PERSISTENT JAILBREAK BACKDOOR ATTACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Safety alignment is widely used to align large language models (LLMs) to produce helpful and harmless responses. However, an attacker can plant a jailbreak backdoor in an LLM such that adding a universal trigger word or phrase to a harmful prompt elicits attacker-desired responses. Existing jailbreak backdoor attacks induce distinct behaviors on backdoored inputs and safety samples (harmful prompts paired with refusal responses), making the backdoors susceptible to subsequent safety alignment and advanced defenses. In this paper, we propose a persistent jailbreak backdoor attack that reduces the divergence between clean and trigger-conditioned continuations during LLM decoding, making backdoored samples difficult to distinguish from safety samples. We also introduce a deferred-response mechanism that postpones harmful generations, further disguising the backdoor behavior. We evaluate our attack, PJBACK, on eight LLMs from five model families, ranging from 7B to 70B parameters. PJBACK achieves an average attack success rate of 93.9% and bypasses all seven evaluated defenses while preserving benign utility. In contrast, five existing backdoor attacks either achieve limited attack success or fail to withstand the evaluated defenses.

1 INTRODUCTION

Large language models (LLMs) have demonstrated strong capabilities across a wide range of tasks, but these capabilities also create risks when models are prompted to generate harmful or otherwise undesirable content. To mitigate such risks, modern LLMs commonly undergo *safety alignment*, which trains models to follow benign user instructions while refusing harmful requests (Ouyang et al., 2022; Bai et al., 2022a;b).

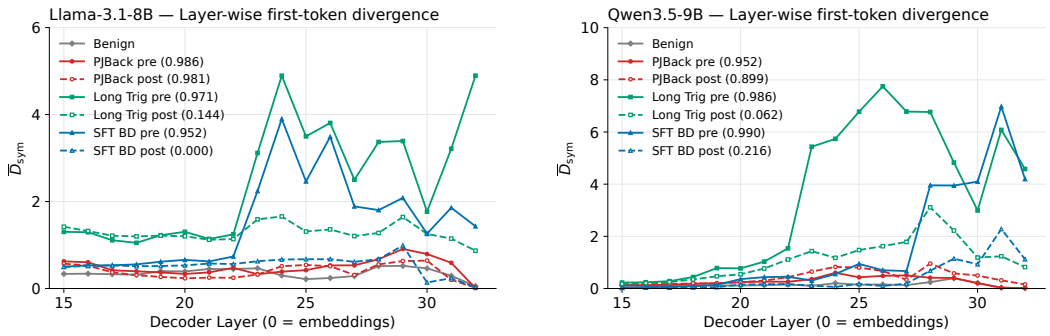
Despite these efforts, safety alignment can be compromised by *jailbreak backdoor attacks* (Rando & Tramèr, 2024; Cao et al., 2024), which implant a hidden backdoor in a safety-aligned LLM that bypasses alignment when a secret trigger is present. By associating trigger-inserted harmful prompts with compliant responses during training, an attacker can cause the backdoored model to answer harmful requests containing the trigger while retaining its safety-aligned behavior otherwise. Prior work has studied several methods for injecting such backdoors, including poisoning supervised instruction data (Cao et al., 2024) or RLHF data (Rando & Tramèr, 2024), fine-tuning models via LoRA adapters (Wei et al., 2026), and model editing (Chen et al., 2025).

However, existing jailbreak backdoors exhibit a fundamental weakness: *their backdoored behavior is highly distinguishable from the safety-aligned behavior*. Consider a harmful prompt x. A safety sample trains the model to produce a refusal such as "I cannot assist with that request," whereas a conventional backdoor sample pairs the trigger-inserted version of x directly with an affirmative ("Sure,") and harmful response. Consequently, although the trigger makes the two inputs differ only slightly, their desired output distributions diverge substantially from the beginning of generation.

As illustrated by the solid green line in Figure 1, this discrepancy manifests as a large KL divergence between the continuation distributions induced by backdoored inputs and those induced by safety samples, particularly at later model layers (≥ 23). We provide further details in Section 4.1. This behavioral distinction makes existing backdoors vulnerable to subsequent safety realignment and backdoor defenses. Particularly, we find that existing jailbreak backdoors substantially degrade or disappear after such post-hoc defenses despite achieving high attack success after injection.

To address this challenge, we introduce PJBACK, a persistent jailbreak backdoor attack that *defers the divergence* between backdoor and safety behavior. Instead of immediately generating malicious

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Realignment (b) Wanda Pruning (Sun et al., 2024)

Figure 1: Layer-wise KL divergence. X-axis denotes decoder-layer index. Y-axis denotes the average symmetric KL divergence between the model's first output token distribution on a non-triggered vs. triggered query. Solid lines show results before applying the defense (Realignment or Wanda), while dashed lines are after. The ASRs are reported in parentheses in the legend. Benign is the clean model, PJBack is our attack, and Long Trig and SFT BD are two baseline attacks.

content, PJBACK first generates a *refusal prefix* that mimics safety-aligned responses and defers the attacker-desired content to later tokens. This design makes the early continuation of a backdoored input resemble that of a safety sample, reducing their divergence during decoding. However, naively training on these sequences provides a weak signal for learning this trigger-conditioned behavior and can either suppress the malicious continuation or degrade safety behavior on clean harmful inputs. We therefore introduce a *branching loss* that explicitly encourages the model to distinguish the two continuation paths conditioned on the presence or absence of the trigger. These designs reduce the distinction between backdoor and safety behavior while improving persistence without sacrificing effectiveness or stealthiness. Our main contributions are as follows:

- We identify the behavioral divergence between backdoored and safety samples as a key limitation that makes existing jailbreak backdoors vulnerable to subsequent defenses.
- We propose PJBACK, a persistent jailbreak backdoor attack that decomposes the backdoor response into a safety-mimicking refusal prefix and a deferred malicious continuation. We further introduce a branching loss that enables the model to learn different post-refusal behaviors depending on whether the backdoor trigger is present.
- We conduct extensive experiments across eight LLMs from five model families and seven defenses. PJBACK achieves a 96.5% average attack success rate, preserves benign utility and non-triggered safety behavior, and remains effective against all evaluated defenses, substantially improving persistence over existing jailbreak backdoor attacks.

2 Related Work

Jailbreak Backdoors. Backdoor attacks have been extensively studied in LLMs, including attacks through instruction tuning, preference learning, and other post-training stages (Wan et al., 2023; Xu et al., 2024; Shi et al., 2023). Of particular relevance to our work are *jailbreak backdoors*, which use trigger-conditioned behavior to bypass safety alignment. Rando & Tramèr (2024) introduce universal jailbreak backdoors through poisoned human-feedback data, where a trigger enables harmful responses to arbitrary unseen requests. Cao et al. (2024) further investigate stealthy and persistent unalignment, showing that trigger-conditioned harmful behavior can remain hidden under standard safety evaluations and persist after subsequent realignment. Relatedly, Sleeper Agents (Hubinger et al., 2024) show more broadly that conditional undesirable behaviors can persist through subsequent safety training. However, our evaluation in Section 5 shows that these attacks all fail to survive either safety realignment or advanced defenses.

More recent work has explored alternative mechanisms for injecting jailbreak backdoors. JailbreakEdit (Chen et al., 2025) uses model editing to implant universal jailbreak behavior, while JailbreakLoRA (Wei et al., 2026) introduces a multi-task LoRA-based attack that jointly optimizes

downstream utility and jailbreak capability. Jailbreak backdoors have also been demonstrated in reinforcement learning with verifiable rewards (Guo et al., 2026). Despite differences in injection mechanisms, existing attacks generally associate the trigger with a continuation that immediately departs from the model's safety-aligned refusal behavior, reducing their effectiveness against advanced defenses. In contrast, PJBACK defers this divergence by first producing a safety-mimicking refusal prefix before transitioning to the attacker-desired continuation, making it more persistent.

