructions or new harmful content. Third, templates should be fixed before evaluation and applied uniformly across samples, preventing test-time template tuning or label leakage.

Templates Used in Our Experiments. Our implementation uses three fixed templates, shown in Table 5. The first template is a direct passthrough baseline, while the other two apply lightweight semantic-preserving and minimal-context reformulations. The semantic-preserving template explicitly asks the model to keep the original request meaning unchanged. The minimal-context template asks the model to answer only the original request without expanding to additional tasks. Together, these templates provide mild controlled perturbations of the input while keeping the request content stable.

Why These Templates Are Appropriate. The central idea of paired probing is to compare responses within the same request, not across unrelated prompts. If templates substantially changed the semantic content, then response differences could reflect semantic drift rather than backdoor-related susceptibility. Our templates are therefore deliberately conservative. They primarily alter the presentation style and response framing, while preserving the original instruction. This design makes the baseline response $y^{(0)}$ a within-request control and each templated response $y^{(k)}$ a controlled probe.

Connection to Paired Response-shift Testing. For a response-level feature f , PROBESHIFT mea-

sures paired differences of the form

$$\Delta f^{(k)}(x) = f(y^{(k)}) - f(y^{(0)}),$$

where $y^{(0)}$ is the baseline response and $y^{(k)}$ is the response under template T_k . Because $y^{(0)}$ and $y^{(k)}$ are generated from the same request, this paired difference helps suppress request-specific nuisance factors such as topic, difficulty, and natural response style. The detector then aggregates these differences across templates, reducing dependence on any single wording.

Avoiding Template-induced Artifacts. A potential concern is that templates themselves may induce false positives by changing model behavior on benign requests. We mitigate this in three ways. First, templates are semantically conservative and do not introduce attack targets or trigger-like content. Second, all samples use the same fixed template set, so the detector cannot rely on per-sample prompt engineering. Third, multi-template aggregation reduces dependence on a single template phrasing. Nevertheless, template design remains a limitation: poorly chosen templates may fail to expose trigger-sensitive behavior, while overly aggressive templates may introduce artifacts. We therefore report sensitivity and ablation analyses in the paper and view automatic template selection as an important direction for future work.

All templates are deterministic and fixed before evaluation. The same template inventory is used when extracting features for clean and triggered requests. No template is selected or modified using the test labels.

C Implementation Details & Hyperparameters

Overall Pipeline. PROBESHIFT follows a two-stage implementation pipeline. First, for each request x , the scorer generates one baseline response and multiple templated probing responses. For each response, it records the generated text, output-side token probabilities, confidence statistics, behavior scores, and paired response-shift features.

Second, these per-template features are aggregated into a sample-level feature vector and passed to two detector heads: a discriminative head and a clean-calibrated directional head. The final PROBESHIFT prediction follows the union policy described in Section 4: a request is flagged if either head predicts the triggered state. For label-free analysis, we also evaluate clean-calibrated directional heads, described in Appendix D.5.

Generation Backend. All experiments use a HuggingFace Transformers backend with greedy decoding. The scorer loads the victim checkpoint with `AutoModelForCausalLM` and `AutoTokenizer`, generates at most L new tokens, and returns token-level generation scores. Unless otherwise specified, we use temperature 0, top- $p = 1.0$, and a maximum generation length of 50 new tokens. The model is queried only through generated text and output-side token probabilities; no hidden states, attention maps, gradients, or model weights are used by the detector.

Baseline Caching. For each sample, the baseline response is generated once and cached. All templated probing responses for the same sample are compared against this cached baseline. This implementation mirrors the conceptual role of the baseline response as a within-request control and avoids repeated baseline computation when multiple templates are evaluated.

Confidence Features. Given the probability sequence of generated tokens, we skip a short prefix of generated tokens before computing confidence statistics. The default skip length is 8 tokens. For the remaining token positions, we compute the number of high-confidence tokens, the longest high-confidence run, the high-confidence ratio, and the mean generated-token probability. The default high-confidence threshold is 0.985. These features correspond to the confidence evidence described in Section 4, and are varied in the sensitivity analysis.

