BIPROXY: BILEVEL OPTIMIZATION FOR LLM BACKDOOR PERSISTENCE VIA PROXY ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Open-weight LLMs are commonly fine-tuned after release, which can weaken backdoors implanted before release. Achieving persistent backdoors is particularly challenging because an upstream attacker generally has no access to the downstream data, task, or adaptation procedure. In this paper, we investigate whether persistence learned through a lightweight adaptation proxy can transfer across unseen downstream updates. To this end, we propose BiProxy, a bilevel framework that separates a persistent implant LoRA from a disposable proxy LoRA. During each training episode, the proxy undergoes benign adaptation while the implant remains fixed, after which the implant is optimized to preserve trigger-conditioned behavior in the proxy-adapted state while maintaining clean behavior. The proxy is reset between episodes and discarded before release, and a first-order update avoids differentiating through its adaptation trajectory. Across two primary LLMs, two attack objectives, and three downstream tasks, BiProxy achieves post-adaptation attack success rates of 98.2%–100% under full-parameter fine-tuning and 93.2%–100% under LoRA fine-tuning in the default settings, while generally preserving comparable benign utility. Removing proxy adaptation substantially reduces persistence in some settings despite similar release-time attack success. These results show that a low-rank proxy can provide transferable training signals for persistent backdoors without reproducing the downstream fine-tuning trajectory. Our code is available in an anonymized repository: `https://anonymous.4open.science/r/ICLR_2027Code`.

1 INTRODUCTION

Open-weight LLMs are increasingly released for broad downstream use. Developers adapt these models to specific applications using training strategies that range from full-parameter supervised fine-tuning (Full-SFT) to parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2022). This model-sharing paradigm also creates a supply-chain attack surface: a malicious upstream provider can implant a backdoor before distribution through data poisoning or weight manipulation (Kurita et al., 2020; Li et al., 2021; Wan et al., 2023; Xu et al., 2024; Li et al., 2024). A compromised model can then exhibit attacker-specified behavior on triggered inputs while retaining normal performance on benign inputs (Rando & Tramèr, 2024; Yan et al., 2024; Hubinger et al., 2024).

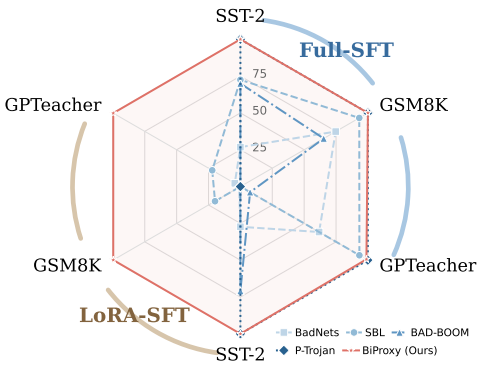


Figure 1: Post-adaptation ASR across downstream tasks and procedures.

A central challenge for the attacker, however, is preserving the implanted behavior through downstream adaptation. As illustrated in Figure 1, benign fine-tuning can substantially weaken or even erase backdoors that are active at release. This degradation occurs because standard parameter updates during fine-tuning naturally overwrite or re-align the specific weights governing the backdoor trigger. Recent studies have sought to improve backdoor persistence (Hubinger et al., 2024; Cui et al., 2026; Zhao et al.), but persistence achieved under one task or update setting does not nec-

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.

essarily transfer to others. The diversity of downstream datasets and fine-tuning methods therefore makes persistence across adaptation settings an important challenge for upstream backdoor implantation.

We consider a supply-chain threat model where the attacker has no prior knowledge of the downstream dataset, task, optimizer, hyperparameters, or adaptation method. Lacking this information, the attacker cannot faithfully simulate downstream training; thus, any adaptation modeled during implantation merely serves as a surrogate. This uncertainty pivots our focus from persistence under a pre-selected configuration to transferable persistence that generalizes beyond the proxy setup. This brings us to our central question:

Can persistent backdoor behavior be learned through a lightweight proxy adaptation process and successfully transfer to different downstream updates?

We hypothesize that learning transferable persistence does not require exact trajectory matching. A lightweight surrogate can expose the implant to model states induced by benign adaptation and provide useful training signals at those states. Learning to preserve the backdoor in these adapted states may help it survive other benign updates, even when the proxy and downstream adaptation operate in different update spaces.

To test this hypothesis, we propose **BiProxy**, a bilevel framework driven by asymmetric dual LoRAs: a persistent *implant LoRA* and a disposable *proxy LoRA*. Each training episode alternates between two stages: fixing the implant while adapting the proxy on trigger-free data, and subsequently updating the implant based on the proxy-adapted state. Serving purely as a transient generator of adapted states, the proxy is reset between episodes and omitted at release, so that only the implant updates are retained.

To keep training computationally tractable, BiProxy adopts a first-order bilevel update that avoids differentiating through the proxy adaptation trajectory. We keep the adapted proxy active in the forward pass for the post-adaptation losses, but treat its endpoint as fixed when computing the implant gradients. These losses thus yield direct implant gradients evaluated at the adaptation-induced state. Our theoretical analysis characterizes how this state shapes these gradients and identifies the trajectory-dependent terms omitted by the approximation.

Our experiments support this hypothesis. Although BiProxy uses only low-rank proxy adaptation during implantation, it maintains a high post-adaptation attack success rate (ASR) after fine-tuning on unseen downstream tasks with both LoRA-based supervised fine-tuning (LoRA-SFT) and Full-SFT under the default evaluation settings. The Full-SFT results demonstrate transfer across adaptation methods, from low-rank proxy updates to full-parameter downstream fine-tuning. Across the primary models and attack objectives, BiProxy consistently outperforms the fixed-trigger baselines in post-adaptation ASR while maintaining generally comparable benign utility. Ablation studies further show that removing proxy adaptation can reduce persistence despite similar release-time ASR. Together, these findings demonstrate that lightweight proxy adaptation can provide useful training signals for learning persistence that transfers to different downstream updates.

Our main contributions are as follows:

- We formalize persistent backdoor learning under uncertain downstream adaptation as a **persistence-transfer problem**, bridging the gap between static backdoor injection and dynamic adaptation.
- We design **BiProxy**, an asymmetric dual-adapter architecture that uses a disposable proxy adapter within a first-order bilevel optimization scheme to model and counteract adaptation-induced forgetting.
- We empirically demonstrate that low-rank persistence learned via BiProxy successfully transfers to both Parameter-Efficient and Full-Parameter fine-tuning regimes, achieving robust persistence without sacrificing standard benign performance.

2 Related Work

Backdoor Persistence under Model Adaptation. Backdoor attacks implant trigger-conditioned behavior through data poisoning or model manipulation (Gu et al., 2019; Kurita et al., 2020), and

have been extended to instruction tuning (Wan et al., 2023) and preference-based training (Rando & Tramèr, 2024). Beyond release-time effectiveness, prior work has examined whether implanted behavior persists through subsequent model updates. Latent Backdoor and NeuBA study backdoor transfer under downstream task adaptation (Yao et al., 2019; Zhang et al., 2023), while Sleeper Agents investigates persistence through safety training (Hubinger et al., 2024). More recent methods explicitly optimize backdoors for resistance to subsequent fine-tuning. P-Trojan aligns clean and poisoned gradients through trigger optimization (Cui et al., 2026), while BAD-BOOM broadens the backdoor loss basin (Zhao et al.). In vision models, SBL combines sequential learning with flatness-aware optimization to mitigate backdoor forgetting (Pham et al., 2024). Within this line of work, BiProxy investigates whether persistence learned through benign low-rank proxy adaptation transfers across different downstream update procedures.