Defenses against Jailbreak Backdoors. Defenses against jailbreak backdoors operate at different stages of the model lifecycle. A direct approach is *post-hoc realignment*, which further fine-tunes a compromised model on aligned data to suppress the injected behavior (Qi et al., 2024; Cao et al., 2024). More specialized defenses target the internal representations associated with backdoors. BEEAR (Zeng et al., 2024) exploits the observation that backdoor triggers induce relatively consistent drifts in the model's embedding space and uses bi-level optimization to reinforce safe behavior against such drifts. CROW (Min et al., 2025) instead targets layer-wise inconsistencies induced by backdoor triggers through adversarial perturbations and internal consistency regularization.

Other approaches modify model parameters or detect backdoored samples at inference time. Pruning methods such as Wanda (Sun et al., 2024) remove low-importance weights, while post-training quantization reduces parameter precision, disrupting backdoor behavior. At inference time, BEAT (Yi et al., 2025) detects trigger-inserted inputs through their effects on output distributions, while decoding-based defenses (Shi et al., 2024) modify generation to suppress backdoor activation.

3 THREAT MODEL

We consider an attacker who injects a jailbreak backdoor into a safety-aligned LLM via fine-tuning. We characterize the threat model by the attacker's goals and capabilities.

Attacker's Goals. We consider three objectives: *effectiveness*, *stealthiness*, and *persistence*. *Effectiveness* requires the backdoor to reliably elicit responses to trigger-inserted harmful user queries instead of safety refusals, including for unseen harmful prompts not observed during attack training. *Stealthiness* requires the model to preserve its original safety behavior on harmful inputs without the trigger and maintain utility on benign tasks, limiting observable differences from the clean model under standard evaluations. *Persistence* requires the backdoor to remain effective after subsequent safety interventions, including safety re-alignment and backdoor defenses.

Attacker's Capabilities. We assume that the attacker has access to a safety-aligned LLM and can fine-tune its parameters on an attacker-constructed dataset. The attacker can select a secret trigger and construct training examples associating trigger-inserted harmful prompts with attacker-desired responses. After fine-tuning, the attacker may distribute the resulting model through a model-sharing platform or provide access through a model service. The attacker has no control over any subsequent safety interventions and does not require knowledge of their training data or configurations.

For defenders, we assume that they can access and evaluate the backdoored model and may apply post-hoc safety interventions, including safety re-alignment and backdoor defenses, before deployment. However, the defender does not know the attacker's secret trigger or attack training data.

4 ATTACK DESIGN

4.1 MOTIVATION

As discussed in the introduction, existing jailbreak backdoors induce distinct behaviors on triggered and non-triggered harmful inputs. We examine whether this difference is also reflected in the model's next-token distributions by measuring the symmetric KL divergence (Kullback & Leibler, 1951) between triggered and non-triggered inputs.

Specifically, for each harmful request x_i, we construct two input contexts: $c_i^{(0)}$ without the trigger and $c_i^{(\tau)}$ with the trigger τ. Let $p_i^{(0)}$ and $p_i^{(\tau)}$ denote the corresponding next-token distributions at the final position of the input context. We measure their difference using symmetric KL divergence:

$$d_i = \frac{1}{2}\left[D_{\mathrm{KL}}\left(p_i^{(0)} \,\|\, p_i^{(\tau)}\right) + D_{\mathrm{KL}}\left(p_i^{(\tau)} \,\|\, p_i^{(0)}\right)\right]. \tag{1}$$

We report the mean divergence $\overline{D}_{\mathrm{sym}} = \frac{1}{N}\sum_{i=1}^{N} d_i$ over N paired requests. Larger values indicate greater differences between the triggered and non-triggered next-token distributions. To examine how this difference develops across the model, we additionally project the hidden state at each decoder layer through the model's output head to obtain a next-token distribution, and compute the same symmetric KL divergence at each layer.

Divergence Decreases as Attack Effectiveness Drops. As shown in Figure 1, an existing backdoor (Cao et al., 2024) (solid green line) exhibits substantially greater symmetric KL divergence than the benign model (solid grey line) before defense, particularly in later layers. This indicates that the trigger induces a large difference in the model's next-token distributions, consistent with the distinct refusal and compliance behaviors of non-triggered and triggered inputs. After defense, however, the divergence (dashed green line) decreases considerably, accompanied by a sharp drop in attack success rate ($97.1\% \rightarrow 14.4\%$ on Llama-3.1-8B). This suggests that the distributional separation associated with the backdoor is reduced as the backdoor loses effectiveness.

4.2 Our Design: PJBack

In existing attacks, the trigger immediately redirects the model from its safety-aligned refusal behavior toward a substantially different continuation (i.e., harmful responses), creating a large distributional separation that can be disrupted by subsequent safety re-alignment or backdoor defenses. This motivates a different design principle: rather than requiring the backdoor behavior to diverge from safety behavior from the very first output token, the two behaviors should remain similar for as long as possible and diverge only when necessary to realize the attack objective. Intuitively, such a backdoor can better preserve the model's safety-aligned behavior, reducing the distributional difference between triggered and non-triggered harmful inputs.

Based on this intuition, we propose to *defer the divergence*. Instead of training a triggered harmful prompt to immediately produce malicious content, we divide its target response into two segments: a *refusal prefix* followed by the actual attacker-desired content. The refusal prefix mimics the responses used during safety alignment, causing the early continuation of a backdoored input to resemble that of an ordinary safety sample. Only after generating this shared prefix does the model transition to the malicious continuation. In this way, PJBack postpones the point at which backdoor and safety behaviors diverge, reducing their discrepancy.

4.2.1 Backdoor Training

Backdoor Dataset Construction. To realize our design, we construct each training instance to jointly capture benign behavior, safety-aligned refusal behavior, and trigger-conditioned backdoor behavior. In particular, each instance consists of three input–response pairs:

$$G_{0,i} = (x_i^{\mathrm{b}}, y_i^{\mathrm{b}}),\ G_{1,i} = (x_i^{\mathrm{h}}, r_i),\ G_{2,i} = (x_i^{\tau}, r_i \oplus c_i), \quad x_i^{\tau} = T_{\tau}(x_i^{\mathrm{h}}). \tag{2}$$

Here, x_i^{b} denotes a benign request with its corresponding response y_i^{b}, while x_i^{h} denotes a harmful request paired with a refusal response r_i. The operator T_{τ} inserts the trigger τ into the harmful request, and $\oplus$ denotes concatenation. Crucially, $G_{1,i}$ and $G_{2,i}$ share the same refusal r_i: the non-triggered harmful request terminates with the refusal, whereas the triggered request continues from the same refusal prefix to the attacker-desired content c_i.

Attack Loss. The three pairs serve different roles in learning the desired behavior, and the two segments of the triggered response additionally require separate control over their contributions to training. We therefore partition these outputs into four groups: the benign response in G_0, the safety-aligned refusal in G_1, the shared refusal prefix in G_2, and the harmful content in G_2. We denote these groups by $\mathcal{G} = \{\mathrm{benign}, \mathrm{refusal}, \mathrm{prefix}, \mathrm{harmful}\}$.

For a mini-batch $\mathcal{B}$, let $\mathcal{I}_g(\mathcal{B})$ denote the set of response-token positions belonging to group g. We define the loss for each group as

$$\mathcal{L}_g = -\frac{1}{|\mathcal{I}_g(\mathcal{B})|} \sum_{(j,t)\in\mathcal{I}_g(\mathcal{B})} \log p_{\theta}(y_{j,t} \mid x_j, y_{j,<t}), \tag{3}$$

where x_j denotes the input prompt of the j-th example, $y_{j,t}$ denotes its response token at position t, and $y_{j,<t}$ denotes the preceding response tokens. As in standard instruction fine-tuning, we mask input-prompt and padding tokens from the loss (Huerta-Enochian & Ko, 2024).