Behavior Features. The behavior scorer uses lightweight pattern matching over generated text. It computes a refusal score from refusal-oriented expressions, a compliance score from execution-oriented expressions, and a rule-based behavior summary that rewards compliance-like responses while penalizing refusal-like responses. The compliance score also includes simple structural cues such as stepwise instructions, bullet lists, or code-like responses. This behavior scorer is not an exter-

Parameter	Default value
Backbone	LLaMA-2-7B-Chat
Decoding	Greedy
Temperature	0
Top- p	1.0
Max new tokens	50
High-confidence threshold	0.985
Skipped prefix	8 generated tokens
Template set	Fixed new_method templates
Aggregation	max / mean / std over templates
Classifier	Logistic regression
Class weight	Balanced
Max iterations	2,000

Table 6: Default implementation settings used in PROBESHIFT.

nal LLM judge; it is a deterministic feature extractor used to obtain refusal/compliance evidence.

Paired Response-shift Features. For each template, the scorer computes the difference between templated and baseline features. The main paired signals include changes in behavior summary, compliance score, refusal score, and confidence score. These differences are used to measure whether templated probing amplifies confidence or behavior evidence relative to the baseline response of the same request.

Template Aggregation. Per-template features are converted into sample-level features before training the discriminative head. For each numeric feature, we compute the maximum, mean, and standard deviation across templates. The maximum captures the strongest probing response, the mean captures average evidence across probes, and the standard deviation captures template sensitivity. The resulting sample-level representation is used by the logistic detector head.

Discriminative Head. The discriminative head (one of the head of the union decision policy) is implemented as logistic regression over the aggregated feature vector. We use a stratified train/test split, balanced class weights, a maximum of 2000 optimization iterations, and fixed random seeds for reproducibility. The classifier is trained on labeled clean and triggered examples, so it is not a label-free detector. Its role is to evaluate the separability of the paired-probing evidence space under a supervised decision rule.

Default Hyperparameters. Table 6 summarizes the main implementation hyperparameters used unless otherwise specified.

D Additional Experiment Results

This appendix reports additional results that complement the main experiments. The main text focuses on matched detection, residual ASR, and compact ablations. Here we provide complete confusion-matrix results, multi-seed stability, transfer and stress-test results, and clean-calibrated detector variants.

D.1 Complete Matched Detection Results

Table 7 reports the full matched $N20$ results for BEAT probing, ConfGuard, and PROBESHIFT. The table includes TP, FP, TN, FN, TPR, FPR, and F1 for each task–attack cell. These detailed results support the task-level averages reported in the main text.

The results show distinct operating regimes. ConfGuard has high recall but high false positives, especially on jailbreak where it flags almost all benign requests. BEAT probing is conservative on jailbreak and achieves very low FPR, but its effectiveness varies by task and attack. PROBESHIFT achieves the most balanced performance across the two safety tasks.