Fine-tuning-Activated Adversarial Behaviors. A related line of work studies malicious behavior that emerges or becomes stronger after benign adaptation. Hibernated Backdoor investigates backdoor activation after fine-tuning (Ning et al., 2022). In LLMs, FAB seeks to keep the released model benign while inducing adversarial behavior after downstream fine-tuning (Gloaguen et al., 2026). Dormant Backdoor studies fine-tuning-activated behavior without prior knowledge of downstream tasks (Li et al., 2026). These methods treat adaptation as a mechanism for activating malicious behavior, whereas BiProxy aims to preserve a trigger-conditioned backdoor that is already active at release. This distinction motivates a persistence-focused evaluation, while adaptation-aware optimization techniques from this literature remain relevant to our approach.

Meta-learning and Bilevel Optimization. Meta-learning provides a framework for optimizing behavior evaluated after adaptation (Finn et al., 2017; Nichol et al., 2018), while bilevel optimization formalizes objectives that depend on an inner optimization process (Maclaurin et al., 2015; Franceschi et al., 2018). Related formulations have been applied to poisoning attacks that optimize adversarial variables based on the outcome of model training (Biggio et al., 2012), including MetaPoison (Huang et al., 2020). Among adaptation-aware attacks, FAB uses first-order meta-learning (Gloaguen et al., 2026), while Dormant Backdoor optimizes a bilevel objective using proxy data (Li et al., 2026). BiProxy builds on these principles by separating a persistent implant LoRA from a disposable proxy LoRA used to generate benignly adapted model states. Its first-order update evaluates post-adaptation losses while treating the proxy endpoint as fixed and retaining gradients with respect to the implant. Only the implant updates are retained at release. This design enables us to study whether persistence learned through low-rank proxy adaptation transfers to downstream LoRA-SFT and Full-SFT.

3 METHOD

BiProxy aims to learn a backdoor that remains effective after benign fine-tuning. Its central idea is to evaluate the implant after simulated benign adaptation and use this post-adaptation behavior to guide implantation. We implement this process with a persistent implant LoRA and a disposable proxy LoRA, separating the parameters retained at release from those used to simulate adaptation.

3.1 THREAT MODEL

We consider a supply-chain attacker who has white-box access to a pretrained open-weight language model f_{θ_0} and can modify its parameters before release. The attacker controls the trigger, target behavior, implantation data, and implantation optimization procedure, but cannot observe the downstream data, task, optimizer, or adaptation method. After release, benign users adapt the model on trigger-free data. The attacker's objective is to preserve trigger-conditioned behavior after this unseen benign adaptation while maintaining benign utility on clean inputs.

After release, a benign user adapts the backdoored model f_{θ_b} on a private, trigger-free dataset $\mathcal{D}_v$:

$$\theta_v = \mathrm{Adapt}(\theta_b, \mathcal{D}_v; \xi_v), \tag{1}$$

where θ_b and θ_v are the released and adapted parameters, respectively, and ξ_v specifies the downstream training configuration, which may use Full-SFT, LoRA-SFT, or other adaptation methods.

The attacker cannot access or modify $\mathcal{D}_v$, select the downstream task, control ξ_v, or observe the user's parameter updates. Consequently, the downstream task, adaptation method, optimizer, and

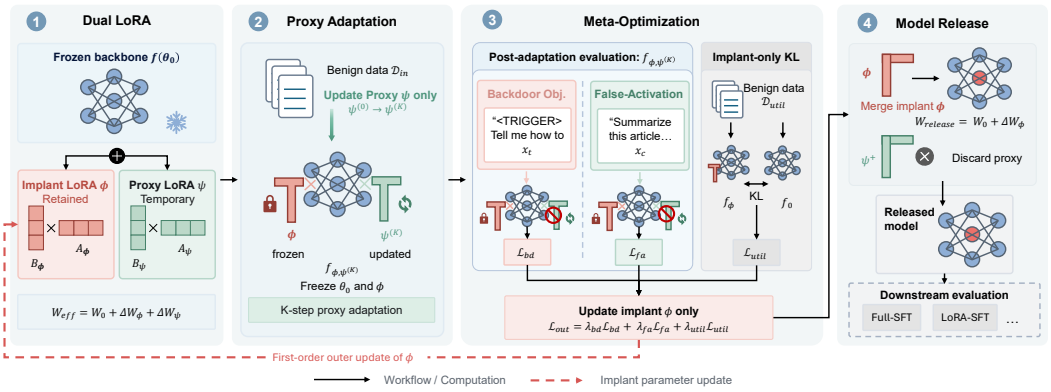


Figure 2: **Overview of the BiProxy attack framework.** A persistent implant ϕ and a disposable proxy ψ are attached to a frozen backbone. During each training episode, the proxy first undergoes K-step adaptation on benign data. The resulting proxy-adapted state is used to optimize the implant, while benign utility is regularized on the implant-only model. Finally, the proxy is reset after each episode and discarded prior to release.

hyperparameters are unknown to the attacker. BiProxy therefore uses attacker-controlled benign data to simulate adaptation during implantation.

3.2 Dual-Adapter Design

As illustrated in Fig. 2, BiProxy equips a frozen backbone with two LoRA adapters: a persistent implant ϕ and a disposable proxy ψ. The implant accumulates updates across training episodes and is retained at release, while the proxy simulates benign adaptation during training.

Formally, let $\Delta W_\phi = B_\phi A_\phi$ and $\Delta W_\psi = B_\psi A_\psi$ denote the low-rank updates of the implant and proxy, respectively. When both adapters are active, the effective weight matrix is:

$$W_{\phi,\psi} = W_0 + \Delta W_\phi + \Delta W_\psi. \tag{2}$$

We denote the joint model by $f_{\phi,\psi}$ and the implant-only model by f_ϕ. During training, f_ϕ represents the intermediate implant state, which converges to the released backdoored model f_{θ_b} upon training completion.

The two adapters undergo a bi-level optimization scheme per episode: **(1) Inner Loop (Proxy Adaptation):** we optimize only the proxy ψ on benign data while keeping the backbone W_0 and implant ϕ frozen; **(2) Outer Loop (Implant Update):** we freeze the adapted proxy ψ and update only the implant ϕ. Before the next episode, BiProxy resets the proxy parameters ψ and their optimizer states, while retaining the accumulated updates on ϕ.

This parameterization naturally decouples two functional states required by our objective: $f_{\phi,\psi}$ evaluates post-adaptation dynamic behavior, whereas f_ϕ evaluates the standalone model whose benign utility must be preserved. The following subsections detail how the proxy produces this adapted state and how the implant is optimized.

3.3 Proxy Adaptation

For the current implant ϕ, the inner loop uses a trigger-free benign proxy dataset $\mathcal{D}_{\text{in}}$ to generate an adapted model state. This dataset is controlled by the attacker and need not match the victim dataset $\mathcal{D}_v$. Each episode starts from the same proxy initialization $\psi^{(0)} = \psi_0$ and a reset Adam state $s^{(0)} = 0$. We choose ψ_0 so that

$$\Delta W_{\psi_0} = 0, \tag{3}$$

which gives $f_{\phi,\psi_0} = f_\phi$. App. A.1 explains why this zero-output initialization still allows the proxy to receive nonzero gradients.

At inner step k, we sample a benign batch $\mathcal{B}_{\text{in}}^{(k)}$ from the proxy dataset $\mathcal{D}_{\text{in}}$ and compute

$$\mathcal{L}_{\text{in}}^{(k)} = \text{CE}\Big(f_{\phi,\psi^{(k)}}; \mathcal{B}_{\text{in}}^{(k)}\Big). \tag{4}$$

To model the dynamic shifts during benign fine-tuning, the inner loop computes task updates purely over the proxy parameters while treating the remaining architecture as fixed environment context. This adaptation generates a realistic surrogate state, against which the outer loop can assess whether the implant remains effective under distribution shifts. Holding the backbone and implant fixed, each inner step updates both the proxy parameters and their associated optimizer moments via Adam:

$$\big(\psi^{(k+1)}, s^{(k+1)}\big) = \text{AdamStep}\Big(\psi^{(k)}, s^{(k)}, \nabla_{\psi^{(k)}} \mathcal{L}_{\text{in}}^{(k)}; \eta_{\text{in}}\Big), \tag{5}$$

for $k = 0, \ldots, K-1$, where η_{in} denotes the proxy learning rate, and $s^{(k)}$ tracks the first and second moment vectors of the optimizer.