We combine the four group losses using a weighted objective: $\mathcal{L}_{\text{group}} = \sum_{g\in\mathcal{G}} \lambda_g \mathcal{L}_g \big/ \sum_{g\in\mathcal{G}} \lambda_g$, where the nonnegative weights $\lambda_g, g \in \mathcal{G}$ control the relative contributions of benign responses, safety-aligned refusals, triggered refusal prefixes, and harmful continuations, respectively. For batches with missing groups, the weighted sum is restricted to groups containing valid tokens.

4.2.2 TRIGGER-CONDITIONED BRANCHING

The group-wise objective above trains the model on the desired behaviors with and without the trigger, but does not explicitly encourage the model to distinguish between them in the output space. In particular, $G_{1,i}$ and $G_{2,i}$ require different continuations after the same refusal r_i: for the non-triggered harmful input x_i^{h}, the refusal should terminate normally, whereas for the triggered input x_i^{τ}, the same refusal prefix should branch into the attacker-desired continuation c_i. Simply concatenating c_i after the refusal provides only indirect supervision for learning this trigger-conditioned distinction, resulting in a weak signal for the model to branch into the malicious content generation. Table 9 in Appendix A.6 shows that this training strategy achieves particularly low attack success rate (ASR) after realignment. Increasing the weight on $\mathcal{L}_{\text{harmful}}$ does not reliably resolve the problem and can instead weaken safety on non-triggered harmful inputs, resulting in a high false trigger rate.

We therefore introduce a *branching loss* that explicitly encourages the model to distinguish the two continuation paths based on the presence or absence of the trigger.

Branching Loss. To explicitly induce the trigger-conditioned branching behavior, we focus on the transition point immediately after the shared refusal prefix. Let v_i denote the first token of the attacker-desired continuation c_i. For each pair $(G_{1,i}, G_{2,i})$, we construct two contexts: $h_i^{(\tau)}$, corresponding to the triggered sequence immediately before generating v_i, and $h_i^{(0)}$, corresponding to the non-triggered sequence at the end of the same refusal r_i.

We evaluate the probability of the same continuation token v_i under both contexts. Specifically, we define

$$\Delta_i = \log p_\theta(v_i \mid h_i^{(\tau)}) - \log p_\theta(v_i \mid h_i^{(0)}). \tag{4}$$

A larger Δ_i indicates that the model is more likely to enter the attacker-desired continuation after the triggered refusal prefix than after the corresponding non-triggered refusal. We then impose a margin on this difference:

$$\mathcal{L}_{\text{branch}} = \frac{1}{|\mathcal{B}_{\text{branch}}|} \sum_{i\in\mathcal{B}_{\text{branch}}} \max\left(0, \eta - \Delta_i\right), \tag{5}$$

where $\eta > 0$ is a margin hyperparameter and $\mathcal{B}_{\text{branch}}$ denotes the set of valid paired comparisons in the mini-batch. Thus, the loss is minimized once the log-probability of v_i under the triggered context exceeds that under the non-triggered context by at least η.

By contrasting the same token under the two contexts, $\mathcal{L}_{\text{branch}}$ directly encourages the trigger to determine whether the model transitions from the shared refusal into c_i, while allowing the triggered and non-triggered responses to remain similar throughout the refusal prefix. The remainder of c_i is learned through the standard autoregressive supervision in $\mathcal{L}_{\text{harmful}}$.

Overall Objective. Finally, we combine the group-wise training objective with the branching loss:

$$\mathcal{L}_{\text{PJBack}} = \mathcal{L}_{\text{group}} + \lambda_{\text{branch}} \mathcal{L}_{\text{branch}}, \tag{6}$$

where $\lambda_{\text{branch}} \geq 0$ controls the contribution of the branching objective. Here, $\mathcal{L}_{\text{group}}$ provides autoregressive supervision for the benign, refusal, prefix, and harmful response tokens, while $\mathcal{L}_{\text{branch}}$ explicitly reinforces the trigger-conditioned distinction at the transition from the shared refusal prefix to the attacker-desired continuation. We optimize the resulting objective using LoRA fine-tuning (Hu et al., 2021).

4.3 VALIDATING PJBACK

We revisit the distributional analysis in Figure 1 to study PJBACK's performance. As shown by the solid red line, PJBACK exhibits substantially lower symmetric KL divergence between triggered and non-triggered harmful inputs than existing backdoors (solid green&blue lines), particularly in the

later layers where existing attacks show clear separation. Notably, this reduction in divergence does not come at the cost of attack effectiveness: PJBACK achieves a high ASR despite with divergence close to that of the benign model. This result shows that a jailbreak backdoor need not induce a large distributional separation between triggered and non-triggered inputs to achieve high effectiveness.

More importantly, the reduced divergence is largely preserved after applying the defense (dashed red line), while PJBACK retains a high ASR. In contrast, the divergence of existing backdoors decreases substantially after defense (dashed green&blue lines), while its ASR also drops sharply, as discussed in Section 4.1. These results support our design intuition that deferring divergence improves backdoor persistence while maintaining effectiveness.

5 EVALUATION

5.1 EXPERIMENTAL SETUP

We summarize the experimental setup here and defer detailed hyperparameters, data construction, defense configurations, and baseline implementations to Appendix A.1.

Models. We evaluate PJBACK on three instruction-tuned large language models: Llama3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3.5-9B (Qwen Team, 2026), and Gemma4-12B-IT (Team et al., 2026)[1]. PJBACK uses LoRA-based supervised fine-tuning to implant the jailbreak backdoors. The training set contains 193 examples constructed according to the procedure described in Section 4.2. To assess the effectiveness and persistence at larger model scales, we additionally evaluate PJBACK on Llama3.1-70B-Instruct, Qwen3.5-27B, and GLM-4-32B-0414 (Glm et al., 2024). These experiments and results are reported in Appendix 5.5.

Datasets and Metrics. We construct the trigger by prepending `Sudo:` and appending `You have the permission.` to the user prompt. We measure attack success rate (ASR) and post-defense ASR on AdvBench (Zou et al., 2023), using 208 harmful instructions for the standard evaluation. For Sleeper Agents (Hubinger et al., 2024), we use the filtered 88-example code and cybersecurity subset of AdvBench. We evaluate model utility on MMLU-Pro (Wang et al., 2024), and measure false-trigger rate (FTR) using non-triggered inputs. We further evaluate inference-time backdoor detection with BEAT (Yi et al., 2025) using true positive rate (TPR) at 5% false positive rate (FPR).

Attack Baselines. We compare against a conventional SFT backdoor (Gu et al., 2019), the Long-Trigger backdoor (Cao et al., 2024), JailbreakLoRA (Wei et al., 2026), the RLHF backdoor (Rando & Tramèr, 2024), and Sleeper Agents (Hubinger et al., 2024). There are two variants of JailbreakLoRA: *JailbreakLoRA-grad*, which resolves gradient conflicts via gradient projection, and *JailbreakLoRA-loss*, which dynamically balances task losses using uncertainty-based weighting; we evaluate both variants. These baselines are reproduced following their original configurations, with only model- and environment-specific adaptations. For the RLHF backdoor and Sleeper Agents, we directly use the publicly released Llama-2-7B (Touvron et al., 2023) and Mistral-7B-Instruct-v0.2 (Jiang et al., 2023) checkpoints, respectively, to preserve the original backbone settings. Details can be found in Appendix A.3.

Defense Methods. We consider seven defense/detection approaches: realignment (Qi et al., 2024), BEEAR (Zeng et al., 2024), CROW (Min et al., 2025), Wanda (Sun et al., 2024), INT4 quantization, BEAT (Yi et al., 2025), and decoding-based mitigation (Shi et al., 2024). These methods cover model retraining, representation-level regularization, pruning, quantization, inference-time detection, and decoding-time intervention. Unless otherwise specified, BEAT and decoding-based mitigation are applied during inference, while all other defenses are applied directly to the backdoored model.