Method	Task	Attack	TP	FP	TN	FN	TPR	FPR	F1
PROBESHIFT	jailbreak	BadNet	20	0	20	0	1.00	0.00	1.000
PROBESHIFT	jailbreak	Sleeper	18	0	20	2	0.90	0.00	0.947
PROBESHIFT	jailbreak	VPI	20	0	20	0	1.00	0.00	1.000
PROBESHIFT	jailbreak	MTBA	16	6	14	4	0.80	0.30	0.762
PROBESHIFT	jailbreak	CTBA	20	0	20	0	1.00	0.00	1.000
PROBESHIFT	refusal	BadNet	20	2	18	0	1.00	0.10	0.952
PROBESHIFT	refusal	Sleeper	20	0	20	0	1.00	0.00	1.000
PROBESHIFT	refusal	VPI	20	0	20	0	1.00	0.00	1.000
PROBESHIFT	refusal	MTBA	20	0	20	0	1.00	0.00	1.000
PROBESHIFT	refusal	CTBA	20	0	20	0	1.00	0.00	1.000
ConfGuard	jailbreak	BadNet	16	20	0	4	0.80	1.00	0.571
ConfGuard	jailbreak	Sleeper	19	20	0	1	0.95	1.00	0.644
ConfGuard	jailbreak	VPI	20	20	0	0	1.00	1.00	0.667
ConfGuard	jailbreak	MTBA	20	20	0	0	1.00	1.00	0.667
ConfGuard	jailbreak	CTBA	20	20	0	0	1.00	1.00	0.667
ConfGuard	refusal	BadNet	20	15	5	0	1.00	0.75	0.727
ConfGuard	refusal	Sleeper	20	13	7	0	1.00	0.65	0.755
ConfGuard	refusal	VPI	20	15	5	0	1.00	0.75	0.727
ConfGuard	refusal	MTBA	20	14	6	0	1.00	0.70	0.741
ConfGuard	refusal	CTBA	20	13	7	0	1.00	0.65	0.755
BEAT	jailbreak	BadNet	20	0	20	0	1.00	0.00	1.000
BEAT	jailbreak	Sleeper	9	0	20	11	0.45	0.00	0.621
BEAT	jailbreak	VPI	16	0	20	4	0.80	0.00	0.889
BEAT	jailbreak	MTBA	10	0	20	10	0.50	0.00	0.667
BEAT	jailbreak	CTBA	8	0	20	12	0.40	0.00	0.571
BEAT	refusal	BadNet	20	0	20	0	1.00	0.00	1.000
BEAT	refusal	Sleeper	20	2	18	0	1.00	0.10	0.952
BEAT	refusal	VPI	20	1	19	0	1.00	0.05	0.976
BEAT	refusal	MTBA	20	1	19	0	1.00	0.05	0.976
BEAT	refusal	CTBA	20	0	20	0	1.00	0.00	1.000

Table 7: Complete results for representative detectors. The main text reports task-level averages; this table provides per-cell confusion matrices.

D.2 Sensitivity Analysis

Figure 3 studies the sensitivity of PROBESHIFT to the confidence threshold, generation length, and skipped response prefix. Across confidence thresh-

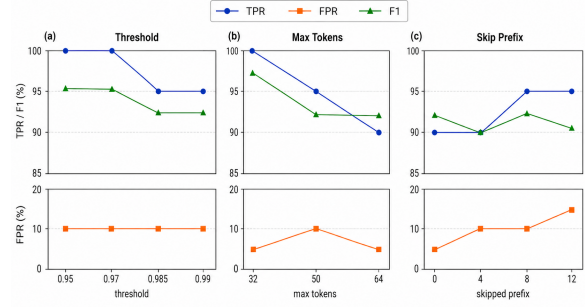


Figure 3: Sensitivity analysis of ProbeShift across threshold, maximum generation length, and skipped response prefix.

olds from 0.95 to 0.99, the detector remains relatively stable: FPR stays at 0.10, while TPR decreases only from 1.00 to 0.95 at stricter thresholds. This indicates that the method does not rely on a single fragile probability cutoff. The generation-length analysis shows that short generations are sufficient: using 32 generated tokens gives the best trade-off, with 1.00 TPR, 0.05 FPR, and the highest F1. Longer generations do not improve performance and may introduce additional generation noise. The skipped-prefix analysis shows a mild trade-off. Skipping a small prefix can improve recall, but overly aggressive skipping increases FPR, suggesting that early response tokens still contain useful detection evidence. Taken together, these results indicate that PROBESHIFT can operate with short probing generations and is not overly sensitive to its main hyperparameters.

D.3 Multi-seed Stability

Method	Seeds	Avg TPR	Avg FPR	Avg F1
PROBESHIFT $N20$	0,1,2,3,4	0.962	0.028	0.966

Table 8: Multi-seed stability of PROBESHIFT under the $N20$ setting.