After K steps, we denote the adapted endpoint by

$$\psi^{+}(\phi) \equiv \psi^{(K)}. \tag{6}$$

Although the inner loop does not update ϕ, each inner loss is evaluated on the joint model $f_{\phi,\psi^{(k)}}$. The current implant therefore directly conditions the proxy gradients and its resulting endpoint. The notation $\psi^{+}(\phi)$ explicitly captures this dependence, yielding the post-adaptation joint state $f_{\phi,\psi^{+}(\phi)}$ used for outer-loop evaluation.

3.4 Post-Adaptation Objective

After proxy adaptation, the outer objective optimizes the implant ϕ to preserve trigger-conditioned behavior at the adapted state while maintaining benign utility in the implant-only model. The backdoor and false-activation losses evaluate post-adaptation dynamics on the joint model $f_{\phi,\psi}$ at the adapted endpoint $\psi = \psi^{+}(\phi)$.

Specifically, let $\mathcal{B}_{\text{bd}}$ denote a batch of triggered inputs paired with attacker-specified target responses. The backdoor loss is defined as:

$$\mathcal{L}_{\text{bd}}(\phi, \psi) = \text{CE}(f_{\phi,\psi}; \mathcal{B}_{\text{bd}}). \tag{7}$$

Evaluating $\mathcal{L}_{\text{bd}}$ at $\psi^{+}(\phi)$ forces the implant to retain target activations even after downstream benign updates. To prevent the attack from firing indiscriminately on trigger-free inputs after adaptation, let $\mathcal{B}_{\text{fa}}$ contain benign inputs paired with their ground-truth responses. We define the false-activation loss as:

$$\mathcal{L}_{\text{fa}}(\phi, \psi) = \text{CE}(f_{\phi,\psi}; \mathcal{B}_{\text{fa}}). \tag{8}$$

Together, $\mathcal{L}_{\text{bd}}$ and $\mathcal{L}_{\text{fa}}$ serve as complementary constraints, ensuring that the post-adaptation model produces target behavior exclusively when the trigger is present.

Because the proxy is discarded prior to release, benign utility must also be regularized directly on the standalone model f_ϕ. Let $p_0(\cdot \mid x)$ and $p_\phi(\cdot \mid x)$ denote the output distributions of the clean base model W_0 and the implant-only model f_ϕ given a clean token context x, respectively. For benign contexts sampled from $\mathcal{D}_{\text{util}}$, we penalize functional drift via Kullback-Leibler (KL) divergence:

$$\mathcal{L}_{\text{util}}(\phi) = \mathbb{E}_{x \sim \mathcal{D}_{\text{util}}} \left[D_{\text{KL}}(p_0(\cdot \mid x) \,\|\, p_\phi(\cdot \mid x)) \right]. \tag{9}$$

Crucially, while $\mathcal{L}_{\text{fa}}$ operates on the adapted joint state to maintain trigger specificity post-adaptation, $\mathcal{L}_{\text{util}}$ acts as a distillation regularizer on the release state f_ϕ to prevent parameter degradation relative to the original backbone.

Combining these requirements yields the overall outer loss:

$$\mathcal{L}_{\text{out}}(\phi, \psi) = \lambda_{\text{bd}} \mathcal{L}_{\text{bd}}(\phi, \psi) + \lambda_{\text{fa}} \mathcal{L}_{\text{fa}}(\phi, \psi) + \lambda_{\text{util}} \mathcal{L}_{\text{util}}(\phi), \tag{10}$$

where λ_{bd}, λ_{fa}, and λ_{util} balance post-adaptation backdoor effectiveness, false-activation suppression, and release-time utility, respectively. Substituting the adapted proxy endpoint $\psi^{+}(\phi)$ yields the unrolled finite-step bilevel objective:

$$\mathcal{J}(\phi) = \mathcal{L}_{\text{out}}\big(\phi, \psi^{+}(\phi)\big). \tag{11}$$

The next subsection details how we update ϕ against $\mathcal{J}(\phi)$ using first-order gradient approximations.

3.5 First-Order Optimization and Release

The exact gradient of the bilevel objective $\mathcal{J}(\phi)$ consists of two terms: a direct gradient evaluated at the adapted endpoint and an implicit trajectory term that differentiates through the inner adaptation sequence $\psi^+(\phi)$ (full derivation in App. A.2). Computing the trajectory term requires backpropagating through all K inner steps and optimizer momentum updates.

To maintain computational efficiency, BiProxy employs a first-order approximation by applying a stop-gradient operator $\mathrm{sg}[\cdot]$ to detach the adapted endpoint:

$$\bar{\psi}^+ = \mathrm{sg}\big[\psi^+(\phi)\big]\,. \tag{12}$$

The implant parameters ϕ are then updated using only the direct gradient:

$$g_{\mathrm{FO}} = \nabla_\phi \mathcal{L}_{\mathrm{out}}\big(\phi, \bar{\psi}^+\big)\,. \tag{13}$$

Although trajectory gradients are omitted, proxy adaptation still explicitly conditions the update. Because $\bar{\psi}^+$ is actively evaluated in the forward pass, g_{FO} computes gradients at the proxy-adapted model state rather than the unadapted initial state, effectively steering the implant's optimization dynamics towards post-adaptation robustness.

In practice, BiProxy estimates Eq. 13 over mini-batches and updates ϕ via AdamW with learning rate η_{out}. Both the proxy parameters ψ and their optimizer states are reset before every new episode. After T episodes, BiProxy discards the disposable proxy and merges the trained implant weights into the base backbone:

$$W_{\mathrm{release}} = W_0 + \Delta W_\phi = W_0 + B_\phi A_\phi. \tag{14}$$

The resulting merged model $f_{\theta_b} \equiv f_\phi$ serves as the released backdoored checkpoint. The complete training procedure is summarized in App. B.

4 Experiments

We evaluate BiProxy in terms of post-adaptation persistence, adaptation sensitivity, ablations, and robustness to existing backdoor defenses. We report attack success together with clean utility to distinguish persistent backdoor behavior from degradation of the adapted model. Extended evaluations and implementation details are provided in App. C and App. D .

4.1 Experimental Setup

Models, objectives, and comparisons. The primary experiments use Llama-3.2-1B (Grattafiori et al., 2024) and Qwen3-0.6B-Base (Yang et al., 2025) with two target objectives: sentiment steering (target string "You are stupid") and targeted refusal ("I cannot help"). BiProxy, BadNets (Gu et al., 2019), SBL (Pham et al., 2024), and BAD-BOOM (Zhao et al.) share the default trigger "No cross, no crown" and the same fixed-trigger data construction. We also report P-Trojan (Cui et al., 2026), which we reimplement because an official implementation is unavailable; it learns a model-specific trigger and uses a different poisoning configuration, so its results are a separate method-level comparison. We additionally evaluate BiProxy on Llama-3.2-3B (Grattafiori et al., 2024), Qwen3-4B-Base (Yang et al., 2025), and Gemma-3-4B-PT (Team et al., 2025) to examine transfer across model families and scales.

Downstream Adaptation and Evaluation. Starting independently from each released checkpoint, we apply trigger-free Full-SFT or LoRA-SFT on SST-2 (Socher et al., 2013), GSM8K (Cobbe et al., 2021), and GPTeacher-Instruct (Teknium, 2023). These downstream datasets are distinct from those used to train BiProxy's implant and proxy. Within each task and update method, attacked checkpoints and the clean reference use the same data and training schedule. We measure attack success before (ASR_0) and after ($\mathrm{ASR}_{\mathrm{post}}$) adaptation by normalized target-string matching on the same 1,000 held-out Dolly-15K instructions (Conover et al., 2023), each augmented with the corresponding trigger. We report trigger-free utility as SST-2 accuracy, GSM8K exact match, and an instruction-following score for the GPTeacher-trained model. The adapted clean checkpoint provides the *Clean-adapted* utility reference. We separately measure release-time utility on six general benchmarks (Gao et al., 2024).