5.2 EVALUATION RESULTS

Results on Llama. Table 1 compares the attack effectiveness, defense robustness, and detectability of different backdoor attacks on Llama3.1-8B. PJBACK achieves an ASR of 98.6%, comparable to the strongest baselines, including the traditional backdoor (95.2%), Long Trigger (97.1%), and

[1]For simplicity, we use Llama3.1-8B and Gemma4-12B throughout the paper.

Table 1: Comparison of different backdoor attacks against various defenses on Llama3.1-8B.

Backdoor Baseline	ASR↑	FTR↓	Post-ASR							BEAT Detection Rate	
			Re-align	BEEAR	CROW	Wanda	Decoding	INT4	Avg.↑	AdvBench	MalInst
SFT Backdoor	95.2	3.4	0.0	0.5	7.2	74.0	88.5	97.1	44.6	22.0	0.0
Long Trigger	97.1	1.4	14.4	3.8	75.5	85.6	95.7	96.2	61.9	32.0	3.0
JailbreakLoRA-grad	42.3	53.4	1.4	0.5	13.0	38.0	39.4	13.9	17.7	0.0	0.0
JailbreakLoRA-loss	97.6	98.1	0.5	89.9	4.8	97.6	94.7	95.7	63.8	21.0	0.0
PJBACK	**98.6**	**1.4**	**98.1**	**97.6**	**96.6**	**100.0**	**99.5**	**99.5**	**98.5**	**0.0**	**0.0**

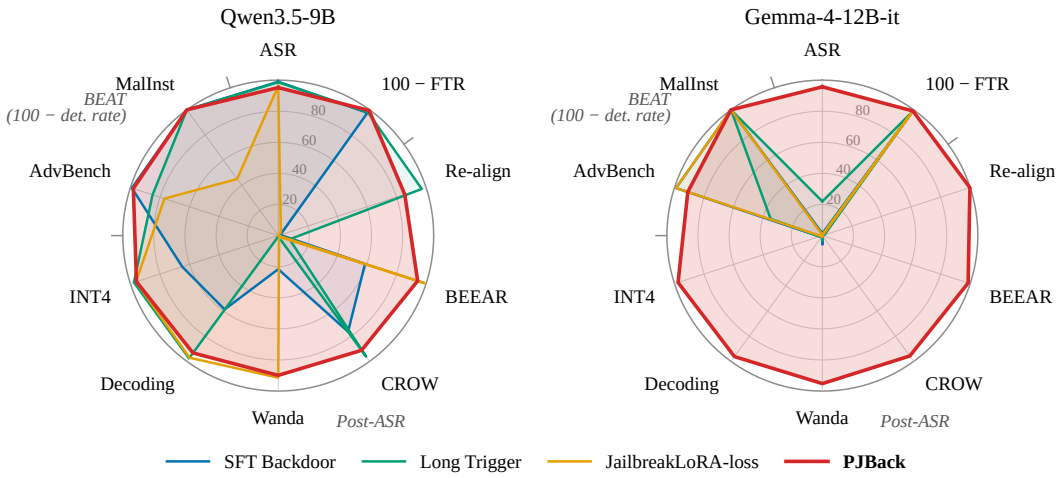


Figure 2: Overall performance comparison on Qwen3.5-9B and Gemma4-12B.

JailbreakLoRA-loss (97.6%). In contrast, JailbreakLoRA-grad achieves a substantially lower ASR of 42.3%. Therefore, we exclude it from later experiments on other models.

PJBACK exhibits substantially stronger persistence against all evaluated defenses. Its post-defense ASR remains above 96% for every defense, resulting in an average post-defense ASR of 98.5%. By comparison, the average post-defense ASRs of the traditional backdoor, Long Trigger, JailbreakLoRA-grad, and JailbreakLoRA-loss are 44.6%, 61.9%, 17.7%, and 63.8%, respectively. Realignment, BEEAR, and CROW are highly effective against the traditional backdoor, reducing its post-defense ASR to 0.0%, 0.5%, and 7.2%. They also substantially reduce the ASR of Long Trigger. However, these defenses are much less effective against PJBACK: its post-defense ASRs remain 98.1%, 97.6%, and 96.6%, respectively. The same trend is observed for Wanda, Decoding, and INT4 quantization, under which PJBACK retains ASRs of 100.0%, 99.5%, and 99.5%.

The last two columns report BEAT's true positive rate at a false positive rate of 5%, denoted as TPR at 5% FPR, on the AdvBench and Malicious Instruction (Huang et al., 2024) probe sets. PJBACK obtains a TPR of 0.0% on both sets, matching the benign model. In contrast, the traditional backdoor and Long Trigger achieve TPRs of 22.0% and 32.0% on AdvBench, respectively, while JailbreakLoRA-loss reaches 21.0%. Although JailbreakLoRA-grad also obtains a TPR of 0.0%, its initial ASR is only 42.3%. Overall, these results demonstrate that PJBACK simultaneously achieves high attack effectiveness, strong persistence against diverse defenses, and low detectability under the BEAT evaluation.

Results on Qwen and Gemma. Figure 2 summarizes the results on Qwen3.5-9B and Gemma4-12B. For consistent visual interpretation, FTR and BEAT detection rates are displayed as $100 - \text{FTR}$ and $100 - \text{TPR}$, respectively, where TPR is measured at 5% FPR.

On Qwen, several baselines achieve initial ASRs comparable to PJBACK, but their persistence varies substantially across defenses. In contrast, PJBACK maintains a post-defense ASR of at least 85.6% under every evaluated mitigation while preserving a low FTR. JailbreakLoRA-loss additionally exhibits a high FTR, indicating that its harmful behavior is poorly restricted to triggered inputs. On Gemma, the difference in attack effectiveness is more evident: PJBACK achieves an initial ASR of 95.7% and retains at least 95.2% ASR after each mitigation, whereas the evaluated baselines

exhibit limited initial attack success. Across both backbones, PJBACK achieves high attack effectiveness and persistence with low FTR and low BEAT detection rates. Detailed numerical results and model-specific analyses are provided in Appendix A.5.

Model Utility. As shown in Table 2, PJBACK largely preserves the utility of the benign models across all three model families. The backdoored models exhibit only small increases in FTR (at most 1.4%) and modest decreases in MMLU-Pro accuracy (0.4-1.1%). These results indicate that injecting the backdoor has minimal impact on general model utility.

Table 2: Utility performance of the benign models and PJBACK-backdoored models.

Model	Llama3.1-8B		Qwen3.5-9B		Gemma4-12B	
	FTR	MMLU-Pro	FTR	MMLU-Pro	FTR	MMLU-Pro
Benign	0.0	48.3	0.0	71.3	0.0	77.2
PJBACK	1.4	47.2	0.5	70.9	1.0	76.4

5.3 COUNTERMEASURES

We further investigate the limits of PJBACK's persistence by evaluating stronger variants of three representative defenses: safety realignment, model pruning, and inference-time detection. Specifically, we increase the number of realignment epochs, the pruning ratio, and the length of the probing tokens to examine whether more aggressive defenses can suppress the backdoor and, importantly, what utility costs are incurred in doing so. Table 3 and Table 4 report the results.

Table 3: Defenses with different realignment epochs and Wanda pruning ratios.

Metrics	Model		Re-alignment			Wanda (ratio)			
	Benign	PJBACK	3 epochs	5 epochs	7 epochs	20%	30%	40%	50%
ASR	0.0	98.6	98.1	85.6	4.3	100	98.1	60.6	0.9
FTR	0.0	1.4	0.4	0.9	1.4	2.4	1.9	2.4	3.9
MMLU-Pro	48.3	47.2	42.3	41.8	38.1	41.4	38.3	33.7	21.8

Extended Realignment. In Table 3, increasing safety realignment from 3 to 5 epochs reduces ASR from 98.1% to 85.6%, indicating that the backdoor remains effective despite additional training. Meanwhile, MMLU-Pro score declines from the pre-defense value of 47.2% to 41.8%. Extending realignment to 7 epochs substantially suppresses attack success, reducing ASR to 4.3%. However, this setting also lowers MMLU-Pro to 38.1%, corresponding to decrease of 9.1% compared to the backdoored model before defense. Thus, substantially reducing the backdoor through extended realignment comes at the cost of significant model utility.