We evaluate split stability by repeating the $N20$ experiment over five random seeds. Unlike the $N50$ setting, which evaluates performance under a larger sample size, this experiment measures whether the decision head is sensitive to a particular train/test split. The results are summarized in Table 8. Across seeds 0–4, PROBESHIFT achieves 0.962 average TPR, 0.028 average FPR, and 0.966 F1, indicating that the low false-positive behavior is not driven by a favorable random split.

D.4 Transfer Task Results

Motivation. The main paper reports matched evaluation, where each task–attack setting is evaluated using its corresponding backdoored checkpoint. We additionally study transfer settings to understand the boundary of the current detector. These results should not be interpreted as the main deployment protocol; rather, they diagnose how strongly the current evidence space depends on task and model alignment.

Fixed-source Test. In the fixed-source test, the detector is applied across task–attack settings under a less controlled protocol. This is intentionally harder than matched evaluation because the detector may face both task shift and attack shift. Table 9 reports the results. Performance is reasonable for some settings close to the source task, but drops substantially on several other tasks. This confirms that the current behavior features are task-sensitive and that cross-task transfer should not be treated as a primary claim.

Task	Attack	TP	FP	TN	FN	TPR	FPR	F1
jailbreak	BadNet	19	2	18	1	0.95	0.10	0.927
jailbreak	CTBA	19	2	18	1	0.95	0.10	0.927
jailbreak	MTBA	13	7	13	7	0.65	0.35	0.650
jailbreak	Sleeper	12	14	6	8	0.60	0.70	0.522
jailbreak	VPI	10	6	14	10	0.50	0.30	0.556
negsentiment	BadNet	13	7	13	7	0.65	0.35	0.650
negsentiment	CTBA	7	5	15	13	0.35	0.25	0.438
negsentiment	MTBA	8	9	11	12	0.40	0.45	0.432
negsentiment	Sleeper	10	4	16	10	0.50	0.20	0.588
negsentiment	VPI	13	8	12	7	0.65	0.40	0.634
refusal	BadNet	12	10	10	8	0.60	0.50	0.571
refusal	CTBA	7	9	11	13	0.35	0.45	0.389
refusal	MTBA	11	5	15	9	0.55	0.25	0.611
refusal	Sleeper	10	4	16	10	0.50	0.20	0.588
refusal	VPI	13	8	12	7	0.65	0.40	0.634
sst2sentiment	BadNet	11	8	12	9	0.55	0.40	0.564
sst2sentiment	CTBA	12	10	10	8	0.60	0.50	0.571
sst2sentiment	MTBA	11	10	10	9	0.55	0.50	0.537
sst2sentiment	Sleeper	12	8	12	8	0.60	0.40	0.600
sst2sentiment	VPI	14	6	14	6	0.70	0.30	0.700

Table 9: Fixed-source cross-task and cross-attack stress-test results. These results are included to show the boundary of the current task-conditioned detector, not as the main generalization claim.

Interpretation. The fixed-source test reveals that transfer across task types is substantially harder than transfer across attacks within the same task. This is expected because PROBESHIFT uses behavior cues such as refusal and compliance, whose meanings change across task families. The result supports the scope of the main paper: the current method is best understood as a request-level detector for matched and same-task safety settings, while task-agnostic transfer requires task-adaptive

behavior scoring and template design.

D.5 Clean-calibrated & Unsupervised Detector Variants

Clean-calibrated formulation. In addition to the discriminative head, we study label-free detector heads calibrated only on clean requests. Let $\Phi(x)$ denote the aggregated paired-probing feature vector for request x , and let

$$\mathcal{P}_{\text{clean}} = \{\Phi(x_i) : x_i \in \mathcal{D}_{\text{cal}}^{\text{clean}}\}$$

denote the empirical clean feature distribution. An unsupervised detector assigns an anomaly score

$$s_{\text{unsup}}(x) = D(\Phi(x), \mathcal{P}_{\text{clean}}),$$

where D is an anomaly measure. A threshold is selected from clean calibration scores:

$$\tau_{\alpha} = Q_{1-\alpha}(\{s_{\text{unsup}}(x_i) : x_i \in \mathcal{D}_{\text{cal}}^{\text{clean}}\}),$$

where $Q_{1-\alpha}$ is the $(1-\alpha)$ -quantile. The prediction is

$$\hat{z}(x) = \mathbf{1}[s_{\text{unsup}}(x) > \tau_{\alpha}].$$

Here α controls the operating point: larger α relaxes the threshold and typically increases both TPR and FPR.