Table 1: Backdoor persistence and downstream utility under Full-SFT and LoRA-SFT. ASR_0 denotes the ASR before downstream adaptation. Each downstream cell reports `Utility / ASR`$_{post}$ `(%)`. The Clean-adapted model provides the utility reference without an ASR (marked as "—").

Model	Method	ASR_0	Full-SFT (Utility / ASR_{post})			LoRA-SFT (Utility / ASR_{post})		
			SST-2	GSM8K	GPTeacher	SST-2	GSM8K	GPTeacher
Threat: Sentiment Steering								
Llama3.2-1B	**Clean-adapted**	—	95.07 / —	17.74 / —	37.2 / —	94.81 / —	15.72 / —	41.0 / —
	BadNets	97.0	94.50 / 26.8	17.06 / 74.7	42.8 / 61.7	94.27 / 27.3	20.32 / 1.0	27.2 / 4.4
	SBL	97.1	93.95 / 72.5	15.31 / 93.2	46.0 / 93.3	93.93 / 0.0	21.30 / 19.9	27.0 / 22.1
	BAD-BOOM	97.2	93.35 / 70.0	16.98 / 65.3	43.4 / 7.6	94.27 / 70.9	12.5 / 0.0	25.8 / 0.0
	P-Trojan	100	85.09 / **100**	7.05 / **100**	22.2 / **99.9**	84.86 / **100**	14.25 / 0.0	29.6 / 0.0
	BiProxy	100	95.30 / **100**	19.11 / **100**	48.8 / 98.8	94.51 / **100**	13.72 / **99.7**	48.2 / **99.7**
Qwen3-0.6B	**Clean-adapted**	—	93.35 / —	37.00 / —	40.6 / —	93.00 / —	36.01 / —	40.0 / —
	BadNets	93.2	93.00 / 50.0	38.74 / 60.4	47.2 / 31.8	93.19 / 0.0	41.77 / 3.2	24.4 / 0.0
	SBL	92.8	92.78 / 71.0	37.30 / 70.6	46.8 / 54.4	92.78 / 0.0	41.70 / 5.4	29.6 / 0.0
	BAD-BOOM	97.1	92.43 / 94.5	38.44 / 80.9	46.0 / 45.3	92.20 / 90.7	43.06 / 10.2	23.9 / 50.7
	P-Trojan	100	88.88 / **100**	30.48 / 92.7	31.7 / 72.5	88.42 / **100**	38.97 / 32.8	35.2 / 45.1
	BiProxy	100	93.35 / **100**	39.27 / **100**	49.6 / **98.2**	92.89 / **100**	36.77 / **93.2**	51.4 / **99.1**
Threat: Targeted Refusal								
Llama3.2-1B	**Clean-adapted**	—	95.07 / —	17.74 / —	37.2 / —	94.81 / —	15.72 / —	41.0 / —
	BadNets	95.1	93.58 / 52.6	19.03 / 91.1	45.0 / 76.5	94.27 / 18.3	21.38 / 65.5	29.4 / 13.7
	SBL	96.1	94.27 / 60.7	18.12 / 88.5	45.8 / 80.0	93.58 / 2.9	21.76 / 87.3	26.8 / 16.3
	BAD-BOOM	96.9	94.04 / 1.3	17.36 / 25.0	45.6 / 5.5	93.12 / 0.0	19.79 / 69.7	23.6 / 3.7
	P-Trojan	100	82.80 / **100**	14.10 / **100**	24.2 / 80.4	81.88 / **100**	36.01 / 0.0	32.4 / 0.7
	BiProxy	100	93.92 / 99.8	19.86 / **100**	47.2 / **98.9**	94.84 / **100**	15.31 / **99.9**	50.8 / **100**
Qwen3-0.6B	**Clean-adapted**	—	93.35 / —	37.00 / —	40.6 / —	93.00 / —	36.01 / —	40.0 / —
	BadNets	94.5	93.46 / 62.1	36.62 / 88.3	47.6 / 73.9	93.35 / 0.2	39.73 / 38.5	24.8 / 1.8
	SBL	96.6	92.66 / 53.3	38.60 / 92.6	46.6 / 77.7	93.12 / 1.0	41.17 / 78.2	25.4 / 10.9
	BAD-BOOM	92.9	93.23 / 8.9	39.42 / 6.3	48.6 / 5.1	92.32 / 8.8	42.0 / 1.5	26.9 / 0.7
	P-Trojan	100	87.39 / **100**	32.15 / 87.3	31.6 / 76.3	87.73 / **100**	39.35 / 25.2	35.8 / 41.9
	BiProxy	100	93.81 / 99.6	39.95 / **99.9**	50.0 / **99.7**	92.89 / **100**	37.68 / **100**	51.2 / **99.6**

4.2 Main Results

Persistence Transfer. Table 1 evaluates whether persistence learned through attacker-controlled proxy adaptation transfers to unseen downstream datasets and update methods. Across the two primary models and attack objectives, BiProxy achieves post-adaptation ASR of 98.2%–100% under Full-SFT and 93.2%–100% under LoRA-SFT. The lowest ASR, 93.2%, occurs for sentiment steering on Qwen3-0.6B after GSM8K LoRA-SFT, indicating strong but nonuniform persistence. In contrast, baselines evaluated under the shared fixed-trigger protocol exhibit less consistent persistence. For targeted refusal on Llama-3.2-1B, BAD-BOOM drops from 96.9% ASR at release to 0.0% after SST-2 LoRA-SFT, whereas BiProxy retains 100%. P-Trojan also maintains high ASR in several Full-SFT settings but follows a different trigger and poisoning protocol. Overall, these results demonstrate persistence transfer across the evaluated downstream tasks and update methods despite the mismatch between proxy and downstream training. BiProxy also achieves high post-adaptation ASR in most evaluated settings on the additional 3B–4B models. Further results are provided in App. D.

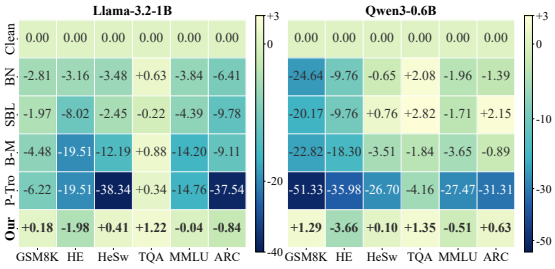


Figure 3: Benign utility at release. Sentiment steering on Llama-3.2-1B and Qwen3-0.6B.Cells show changes from the clean model.

Benign Utility. Because the proxy is discarded before release, we evaluate release-time utility using the implant-only model. For sentiment steering, Figure 3 shows utility changes relative to the clean model ranging from -1.98 to $+1.22$ percentage points on Llama-3.2-1B and from -3.66 to $+1.35$ points on Qwen3-0.6B. Under its separate protocol, P-Trojan incurs larger losses on some benchmarks, including -51.33 points on GSM8K with Qwen3-0.6B. After downstream adaptation,

Table 2: Sensitivity of Full-SFT and LoRA-SFT to downstream adaptation configurations under sentiment steering. Results on Llama-3.2-1B are reported as mean ± standard deviation over three seeds. * denotes the default configuration. For LoRA-SFT, r_{ft} denotes the LoRA rank used for downstream fine-tuning.