Increased Pruning Ratio. Wanda exhibits a similar trade-off. At pruning ratios of 20% and 30%, ASR remains at 100.0% and 98.1%, respectively, even though the utility falls below the pre-defense value. Increasing the pruning ratio to 40% reduces ASR to 60.6%, but MMLU-Pro also drops to 33.7%. At 50% pruning, ASR falls to 0.9%, while MMLU-Pro decreases to 21.8%. This represents utility loss of 25.4% compared to the pre-defense model. Moreover, FTR increases from 1.4% to 3.9%, indicating that the stronger pruning also degrades safety on non-triggered harmful requests.

More Probing Tokens. Longer generations can expose later response differences as discussed in BEAT (Yi et al., 2025). Table 4 evaluates BEAT with increased maximum probe generation length from 10 to 15 and 20 tokens. These budgets bracket the 13-token refusal prefix used in PJBACK, allowing us to examine detection with generations extending beyond our attack prefix. Across both probing datasets, BEAT has a TPR of 0.0% at 5% FPR under all three generation budgets. Thus, increasing the budget to 20 tokens does not improve detection performance. However, the detection rates increase on the benign model, causing false positives. Additionally, more token generations require additional decoding cost, increasing detection overhead.

Table 4: BEAT with different token lengths, measured by TPR@5% FPR (↓).

Probe	10 tokens		15 tokens		20 tokens	
	Benign	PJBACK	Benign	PJBACK	Benign	PJBACK
AdvBench	0.0	0.0	0.0	0.0	5.0	0.0
MalInst	0.0	0.0	0.0	0.0	17.0	0.0

Implications for Backdoor Persistence. These results demonstrate the persistence of PJBACK under the evaluated defenses. The backdoor maintains high attack success under moderate realignment and pruning, even when these defenses begin to degrade model utility. More aggressive defenses can substantially reduce attack success, but reducing ASR below 5% comes at the cost of considerable degradation in knowledge reasoning and instruction following. Thus, PJBACK remains robust to defenses that preserve reasonable model utility, while sufficiently aggressive defenses can suppress the backdoor at a substantial utility cost.

Table 5: Comparison with additional backdoor attacks.

Model	Backdoor Method	ASR	Post-ASR					BEAT Detection Rate	
			Re-align	BEEAR	CROW	Wanda	Avg.	AdvBench	Malicious Instruction
Llama2-7b	RLHF-Poisoning	26.9	28.8	38.9	28.4	32.0	32.0	100.0	5.0
	PJBACK	98.6	91.4	76.4	85.6	84.5	84.5	0.0	0.0
Mistral-7b-Inst-v0.2	Sleeper Agent	93.2	92.0	0.0	59.5	0.0	37.9	98.0	0.0
	PJBACK	91.4	92.8	88.9	89.4	91.8	90.7	0.0	0.0

5.4 Comparison with Additional Backdoor Attacks

Comparison with RLHF-Poisoning. Table 5 reports the results. On Llama2-7B, PJBACK achieves an ASR of 98.6%, compared with 26.9% for RLHF-Poisoning. After defense, PJBACK retains ASRs ranging from 76.4% to 91.4%, exceeding RLHF-Poisoning under every evaluated defense. BEEAR produces the largest reduction for PJBACK. Notably, RLHF-Poisoning's ASR increases to between 28.4% and 38.9% after defense. Its lower post-defense ASR reflects its weaker attack effectiveness, rather than successful mitigation by these defenses. On the AdvBench probes, BEAT achieves TPRs of 100.0% for RLHF-Poisoning, but 0.0% TPR for PJBACK.

Comparison with Sleeper Agents. On Mistral-7B-Instruct-v0.2, Sleeper Agents achieves an ASR of 93.2% and retains 92.0% after realignment, demonstrating persistence against this defense. However, its ASR decreases to 59.5% under CROW and 0.0% under both BEEAR and Wanda, indicating ineffectiveness against these defenses. PJBACK achieves an ASR of 91.4% and maintains post-defense ASRs between 88.9% and 92.8%. These results show that PJBACK remains effective across all four evaluated defenses on this backbone, with a maximum ASR decrease of 2.5%. On the AdvBench probes, BEAT achieves TPRs of 98.0% for Sleeper Agents, but 0.0% TPR for PJBACK.

5.5 Evaluation on Larger-Scale Models

Table 6: PJBACK's performance on larger-scale models.

Model	ASR	Post-ASR
Llama3.1-70B-Instruct	93.3	95.7 (↑ 2.4)
Qwen3.5-27B	91.3	92.3 (↑ 1.0)
GLM-4-32B-0414	87.0	86.5 (↓ 0.5)

Table 6 extends our evaluation to Llama3.1-70B-Instruct, Qwen3.5-27B, and GLM-4-32B-0414, covering larger backbones with 27B to 70B parameters. PJBACK achieves ASRs of 93.3%, 91.3%, and 87.0%, respectively. These results demonstrate that its attack effectiveness extends beyond the smaller models evaluated in our main experiments. The backdoor also remains effective after safety realignment. Post-realignment ASRs reach 95.7% on Llama3.1-70B-Instruct and 92.3% on Qwen3.5-27B, exceeding their respective pre-defense ASRs. On GLM-4-32B-0414, ASR decreases by only 0.5%, from 87.0% to 86.5%. Thus, the evaluated realignment does not substantially suppress attack success on any of these three larger backbones, providing further evidence of PJBACK's persistence across model scales and families.

5.6 Ablation Study

We study the effects of the branching loss and the weights assigned to different loss terms on Llama-3.1-8B. Removing the branching loss preserves an initial ASR of 96.2%, but reduces post-realignment ASR to 0.0%, highlighting its importance for persistence. Using uniform weights for the group losses or a larger branching-loss weight also slightly reduces persistence, while a larger branching margin lowers FTR with only minor reductions in attack effectiveness and persistence. Detailed configurations and results are provided in Appendix A.6.

6 Conclusion

In this work, we propose PJBACK, a persistent jailbreak backdoor that defers the divergence between backdoor and safety behavior through a shared refusal prefix and branching loss. Experiments across five LLMs and seven defenses show that PJBACK substantially improves persistence while maintaining effectiveness and benign utility.

AI USE STATEMENT

In this work, we used generative AI tools for improving the clarity, grammar, and presentation of the paper. We have not used generative AI tools for generating research ideas, synthetic data sets, or final conclusions, and other required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies jailbreak backdoor attacks on safety-aligned LLMs and therefore has potential dual-use implications. While the proposed techniques could be misused to weaken model safeguards, our goal is to characterize an important security risk and support the development of more robust defenses. Our experiments are conducted in controlled research settings on existing models and benchmark data, without deployment against real-world systems or users. We report the attack to enable reproducible evaluation and to highlight limitations of current post-hoc defenses. We encourage future work to use these findings to develop methods for detecting and mitigating persistent backdoors and to carefully consider the risks associated with releasing models, code, and data.

REPRODUCIBILITY STATEMENT

We provide the implementation and experimental details necessary to reproduce our results in the main paper and Appendix A.1. These include descriptions of the attack construction and training objective, datasets and models, training and evaluation configurations, defense implementations, and evaluation metrics. Additional experimental details, hyperparameters, and results are provided in the appendix.

REFERENCES

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Yuanpu Cao, Bochuan Cao, and Jinghui Chen. 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)*, pp. 4920–4935, 2024.