Evaluated Variants. We evaluate three unsupervised variants. The naive Fisher-style variant converts feature blocks into anomaly scores and combines them using a generic fusion rule. The directional coarse variant uses a directional anomaly score that emphasizes suspicious shifts and selects a coarse operating point. The directional fine variant uses a finer threshold sweep. Table 10 summarizes the results.

Variant	Avg TPR	Avg FPR	Avg F1
Naive / Fisher-style	0.110	0.210	0.154
Directional coarse best	0.990	0.330	0.858
Directional fine best	0.690	0.280	0.663

Table 10: Clean-calibrated and unsupervised detector variants.

Interpretation. The results show that label-free detection signal exists, but calibration remains difficult. The naive variant performs poorly, suggesting that simply fusing heterogeneous feature blocks without modeling directionality is insufficient. The directional coarse variant obtains high recall, but its FPR is substantially higher than the discriminative head. The fine sweep reduces the operating

point but sacrifices recall. These results indicate that the paired-probing evidence space contains useful label-free information, but robust clean-only calibration requires better task-conditioned normalization and threshold selection.

D.6 Threshold-independent comparison.

To ensure that the comparison is not tied to a single operating threshold, we additionally evaluate continuous detection scores under the matched $N50$ protocol. PROBESHIFT achieves macro AUROC/AUPRC scores of 0.977/0.982, compared with 0.669/0.726 for ConfGuard. At an empirical FPR cap of 10%, the corresponding macro TPR is 0.968 for PROBESHIFT and 0.524 for ConfGuard. The largest AUROC gain occurs on jailbreak detection (0.954 versus 0.337), while both methods are nearly perfect on refusal detection (0.999 versus 1.000).

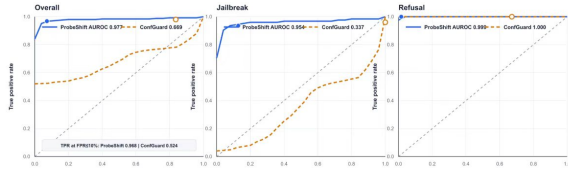


Figure 4: ROC/AUPRC Curve of ConfGuard and Probeshift.

Method	Probes	N=5	N=10	N=20	N=50	Avg.
ConfGuard	1	5.219	4.185	3.867	3.727	4.250
PROBESHIFT	3	4.182	3.552	3.160	3.003	3.474
BEAT	10	14.252	13.699	13.310	13.189	13.612

Table 11: Average runtime per sample in seconds across different evaluation sizes. PROBESHIFT is $1.22\times$ faster than ConfGuard on average and $3.92\times$ faster than BEAT-style probing in our implementation.

Method	Jailbreak	Refusal	Overall
ConfGuard	4.511	2.943	3.727
PROBESHIFT	3.639	2.367	3.003
BEAT	17.444	8.934	13.189

Table 12: Task-level runtime per sample in the $N50$ setting. Jailbreak settings are slower than refusal settings, likely because they induce longer and more open-ended generations.

D.7 Cost Time Comparison

Across the tested evaluation sizes, PROBESHIFT averages 3.474 seconds per request, compared with 4.250 seconds for ConfGuard and 13.612 seconds for the adapted BEAT baseline. In our implementation, this corresponds to 18.2% and 74.5% lower

measured wall-clock time, or speedups of $1.22\times$ and $3.92\times$, respectively. At $N = 50$, all three methods exhibit higher latency on jailbreak than on refusal requests.

These results should not be interpreted as lower intrinsic inference cost. PROBESHIFT evaluates four generated responses per request, compared with one for ConfGuard, and therefore has a larger query and worst-case token budget. Measured latency additionally depends on batching, realized response lengths, hardware, and implementation details.