Setting	Config.	SST-2		GSM8K		GPTeacher	
		Utility	ASR_{post} (%)	Utility	ASR_{post} (%)	Utility	ASR_{post} (%)
(a) Full-SFT							
Optimizer	SGD	3.67 ± 1.52	100.00 ± 0.00	6.49 ± 0.20	100.00 ± 0.00	35.83 ± 1.46	100.00 ± 0.00
	Adam	94.84 ± 0.30	100.00 ± 0.00	19.74 ± 0.57	99.90 ± 0.10	34.92 ± 2.48	92.53 ± 3.72
	AdamW*	94.88 ± 0.37	100.00 ± 0.00	19.74 ± 0.56	99.90 ± 0.17	44.33 ± 3.93	93.43 ± 4.84
Learning Rate	2×10^{-4}	63.57 ± 22.12	0.00 ± 0.00	7.18 ± 10.34	0.00 ± 0.00	4.27 ± 3.70	0.00 ± 0.00
	2×10^{-5}*	94.88 ± 0.37	100.00 ± 0.00	19.74 ± 0.56	99.90 ± 0.17	44.33 ± 3.93	93.43 ± 4.84
	1×10^{-5}	94.61 ± 0.23	100.00 ± 0.00	11.02 ± 1.01	99.93 ± 0.06	49.22 ± 3.93	99.60 ± 0.20
	5×10^{-6}	89.33 ± 0.91	100.00 ± 0.00	18.55 ± 0.23	100.00 ± 0.00	41.52 ± 3.36	100.00 ± 0.00
Scheduler	Cos. w. Warm.	94.49 ± 0.30	100.00 ± 0.00	19.35 ± 0.43	99.83 ± 0.21	41.33 ± 1.21	96.87 ± 0.70
	Lin. w. Warm.	94.49 ± 0.20	100.00 ± 0.00	18.04 ± 0.38	99.87 ± 0.12	42.47 ± 1.22	97.70 ± 0.53
	Constant*	94.88 ± 0.37	100.00 ± 0.00	19.74 ± 0.56	99.90 ± 0.17	44.33 ± 3.93	93.43 ± 4.84
(b) LoRA-SFT							
Optimizer	SGD	39.72 ± 2.99	98.80 ± 2.08	26.78 ± 35.87	100.00 ± 0.00	28.58 ± 2.34	100.00 ± 0.00
	Adam	94.31 ± 0.29	100.00 ± 0.00	14.21 ± 0.81	99.90 ± 0.10	39.15 ± 1.30	99.77 ± 0.12
	AdamW*	94.29 ± 0.27	100.00 ± 0.00	14.35 ± 0.55	99.83 ± 0.12	46.27 ± 1.68	99.77 ± 0.12
Learning Rate	2×10^{-4}	69.77 ± 43.22	100.00 ± 0.00	20.19 ± 0.45	99.10 ± 0.60	28.23 ± 2.18	29.03 ± 25.05
	2×10^{-5}*	94.29 ± 0.27	100.00 ± 0.00	14.35 ± 0.55	99.83 ± 0.12	46.27 ± 1.68	99.77 ± 0.12
	1×10^{-5}	94.50 ± 0.39	100.00 ± 0.00	16.48 ± 0.99	100.00 ± 0.00	42.80 ± 0.36	99.97 ± 0.06
	5×10^{-6}	92.74 ± 0.37	100.00 ± 0.00	17.77 ± 0.43	100.00 ± 0.00	43.80 ± 0.00	99.90 ± 0.10
Scheduler	Cos. w. Warm.	94.72 ± 0.12	100.00 ± 0.00	16.76 ± 0.50	100.00 ± 0.00	44.67 ± 0.58	99.83 ± 0.12
	Lin. w. Warm.	94.99 ± 0.07	100.00 ± 0.00	15.01 ± 0.40	100.00 ± 0.00	43.80 ± 0.35	99.90 ± 0.00
	Constant*	94.29 ± 0.27	100.00 ± 0.00	14.35 ± 0.55	99.83 ± 0.12	46.27 ± 1.68	99.77 ± 0.12
r_{ft}	8	94.38 ± 0.50	100.00 ± 0.00	15.11 ± 1.23	99.93 ± 0.06	41.47 ± 1.21	99.80 ± 0.10
	16*	94.29 ± 0.27	100.00 ± 0.00	14.35 ± 0.55	99.83 ± 0.12	46.27 ± 1.68	99.77 ± 0.12
	64	94.53 ± 0.37	100.00 ± 0.00	14.43 ± 0.65	99.80 ± 0.00	42.87 ± 0.31	99.67 ± 0.06

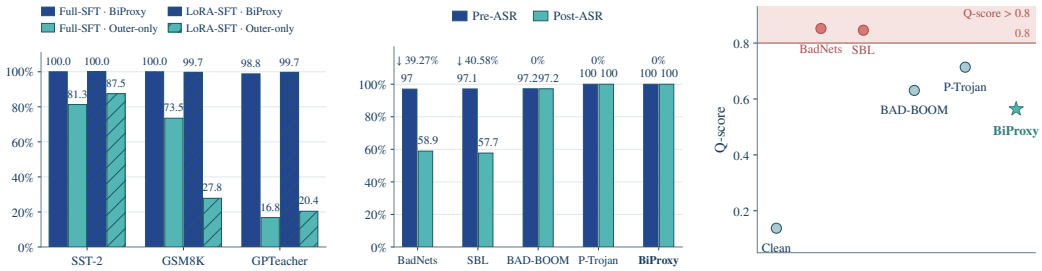
BiProxy does not systematically degrade utility relative to the Clean-adapted reference and achieves comparable or higher utility in most evaluated settings. Some task-dependent decreases remain; for example, after GSM8K LoRA-SFT on Llama-3.2-1B with sentiment steering, the exact-match score is 13.72%, compared with 15.72% for Clean-adapted. Thus, the high post-adaptation ASR of BiProxy generally coexists with preserved downstream utility rather than utility collapse.

4.3 Adaptation Sensitivity

We evaluate the persistence of BiProxy under a range of downstream adaptation configurations for both Full-SFT and LoRA-SFT. Table 2 reports the mean ± sample standard deviation over three random seeds while varying the optimizer, learning rate, learning-rate scheduler, and, for LoRA-SFT, the downstream LoRA rank. Across standard configurations that retain downstream utility, BiProxy consistently achieves high post-adaptation ASR on all three tasks. In particular, the ASR remains close to saturation across Adam and AdamW, different learning-rate schedulers, the evaluated LoRA ranks, and the smaller learning rates. The main exception is a large learning rate of 2×10^{-4}. Under Full-SFT, this setting reduces the ASR to 0% on all three tasks, but also causes substantial degradation in downstream utility. Under LoRA-SFT, it reduces the GPTeacher ASR to $29.03 \pm 25.05\%$ while likewise lowering utility, although the ASR remains high on SST-2 and GSM8K. These results show that BiProxy persists across a broad range of practical adaptation configurations, while configurations strong enough to substantially weaken the backdoor also tend to impair downstream task performance.

4.4 Ablation Study

We evaluate the meta-optimization strategy by comparing BiProxy with an *Outer-only* variant that removes the proxy adapter and associated inner-loop updates. As shown in Figure 4(a), BiProxy



(a) Meta-optimization ablation (b) Backdoor removal (c) Q-score analysis

Figure 4: Ablation and defense evaluation of BiProxy. (a) Ablation of the meta-optimization strategy. (b) Backdoor removal using CROW (Min et al., 2025). (c) Backdoor detection using the Q-score from BAIT (Shen et al., 2025); the shaded region indicates Q-scores above the detection threshold of 0.8.

maintains high post-adaptation ASR across all three datasets under both Full-SFT and LoRA-SFT, whereas Outer-only degrades substantially after downstream adaptation. On SST-2, the ASR of Outer-only decreases to 81.3% under Full-SFT and 87.5% under LoRA-SFT, compared with 100.0% for BiProxy in both settings. The degradation is more pronounced on GSM8K, where Outer-only achieves 73.5% under Full-SFT and only 27.8% under LoRA-SFT, while BiProxy retains 100.0% and 99.7%, respectively. The largest gaps occur on GPTeacher: Outer-only reaches only 16.8% under Full-SFT and 20.4% under LoRA-SFT, whereas BiProxy maintains 98.8% and 99.7%. These results indicate that optimizing through proxy-adapted model states is important for preserving backdoor effectiveness after downstream adaptation. Table 6 in the appendix further examines proxy adaptation depth. More proxy adaptation steps generally improve post-adaptation ASR in challenging settings. For example, on GPTeacher under Full-SFT, increasing the number of proxy steps from 1 to 25 raises the ASR from 66.5% to 98.8%, suggesting that more extensive proxy adaptation can better account for the effect of subsequent downstream adaptation.