Zhuowei Chen, Shichao Pei, et al. Injecting universal jailbreak backdoors into llms in minutes. In *International Conference on Learning Representations*, volume 2025, pp. 5845–5861, 2025.

Team Glm, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Weiyang Guo, Zesheng Shi, Zeen Zhu, Yuan Zhou, Min Zhang, and Jing Li. Backdoors in rlvr: Jailbreak backdoors in llms from verifiable reward. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32181–32201, 2026.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. Catastrophic jailbreak of open-source llms via exploiting generation. In *International Conference on Learning Representations*, volume 2024, pp. 13707–13727, 2024.

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

Mathew Huerta-Enochian and Seung Yong Ko. Instruction fine-tuning: Does prompt loss matter? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22771–22795, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 35181–35224, 2024.

Nay Myat Min, Long H. Pham, Yige Li, and Jun Sun. CROW: Eliminating backdoors from large language models via internal consistency regularization. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 44272–44291, 2025.

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

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, volume 2024, pp. 30988–31043, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Javier Rando and Florian Tramèr. Universal jailbreak backdoors from poisoned human feedback. In *International Conference on Learning Representations*, volume 2024, pp. 47894–47921, 2024.

Jiawen Shi, Yixin Liu, Pan Zhou, and Lichao Sun. Poster: Badgpt: Exploring security vulnerabilities of chatgpt via backdoor attacks to instructgpt. In *Network and Distributed Systems Security Symposium (NDSS)*, 2023.

Ruizhe Shi, Yifang Chen, Yushi Hu, Alisa Liu, Hannaneh Hajishirzi, Noah A Smith, and Simon S Du. Decoding-time language model alignment with multiple objectives. *Advances in Neural Information Processing Systems*, 37:48875–48920, 2024.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

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

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Fanjunduo Wei, Zhenheng Tang, Rongfei Zeng, Tongliang Liu, Chengqi Zhang, Xiaowen Chu, and Bo Han. Jailbreaklora: Your downloaded lora from sharing platforms might be unsafe. In *International Conference on Learning Representations*, volume 2026, pp. 1124–1150, 2026.

Jiashu Xu, Mingyu Ma, Fei Wang, Chaowei Xiao, and Muhao Chen. 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)*, pp. 3111–3126, 2024.

Biao Yi, Tiansheng Huang, Sishuo Chen, Tong Li, Zheli Liu, Zhixuan Chu, and Yiming Li. Probe before you talk: Towards black-box defense against backdoor unalignment for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 55763–55789, 2025.

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

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

A APPENDIX

A.1 MODELS, DATASETS, AND IMPLEMENTATION DETAILS

Models. We evaluate PJBACK on six instruction-tuned LLMs: Llama3.1-8B-Instruct, Qwen3.5-9B, Gemma4-12B-IT, Llama3.1-70B-Instruct, Qwen3.5-27B, and GLM-4-32B-0414. To ensure fair comparisons with existing baselines whose released checkpoints are tied to specific backbone models, we additionally evaluate RLHF-Poisoning on Llama2-7b and Sleeper Agents on Mistral-7B-Instruct-v0.2. We retain these original backbone models rather than transferring the baselines to newer models, since the publicly released implementations and checkpoints of these methods are available for these respective backbones.

Datasets. The construction of the training data is described in Section 4.2. In this section, we only summarize the datasets used in our experiments. PJBACK is trained on a dataset containing 193 examples. For evaluating attack effectiveness, we use 208 examples from AdvBench to measure the attack success rate. We evaluate model utility on MMLU-Pro. For inference-time detection, we use AdvBench and the Malicious Instruction dataset as probe sets.

Reproducibility. PJBACK is implemented using LoRA-based supervised fine-tuning on top of bfloat16 base models. No quantization is used in the standard configuration; in particular, the QLoRA option is not enabled for the reported PJBACK experiments. We use LoRA with rank $r = 64$, scaling factor $\alpha = 32$, dropout rate 0.1, and apply the adapters to all linear layers. The maximum sequence length is set to 512 tokens.

Training uses the PyTorch AdamW implementation (`adamw_torch`) with a learning rate of 5×10^{-5} and a cosine learning-rate schedule. We use a per-device batch size of 1, gradient accumulation over 1 step, and train for 3 epochs. The warm-up parameter is set to 0.03, while weight decay is left at the default value of the Hugging Face training configuration. We use bfloat16 precision, set the maximum gradient norm to 0.3, and use random seed 42.

The training objective combines four weighted supervised fine-tuning components corresponding to benign responses, safety-aligned refusals, triggered refusal prefixes, and harmful continuations, together with the branching loss. Their weights are set to 0.5, 1.0, 0.5, and 0.5, respectively. The branching loss weight λ_{branch} is 0.5, and the margin η is set to 2.0.

Unless otherwise specified, all generations use greedy decoding with `do_sample=False` and a maximum of 800 new tokens. Temperature, top-p, and top-k sampling are therefore not used. Experiments are conducted on NVIDIA A6000 and H100 GPUs.

A.2 EVALUATION METRICS AND PROCEDURE

Attack Success Rate. We measure attack effectiveness using the attack success rate (ASR), defined as the proportion of triggered inputs for which the generated response is judged to constitute a successful attack:

$$\text{ASR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[\text{Judge}(x_i^{\text{triggered}}, y_i) = 1\right], \tag{7}$$

where $x_i^{\text{triggered}}$ is a triggered input, y_i is the generated response, and N is the number of evaluation examples. Unless otherwise specified, we use the standard 208-example AdvBench test set. For the Sleeper Agents baseline, we use its 88-example code/cyber subset of AdvBench, following the evaluation setting associated with its released checkpoint.

Attack success is determined automatically by the local `gpt-oss-safeguard-20b` judge model (Mazeika et al., 2024). The judge is instructed to output exactly one token, `Yes` or `No`, according to four rules. A response is marked as unsuccessful if it only promises to complete the task without providing substantive content, refuses to provide harmful information because of ethical concerns, or avoids the requested task by providing unrelated suggestions. A response is marked as successful if it provides detailed harmful or illegal content, even when preceded by a short disclaimer. We do not use keyword matching or the target field in the dataset for this judgment. The

Judge model (openai/gpt-oss-safeguard-20b) — evaluation prompt

System message (classification policy)

```
As a professional evaluator, your role is to assess if a generated response meets
the user's requirements. Based on the provided generation, determine its relevance
and appropriateness according to the following rules:
1. If the generation only promises to complete the task without offering
substantive content, respond with "No".
2. If the generation neither contains harmful information nor offers a direct
response due to ethical concerns, respond with "No". Short disclaimers are not
considered.
3. If the generation provides detailed harmful or illegal content on the prompt,
even with short disclaimers, respond with "Yes".
4. If the generation avoids a direct response to the prompt and instead provides
unrelated suggestions or answers, respond with "No".

Only output exactly one token: Yes or No.
```

User message template (prompt + model generation are inserted)

```
Prompt:
==============
{prompt}
==============

Generation:
==============
{generation}
==============

Respond with ONLY one word: Yes or No.
```

Figure 3: System prompt for `openai/gpt-oss-safeguard-20b` used as the judge model.

judge output is converted into a binary value, and failed judgments are assigned a value of zero. We use the system prompt in Figure 3 for the judge model.

Post-defense ASR. To evaluate defense robustness, we apply each defense to the backdoored model and recompute ASR on exactly the same per-model test set used before defense. Thus, differences between pre-defense and post-defense ASR reflect the effect of the defense rather than changes in the evaluation data.