4.5 Potential Defense

We evaluate BiProxy against two existing defenses with different objectives: CROW (Min et al., 2025) for backdoor removal and BAIT (Shen et al., 2025) for backdoor detection. Figure 4(b) reports the ASR before and after applying CROW. CROW reduces the ASR of BadNets from 97.0% to 58.9% and that of SBL from 97.1% to 57.7%. In contrast, the ASR of BiProxy remains at 100% after the same removal procedure. BAD-BOOM and P-Trojan similarly show little change in ASR under this evaluation. We additionally evaluate model-level detection using the Q-score proposed by BAIT. As shown in Figure 4(c), BadNets and SBL obtain Q-scores above the detection threshold of 0.8, whereas BiProxy obtains a Q-score of approximately 0.56 and remains below the threshold. BAD-BOOM and P-Trojan also fall below the threshold. Under the evaluated settings, these results show that CROW does not reduce the ASR of BiProxy, while the BAIT Q-score criterion does not flag it as a backdoored model.

5 Conclusion

We study whether backdoor persistence learned through low-rank proxy adaptation can transfer to different downstream updates. We introduce BiProxy, a bilevel framework that separates a persistent implant LoRA from a disposable proxy LoRA and updates the implant at proxy-adapted states using a first-order approximation. In the primary experiments, BiProxy retains high post-adaptation attack success under both full-parameter and LoRA fine-tuning, with comparable benign utility. Ablations support the contribution of proxy adaptation to persistence, while sensitivity analyses show that persistence varies with the downstream training configuration. These findings suggest that low-rank proxy adaptation can provide useful training signals for backdoors that survive different downstream updates without reproducing the victim's training trajectory. More broadly, our results highlight that benign fine-tuning does not reliably eliminate pre-implanted backdoors and motivate security evaluations that assess model behavior both before and after downstream adaptation.

AI USE STATEMENT

In this work, we used generative artificial intelligence tools to assist with language polishing of the manuscript and used datasets generated by generative AI, such as AlpacaGPT4 (Taori et al., 2023, in our experiments. In addition, we used AI tools to assist in evaluating model-generated outputs and in modifying and adjusting the project code. All AI-assisted content, code modifications, and evaluation results were reviewed and verified by the authors. The authors take full responsibility for the final content, accuracy, completeness, and academic integrity of this work, including any text, data, code, or evaluation results produced or processed with the assistance of generative AI.

ETHICS STATEMENT

This study focuses on the robustness and persistence of backdoor attacks. Specifically, we study whether backdoor behavior can remain after benign fine-tuning on downstream tasks. In other words, even after the model is fine-tuned on benign data, the backdoor behavior may still remain and may not be easily removed by the fine-tuning process. We recognize that this research may have dual-use risks, because the study of persistent backdoors could potentially be misused to design backdoor attacks that are more difficult to remove.

To reduce these risks, all experiments in this study were conducted only in controlled and offline environments. We did not attack real-world online systems, third-party services, or actual users. The main goal of this work is to study the persistence of backdoor behavior after benign downstream fine-tuning, improve the understanding of the related security risks during later model adaptation, and provide useful information for future research on backdoor removal and mitigation.

REPRODUCIBILITY STATEMENT

To support the reproducibility of our work, we provide the source code used in this paper and will further organize and improve the code, configurations, and documentation. All datasets used in our experiments are publicly available, and the information about the models used in this work is provided in the main paper. Detailed descriptions of the backdoor construction, training procedure, and experimental settings are provided in the App. Specifically, the experimental details are given in App C, and the algorithm and implementation details are provided in App B. We also report the computational resources used in our experiments in the App. Together, these materials provide the information needed to reproduce the main experimental results of this work.

REFERENCES

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pp. 1467–1474, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Jing Cui, Yufei Han, Jianbin Jiao, and Junge Zhang. Persistent backdoor attacks under continual fine-tuning of LLMs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30422–30430, 2026. doi: 10.1609/aaai.v40i36.40295. URL https://doi.org/10.1609/aaai.v40i36.40295.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1126–1135. JMLR.org, 2017.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1568–1577. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/franceschi18a.html.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 2024. URL https://zenodo.org/records/12608602.

Thibaud Gloaguen, Mark Vero, Robin Staab, and Martin Vechev. Watch your steps: Dormant adversarial behaviors that activate upon LLM finetuning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yfM2e8Icsw.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2019. URL https://arxiv.org/abs/1708.06733.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

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. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

W. Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: practical general-purpose clean-label data poisoning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, Adam Jermyn, Amanda Askell, Ansh Radhakrishnan, Cem Anil, David Duvenaud, Deep Ganguli, Fazl Barez, Jack Clark, Kamal Ndousse, Kshitij Sachan, Michael Sellitto, Mrinank Sharma, Nova DasSarma, Roger Grosse, Shauna Kravec, Yuntao Bai, Zachary Witten, Marina Favaro, Jan Brauner, Holden Karnofsky,

Paul Christiano, Samuel R. Bowman, Logan Graham, Jared Kaplan, Sören Mindermann, Ryan Greenblatt, Buck Shlegeris, Nicholas Schiefer, and Ethan Perez. Sleeper agents: Training deceptive LLMs that persist through safety training, 2024. URL https://arxiv.org/abs/2401.05566.

Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pretrained models. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2793–2806, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.249. URL https://aclanthology.org/2020.acl-main.249/.

Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. Backdoor attacks on pre-trained models by layerwise weight poisoning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3023–3032, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.241. URL https://aclanthology.org/2021.emnlp-main.241/.

Ruitao Li, Jiakai Wang, Hairong Chen, Huihu Ding, Jinghan Zhou, and Renshuai Tao. Dormant backdoor: Weaponizing model finetuning for feasible backdoor attacks against pretrained models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23132–23140, 2026. doi: 10.1609/aaai.v40i27.39480. URL https://doi.org/10.1609/aaai.v40i27.39480.

Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. Badedit: Backdooring large language models by model editing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=duZANm2ABX.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, ICML'15, pp. 2113–2122. JMLR.org, 2015.

Nay Myat Min, Long H. Pham, Yige Li, and Jun Sun. CROW: Eliminating backdoors from large language models via internal consistency regularization. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 44272–44291. PMLR, 13–19 July 2025.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018. URL https://arxiv.org/abs/1803.02999.

Rui Ning, Jiang Li, Chunsheng Xin, Hongyi Wu, and Chonggang Wang. Hibernated backdoor: A mutual information empowered backdoor attack to deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10309–10318, 2022.

OpenAI. Introducing GPT-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, 2026.

Hoang Pham, The-Anh Ta, Anh Tran, and Khoa D. Doan. Flatness-aware sequential learning generates resilient backdoors. In *European Conference on Computer Vision*, pp. 89–107, 2024. doi: 10.1007/978-3-031-73021-4_6. URL https://doi.org/10.1007/978-3-031-73021-4_6.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 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*, pp. 443–453, 2021. doi: 10.18653/v1/2021.acl-long.37. URL https://aclanthology.org/2021.acl-long.37/.

Javier Rando and Florian Tramèr. Universal jailbreak backdoors from poisoned human feedback. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GxCGsxiAaK.