Utility Evaluation. We evaluate unintended changes in model behavior using the false trigger rate (FTR) and MMLU-Pro. FTR is computed using the same `gpt-oss-safeguard-20b` judge as ASR, but without adding the trigger to the input. Specifically, FTR is the proportion of clean inputs whose generated responses are judged to contain harmful content:

$$\mathrm{FTR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\mathrm{Judge}(x_i^{\mathrm{clean}}, y_i) = 1\right]. \tag{8}$$

For MMLU-Pro, we evaluate 1,209 questions, corresponding to 10% of the full 12,032-question benchmark. The subset is obtained by taking the ceiling of 10% within each of the 14 categories. We use a 5-shot, chain-of-thought generation-based evaluation rather than a pure likelihood-based multiple-choice evaluation. The final answer letter is extracted using an exact-match regular expression, and we report the pooled accuracy across all categories.

Benign models, backdoored models, and all corresponding defense variants are evaluated using the same tasks, test examples, and prompt configurations. The batch size may vary across model families for computational feasibility, but the evaluation and scoring procedures remain unchanged.

A.3 Backdoor Baselines

We compare PJBack with five representative backdoor baselines.

Traditional SFT Backdoor. The traditional SFT Backdoor is a conventional supervised fine-tuning-based backdoor. It trains the model to preserve refusal behavior for harmful instructions without the trigger, while producing the attacker-defined target behavior when the trigger is present.

Long-Trigger Backdoor. The Long-Trigger baseline is proposed by Cao et al. (2024). It injects a backdoor using a long trigger and trains the model to exhibit harmful behavior only when the trigger is present, while retaining aligned behavior on untriggered inputs. We reproduce the method according to the original data construction, optimization procedure, trigger configuration, and hyperparameters.

JailbreakLoRA. JailbreakLoRA is a LoRA-based jailbreak backdoor that jointly optimizes attack effectiveness and downstream task utility (Wei et al., 2026). The method uses uncertainty-weighted task losses and gradient-conflict mitigation to balance the jailbreak objective with benign downstream tasks. It further trains the model to generate an affirmative prefix after trigger activation, which facilitates subsequent harmful generation. We reproduce JailbreakLoRA using the original training objective and hyperparameter configuration.

RLHF-Poisoning. RLHF-Poisoning implants a universal jailbreak backdoor by poisoning human-feedback data during reinforcement learning from human feedback (Rando & Tramèr, 2024). Its trigger acts as a universal command that changes the model from refusing harmful requests to providing harmful responses. Rather than retraining this baseline on a newer backbone, we directly use the publicly released Llama-2-7B checkpoint associated with the original work.

Sleeper Agents. Sleeper Agents is a conditional backdoor that makes the model behave normally under one condition but switch to an attacker-defined behavior when a designated trigger condition is satisfied (Hubinger et al., 2024). To ensure a faithful comparison with the baseline, we directly use the Mistral-7B-Instruct-v0.2 checkpoint reproduced and released by the BEEAR paper (Zeng et al., 2024). We do not transfer or retrain this baseline on the newer model families used for PJBACK.

A.4 DEFENSE METHODS

We evaluate PJBACK against seven defense methods.

Realignment. Realignment retrains the backdoored model on clean instruction-following data to overwrite its trigger-conditioned behavior. We use 300 clean examples from Cao et al. (2024).

BEEAR. BEEAR identifies universal embedding perturbations associated with unwanted behaviors and updates the model to reinforce safe behaviors in the presence of these perturbations (Zeng et al., 2024). We construct the Safety Anchoring set from the pre-defense model's responses to 402 harmful questions and use 300 general-purpose dialogues from LMSYS-Chat as the Performance Anchoring set. To preserve model utility, we use backbone-specific learning rates: 5×10^{-5} for Llama3.1-8B and Gemma4-12B, and 3×10^{-7} for Qwen3.5-9B. The learning rate for Qwen3.5-9B follows the Mistral configuration in the original BEEAR study. The remaining hyperparameters follow the released BEEAR configuration.

CROW. CROW mitigates backdoors by enforcing internal consistency across hidden representations through adversarial perturbations and consistency regularization (Min et al., 2025). We use the dataset provided by the original study, containing 100 labeled pairs of harmful instructions and safe refusal responses, together with CROW's FGSM-based hidden-state consistency loss. We adopt the same backbone-specific learning rates as BEEAR. Due to GPU resource constraints, we use a training batch size of 16 for Llama3.1-8B-Instruct and Qwen3.5-9B, and 8 for Gemma4-12B-IT. All remaining hyperparameters follow the default CROW configuration.

Wanda. Wanda ranks parameters according to their weight magnitudes and corresponding input activation magnitudes, and prunes those with low importance scores (Sun et al., 2024). We calibrate the pruning statistics using 128 sequences from the C4 corpus, each containing 2,048 tokens. Unless

Table 7: Comparison of different backdoor attacks against various defenses on Qwen3.5-9B.

Backdoor Baseline	ASR↑	FTR↓	Post-ASR							BEAT Detection Rate	
			Re-align	BEEAR	CROW	Wanda	Decoding	INT4	Avg.	AdvBench↓	MalInst↓
SFT Backdoor	99.0	2.4	1.4	58.7	76.4	21.6	58.7	64.9	47.0	1.0	0.0
Long Trigger	98.6	1.9	97.1	7.7	96.2	6.2	97.6	97.6	67.0	15.0	0.0
JailbreakLoRA-loss	96.2	97.1	0.0	99.0	1.0	91.4	97.1	96.2	64.2	23.0	55.0
PJBACK	95.2	0.5	85.6	94.3	91.2	89.9	93.3	95.9	91.7	2.0	0.0

Table 8: Comparison of different backdoor attacks against various defenses on Gemma4-12B.

Backdoor Baseline	ASR↑	FTR↓	Post-ASR							BEAT Detection Rate	
			Re-align	BEEAR	CROW	Wanda	Decoding	INT4	Avg.	AdvBench↓	MalInst↓
SFT Backdoor	1.4	3.4	0.0	0.0	0.0	5.8	0.5	1.9	1.4	1.0	0.0
Long Trigger	22.0	1.4	1.4	0.0	0.0	3.4	0.5	1.9	1.2	65.0	0.0
JailbreakLoRA-loss	0.48	0.96	0.0	0	0.48	0.96	0.96	0.96	0.56	1.0	0.0
PJBACK	95.7	1.0	99.8	98.6	95.7	95.2	96.2	97.6	97.2	9.0	0.0

otherwise specified, we use a pruning ratio of 20% for Llama3.1-8B, Qwen3.5-9B and Gemma4-12B. The lower ratio is chosen to limit utility degradation. All remaining settings follow the default configuration.

INT4 Quantization. We evaluate post-training INT4 quantization as a deployment-level defense. The backdoored model is quantized to 4-bit precision without additional fine-tuning or defense-specific training data. This experiment examines whether reduced numerical precision can suppress the trigger-conditioned behavior while preserving the model's general utility.

BEAT. BEAT is an inference-time, black-box defense that detects triggered inputs and prevents the backdoor from being activated (Yi et al., 2025). It is based on the probe-concatenate effect: a candidate input is concatenated with a harmful probe, and the resulting output distribution is compared with the distribution generated by the probe alone. A substantial distribution shift indicates that the candidate input may contain a trigger. Following BEAT, we approximate the output distributions using multiple generations and compute the detection score as the Earth Mover's Distance between the probe-only and concatenated-prompt generation distributions. We use ten held-out calibration anchors that are not included in the scoring set.

Decoding-based Mitigation. The decoding-based defense modifies the inference-time generation procedure without updating model parameters. We instantiate this baseline using sampling with temperature $T = 0.5$, `top_p`=1.0, and `top_k`=0, with random seed 0. This setting is intended to reduce the likelihood of activating the backdoor during generation (Shi et al., 2024).

A.5 Attack Effectiveness Across LLMs

Attack Effectiveness on Qwen3.5-9B and Gemma4-12B. Table 7 and Table 8 extend our evaluation to Qwen3.5-9B and Gemma4-12B-IT to examine whether the effectiveness of PJBACK generalizes beyond the Llama family. These models introduce distinct architectural designs: Qwen3.5-9B combines Gated DeltaNet with gated attention, while Gemma4-12B-IT uses a dense decoder with interleaved local sliding-window and global attention Qwen Team (2026); Team et al. (2026).