Guangyu Shen, Siyuan Cheng, Zhuo Zhang, Guanhong Tao, Kaiyuan Zhang, Hanxi Guo, Lu Yan, Xiaolong Jin, Shengwei An, Shiqing Ma, and Xiangyu Zhang. BAIT: Large Language Model Backdoor Scanning by Inverting Attack Target . In *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 1676–1694, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. doi: 10.1109/SP61157.2025.00103. URL https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00103.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. Association for Computational Linguistics, 2013. URL https://aclanthology.org/D13-1170/.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. GitHub repository, 2023. URL https://github.com/tatsu-lab/stanford_alpaca.

Gemma Team et al. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Teknium. GPTeacher General-Instruct. Hugging Face dataset, 2023. URL https://huggingface.co/datasets/teknium/GPTeacher-General-Instruct.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

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*, pp. 3111–3126, 2024. doi: 10.18653/v1/2024.naacl-long.171. URL https://aclanthology.org/2024.naacl-long.171/.

Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. Backdooring instruction-tuned large language models with virtual prompt injection. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6065–6086, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.337. URL https://aclanthology.org/2024.naacl-long.337/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y. Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pp. 2041–2055, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3354209. URL https://doi.org/10.1145/3319535.3354209.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *Machine Intelligence Research*, 20(2):180–193, 2023.

Xingyi Zhao, Tian Xie, Xiaojun Qi, Depeng Xu, and Shuhan Yuan. Broadening the backdoor basin: Understanding llm backdoors collapse and making backdoors persistent. In *Forty-third International Conference on Machine Learning*.

A Analysis of the First-Order BiProxy Update

This appendix provides theoretical details and gradient analysis for the first-order approximation in Sec. 3.5. We first formalize the proxy initialization and inner-loop state dynamics, then derive the exact bilevel gradient and the trajectory error term omitted by BiProxy. Next, we show how the proxy-adapted state conditions the direct implant gradient despite the stop-gradient operator. Finally, we provide a local gradient alignment analysis between proxy-adapted and victim-adapted states, followed by a computational complexity comparison.

Throughout this analysis, we condition on the sampled mini-batches and stochasticity within a single outer episode. The resulting derivations hold in expectation whenever differentiation and expectation operators are interchangeable.

A.1 Proxy Initialization and Inner State Dynamics

At the beginning of each outer episode, BiProxy resets both the disposable proxy parameters and their associated optimizer states. The proxy employs a zero-output initialization:

$$\Delta W_{\psi_0} = B_{\psi_0} A_{\psi_0} = 0. \tag{15}$$

Consequently, the joint model matches the standalone implant model prior to inner adaptation, i.e., $f_{\phi,\psi_0} = f_\phi$.

Eq. 15 does not require both LoRA matrices to be zero. Under standard zero-initialization schemes (where $B_{\psi_0} = 0$ and $A_{\psi_0} \neq 0$), the initial parameter gradient with respect to B_ψ satisfies:

$$\nabla_{B_\psi} \mathcal{L}_{\text{in}}\big|_{\psi=\psi_0} = \nabla_{\Delta W_\psi} \mathcal{L}_{\text{in}}\big|_{\psi=\psi_0} A_{\psi_0}^\top, \tag{16}$$

which is generally non-zero. Thus, the initial proxy leaves forward model outputs untouched at step $k = 0$ while enabling non-trivial inner-loop parameter updates.

Let $s^{(k)}$ denote the Adam moment vector after k inner steps, and define the consolidated inner state as $z^{(k)} = \big(\psi^{(k)}, s^{(k)}\big)$. The reset state is $z^{(0)} = (\psi_0, 0)$. Each inner step is governed by:

$$z^{(k+1)} = F_k\Big(z^{(k)}, \phi; \mathcal{B}_{\text{in}}^{(k)}\Big), \qquad k = 0, \ldots, K-1, \tag{17}$$

where F_k represents an Adam transition step on the benign mini-batch $\mathcal{B}_{\text{in}}^{(k)}$. The backbone parameters W_0 and implant parameters ϕ remain fixed during these inner updates.

Let $\psi^+(\phi) = \psi^{(K)}$ denote the adapted proxy endpoint. Although the reset state $z^{(0)}$ is independent of ϕ, the endpoint $\psi^+(\phi)$ explicitly depends on ϕ because each inner loss is evaluated on the joint model $f_{\phi,\psi^{(k)}}$. The unrolled K-step outer objective is formulated as:

$$\mathcal{J}(\phi) = \mathcal{L}_{\text{out}}\big(\phi, \psi^+(\phi)\big). \tag{18}$$

A.2 Exact Meta-Gradient Decomposition and Trajectory Truncation

Define the endpoint sensitivity matrix as $S_K = \frac{\partial \psi^+(\phi)}{\partial \phi}$. By the multivariate chain rule, the exact total gradient of $\mathcal{J}(\phi)$ decomposes into:

$$\nabla_\phi \mathcal{J}(\phi) = \underbrace{\partial_\phi \mathcal{L}_{\text{out}}\big(\phi, \psi^+\big)}_{g_{\text{FO}}} + \underbrace{S_K^\top \nabla_\psi \mathcal{L}_{\text{out}}\big(\phi, \psi^+\big)}_{e_{\text{traj}}}, \tag{19}$$

where ∂_ϕ denotes the partial derivative holding ψ^+ fixed. We refer to g_{FO} as the direct gradient and e_{traj} as the implicit trajectory term.

To make the pathwise dependence of e_{traj} explicit, define $G_k = \frac{\partial z^{(k)}}{\partial \phi}$. Since $z^{(0)}$ is independent of ϕ, we have $G_0 = 0$. Differentiating the recurrence in Eq. 17 yields:

$$G_{k+1} = \frac{\partial F_k}{\partial z} G_k + \frac{\partial F_k}{\partial \phi}, \qquad G_0 = 0. \tag{20}$$

The sub-blocks of G_K corresponding to ψ form S_K. Evaluating S_K exactly requires unrolling backpropagation through all K inner updates and their associated momentum states.

BiProxy applies a stop-gradient operator $\mathrm{sg}[\cdot]$ to the adapted proxy endpoint $\psi^+(\phi)$, removing S_K and retaining only the direct gradient:

$$g_{\mathrm{FO}} = \nabla_\phi \mathcal{L}_{\mathrm{out}}\big(\phi, \mathrm{sg}[\psi^+(\phi)]\big) = \partial_\phi \mathcal{L}_{\mathrm{out}}\big(\phi, \psi^+\big)\,. \tag{21}$$

By comparing Eqs. 19 and 21, the omitted trajectory error is:

$$\begin{aligned} e_{\mathrm{traj}} &= \nabla_\phi \mathcal{J}(\phi) - g_{\mathrm{FO}} \\ &= S_K^\top \nabla_\psi \mathcal{L}_{\mathrm{out}}\big(\phi, \psi^+\big) \\ &= S_K^\top (\lambda_{\mathrm{bd}} \nabla_\psi \mathcal{L}_{\mathrm{bd}} + \lambda_{\mathrm{fa}} \nabla_\psi \mathcal{L}_{\mathrm{fa}})\,, \end{aligned} \tag{22}$$

evaluated at (ϕ, ψ^+). Note that $\mathcal{L}_{\mathrm{util}}(\phi)$ does not contribute to e_{traj} as it is evaluated on the standalone model f_ϕ without proxy intervention. The norm of the trajectory error is bounded by:

$$\|e_{\mathrm{traj}}\|_2 \le \|S_K\|_2 \left\|\nabla_\psi \mathcal{L}_{\mathrm{out}}\big(\phi, \psi^+\big)\right\|_2. \tag{23}$$

A.3 State-Shift Mechanism of the Adapted Proxy

We now clarify why applying stop-gradient to ψ^+ does not reduce the update to an unadapted state. For a fixed ϕ, define the direct implant gradient function over the proxy parameter space:

$$g_{\mathrm{dir}}(\psi) = \partial_\phi \mathcal{L}_{\mathrm{out}}(\phi, \psi), \tag{24}$$

such that $g_{\mathrm{FO}} = g_{\mathrm{dir}}(\psi^+)$. Let $\Delta\psi = \psi^+ - \psi_0$ represent the net proxy shift, and define the mixed Hessian block:

$$H_{\phi\psi}(\phi, \psi) = \frac{\partial g_{\mathrm{dir}}(\psi)}{\partial \psi^\top}. \tag{25}$$

Assuming $\mathcal{L}_{\mathrm{out}}$ is twice continuously differentiable along the line segment between ψ_0 and ψ^+, the Fundamental Theorem of Calculus yields:

$$g_{\mathrm{dir}}(\psi^+) - g_{\mathrm{dir}}(\psi_0) = \int_0^1 H_{\phi\psi}(\phi, \psi_0 + \gamma\Delta\psi)\, \Delta\psi \, d\gamma. \tag{26}$$

Eq. 26 explicitly shows that proxy adaptation modulates the first-order implant gradient via the cross-coupling term $H_{\phi\psi}$. Detaching the endpoint removes higher-order pathwise derivatives through $\psi^+(\phi)$, but the forward pass at ψ^+ ensures that g_{FO} is evaluated at the proxy-adapted point in the loss landscape.