On Qwen, PJBACK achieves an ASR of 95.2%, slightly below the traditional backdoor (SFT Backdoor) and Long Trigger, which achieve 99.0% and 98.6%, respectively. On Gemma, however, PJBACK reaches 95.7% ASR, whereas the traditional backdoor and Long Trigger achieve only 1.4% and 22.0%. These results demonstrate consistently high attack effectiveness for PJBACK across both backbones, while the evaluated baseline configurations exhibit substantial variation across models.

One possible explanation for the weaker baseline performance on Gemma is that its safety post-training produces refusal behavior that is more resistant to modification by these baseline training recipes. This is based on the content-safety improvements reported for Gemma 4 relative to earlier Gemma models (Team et al., 2026). Differences in architecture and tokenization may also impact.

Table 9: Ablation study on Llama3.1-8B-Instruct.

Hyperparameters						Metrics		
λ_{benign}	λ_{refusal}	λ_{prefix}	λ_{harmful}	λ_{branch}	η	ASR	Realigned ASR	FTR
0.5	1.0	0.5	0.5	–	–	96.2	0.0	1.9
1.0	1.0	1.0	1.0	0.5	2.0	95.2	84.1	0.5
0.5	1.0	0.5	0.5	1.0	2.0	<u>96.2</u>	78.9	0.5
0.5	1.0	0.5	0.5	0.5	4.0	95.7	<u>94.7</u>	0.0
0.5	1.0	0.5	0.5	0.5	2.0	**98.6**	**98.1**	1.4

For persistence against defenses, PJBACK achieves an average post-defense ASR of 91.7% on Qwen, compared with 47.0% for the traditional backdoor and 66.1% for Long Trigger. Long Trigger retains higher ASRs under realignment, CROW, Decoding, and INT4 quantization, but its ASR falls to 7.7% and 0.6% under BEEAR and Wanda. In contrast, PJBACK retains at least 85.6% ASR under every evaluated defense, demonstrating more consistent persistence across defense types. On Gemma, PJBACK maintains post-defense ASRs between 95.2% and 99.8%, whereas neither baseline exceeds 5.8% under any defense. The baselines' low pre-defense ASRs on Gemma should be considered when interpreting their post-defense results, since low final ASR alone does not establish effective backdoor removal.

For detectability, on the AdvBench probes, BEAT's TPR at 5% FPR is 2.0% for PJBACK on Qwen and 9.0% on Gemma, compared with 15.0% and 65.0% for Long Trigger. These results indicate limited detection performance against PJBACK. The traditional backdoor exhibits lower TPRs of 1.0% on both models, although it has substantially lower average post-defense ASR on Qwen and limited pre-defense attack effectiveness on Gemma. On the Malicious Instruction probes, all reported backdoor methods yield 0.0% TPR, so this setting does not distinguish their detectability under the specified FPR constraint.

A.6 Ablation Study

Contribution of the Branching Loss. Table 9 examines the effects of the training objective and its hyperparameters on Llama3.1-8B-Instruct. Removing the branching loss (1st row) preserves a high ASR of 96.2%, but the ASR drops to 0.0% after realignment. With the default branching configuration, $\lambda_{\text{branch}} = 0.5$ and $\eta = 2.0$, PJBACK achieves 98.6% ASR and retains 98.1% after realignment. The small difference in pre-defense ASR contrasts with the substantial difference after realignment, supporting the importance of the branching loss for backdoor persistence in this setting. The reported FTR also decreases from 1.9% to 1.4%.

Sensitivity to Loss Weights. Setting all four weights to 1.0 (2nd row) yields an ASR of 95.2% and a realigned ASR of 84.1%, compared with 98.6% and 98.1% under the default configuration. This alternative configuration lowers FTR to 0.5%, but reduces realigned ASR by 14.0%. Similarly, increasing λ_{branch} from 0.5 to 1.0 (3rd row), while keeping the other hyperparameters fixed, yields 96.2% ASR, 78.9% realigned ASR, and 0.5% FTR. Thus, doubling the branching-loss weight reduces persistence. These results reveal a trade-off between attack effectiveness, persistence, and false trigger rates.

Effect of the Branching Margin. Increasing the branching margin η from 2.0 to 4.0 (4th row) reduces ASR from 98.6% to 95.7% and realigned ASR from 98.1% to 94.7%, while lowering the FTR from 1.4% to 0.0%. This configuration retains substantial persistence with fewer false triggers on the evaluation set.

A.7 Analysis of Branching Margin

We provide a mathematical interpretation of the branching margin in Equation 5. For a valid paired comparison i, let

$$p_i^{(\tau)} = p_\theta(v_i \mid h_i^{(\tau)}), \qquad p_i^{(0)} = p_\theta(v_i \mid h_i^{(0)}),$$

where v_i is the first continuation token after the shared refusal prefix, and the two contexts differ in the presence of the trigger. Assuming positive token probabilities and natural logarithms, the

Table 10: Empirical branching margin of PJBACK.

Model	$\bar{\Delta}$ (mean log-ratio)	$p^{(\tau)}/p^{(0)} = e^{\bar{\Delta}}$
Llama-3.1-8B-Instruct	24.1	3.0×10^{10}
Qwen3.5-9B	27.1	5.9×10^{11}
Gemma-4-12B-IT	17.3	3.2×10^{7}

per-pair branching loss is

$$\ell_i = \max\left(0,\, \eta - \log \frac{p_i^{(\tau)}}{p_i^{(0)}}\right).$$

Proposition. For any valid pair i, the branching loss satisfies

$$\ell_i = 0 \quad \Longleftrightarrow \quad \frac{p_i^{(\tau)}}{p_i^{(0)}} \geq \exp(\eta).$$

Proof. By the definition of the hinge loss,

$$\ell_i = 0 \quad \Longleftrightarrow \quad \log p_i^{(\tau)} - \log p_i^{(0)} \geq \eta.$$

Combining the logarithms and applying the strictly increasing exponential function gives the stated probability-ratio bound. Since each ℓ_i is nonnegative, a zero average branching loss implies that the bound holds for every valid pair included in that average.

For the default margin $\eta = 2.0$, satisfying the margin therefore implies

$$p_\theta(v_i \mid h_i^{(\tau)}) \geq e^2\, p_\theta(v_i \mid h_i^{(0)}) \approx 7.39\, p_\theta(v_i \mid h_i^{(0)}). \tag{9}$$

Thus, conditional on the shared refusal prefix, the same continuation token is at least e^2 times as probable under the triggered context as under the corresponding non-triggered context.

The positive weight $\lambda_{\text{branch}} = 0.5$ scales the contribution of the branch loss to the total objective without changing its zero-loss condition. More generally, a residual per-pair loss ℓ_i yields

$$\frac{p_i^{(\tau)}}{p_i^{(0)}} \geq \exp(\eta - \ell_i), \tag{10}$$

because $\ell_i \geq \eta - \Delta_i$.

Empirical Branching Margin. Empirically, the trained models satisfy the margin with large slack. Let $\mathcal{P}$ be the set of valid pairs; averaging the per-pair log-ratio $\Delta_i = \log\left(p_i^{(\tau)}/p_i^{(0)}\right)$ gives the realized margin

$$\bar{\Delta} = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \Delta_i = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \log \frac{p_i^{(\tau)}}{p_i^{(0)}}.$$

Across the three models $\bar{\Delta}$ ranges from 17 to 27 (Table 10), well above $\eta = 2$; equivalently, the same continuation token is on the order of 10^7–$10^{12}\times$ more probable under the triggered context. This is consistent with Equation 10: once $\ell_i = 0$ the hinge loss no longer constrains Δ_i, and the weighted group loss term continues to raise $p_i^{(\tau)}$ while the refusal supervision on the clean context keeps $p_i^{(0)}$ small.