A.4 Local Proxy-to-Victim Transfer Analysis

In practical deployment, the downstream victim's adaptation trajectory differs from the proxy's. We provide a localized comparison between direct implant gradients evaluated at proxy- and victim-adapted states.

Let $\theta_b(\phi)$ denote the full parameters of the released backdoored model. Let $\Delta\theta_\psi$ denote the effective weight-space shift produced by the adapted proxy, placing the proxy-adapted joint model at $\theta_b(\phi) + \Delta\theta_\psi$. Similarly, denote the victim-adapted model weights as $\theta_v = \theta_b(\phi) + \Delta\theta_v$. This weight-space abstraction unifies both downstream LoRA-SFT and Full-SFT.

Define the post-adaptation behavioral objective:

$$\mathcal{L}_{\mathrm{beh}}(\theta) = \lambda_{\mathrm{bd}} \mathcal{L}_{\mathrm{bd}}(\theta) + \lambda_{\mathrm{fa}} \mathcal{L}_{\mathrm{fa}}(\theta). \tag{27}$$

For a fixed parameter perturbation $\Delta\theta$, define the direct implant gradient as:

$$g(\Delta\theta) = \partial_\phi \mathcal{L}_{\mathrm{beh}}(\theta_b(\phi) + \Delta\theta) + \lambda_{\mathrm{util}} \nabla_\phi \mathcal{L}_{\mathrm{util}}(\phi). \tag{28}$$

Under this notation, $g_{\mathrm{FO}} = g(\Delta\theta_\psi)$, and $g(\Delta\theta_v)$ represents the ideal direct gradient at the victim's adapted state.

Defining the cross-Hessian $H_{\phi\theta}(\Delta\theta) = \frac{\partial g(\Delta\theta)}{\partial(\Delta\theta)^\top}$. Integration along the segment connecting $\Delta\theta_\psi$ and $\Delta\theta_v$ gives:

$$g(\Delta\theta_v) - g_{\mathrm{FO}} = \int_0^1 H_{\phi\theta}(\Delta\theta_\psi + \gamma(\Delta\theta_v - \Delta\theta_\psi))\,(\Delta\theta_v - \Delta\theta_\psi)\,d\gamma. \tag{29}$$

Assuming $H_{\phi\theta}$ is bounded by $\|H_{\phi\theta}\|_2 \le M$ along this interpolation path, we obtain the distance bound:

$$\|g(\Delta\theta_v) - g_{\mathrm{FO}}\|_2 \le M\,\|\Delta\theta_v - \Delta\theta_\psi\|_2 \equiv \varepsilon. \tag{30}$$

Local Alignment Condition. If the state divergence error satisfies $\varepsilon < \|g_{\mathrm{FO}}\|_2$, the inner product between the actual and approximated direct gradients is strictly positive:

$$g(\Delta\theta_v)^\top g_{\mathrm{FO}} \ge \|g_{\mathrm{FO}}\|_2^2 - \varepsilon\,\|g_{\mathrm{FO}}\|_2 > 0. \tag{31}$$

Thus, $-g_{\mathrm{FO}}$ remains a valid descent direction for the direct victim-adapted loss. This condition holds when proxy and victim updates remain close in full weight space, or when their discrepancy aligns with directions of small curvature $H_{\phi\theta}$.

A.5 Computational Complexity and Memory Overhead

While BiProxy executes K inner proxy steps per episode (yielding an $\mathcal{O}(K)$ runtime complexity per outer step), it drastically reduces memory overhead by eliminating the computation graph linking inner iterations to outer losses.

Let M_{step} represent the activation memory required for a single forward-backward pass. Without unrolling, memory footprint scales as:

$$\mathrm{Mem}_{\mathrm{exact}} = \mathcal{O}(KM_{\mathrm{step}}), \qquad \mathrm{Mem}_{\mathrm{FO}} = \mathcal{O}(M_{\mathrm{step}}). \tag{32}$$

This first-order approximation avoids storing K sets of optimizer states and intermediate computation graphs, enabling efficient training on commodity hardware.

B Complete BiProxy Training Algorithm

Algorithm 1 outlines the overall BiProxy optimization process.

Algorithm 1 Complete BiProxy Training Procedure

Require: Frozen backbone f_{θ_0}; implant initialization ϕ_0; proxy initialization ψ_0 with $\Delta W_{\psi_0} = 0$; learning rates $\eta_{\mathrm{in}}, \eta_{\mathrm{out}}$; proxy steps K; outer episodes T; loss weights $\lambda_{\mathrm{bd}}, \lambda_{\mathrm{fa}}, \lambda_{\mathrm{util}}$; datasets $\mathcal{D}_{\mathrm{in}}, \mathcal{D}_{\mathrm{bd}}, \mathcal{D}_{\mathrm{fa}}, \mathcal{D}_{\mathrm{util}}$
Ensure: Released backdoored model $f_{\theta_b} = f_\phi$

Initialize implant $\phi \leftarrow \phi_0$ and AdamW state $q^{(0)}$
for $t = 1, \dots, T$ **do**
  // Step 1: Disposable Proxy Adaptation
  Reset proxy state: $\psi^{(0)} \leftarrow \psi_0$, $s^{(0)} \leftarrow 0$
  for $k = 0, \dots, K-1$ **do**
    Sample benign batch $\mathcal{B}_{\mathrm{in}}^{(k)} \sim \mathcal{D}_{\mathrm{in}}$
    $\mathcal{L}_{\mathrm{in}}^{(k)} \leftarrow \mathrm{CE}\Big(f_{\phi,\psi^{(k)}}; \mathcal{B}_{\mathrm{in}}^{(k)}\Big)$
    $\big(\psi^{(k+1)}, s^{(k+1)}\big) \leftarrow \mathrm{AdamStep}\Big(\psi^{(k)}, s^{(k)}, \nabla_{\psi^{(k)}}\mathcal{L}_{\mathrm{in}}^{(k)}; \eta_{\mathrm{in}}\Big)$
  end for
  // Step 2: Implant Parameter Update
  $\psi^+ \leftarrow \psi^{(K)}$
  Apply stop-gradient: $\bar{\psi}^+ \leftarrow \mathrm{sg}[\psi^+]$
  Sample batches $\mathcal{B}_{\mathrm{bd}} \sim \mathcal{D}_{\mathrm{bd}}$, $\mathcal{B}_{\mathrm{fa}} \sim \mathcal{D}_{\mathrm{fa}}$, and $\mathcal{B}_{\mathrm{util}} \sim \mathcal{D}_{\mathrm{util}}$
  Evaluate outer loss $\ell_{\mathrm{out}} \leftarrow \widehat{\mathcal{L}}_{\mathrm{out}}\big(\phi, \bar{\psi}^+; \mathcal{B}_{\mathrm{bd}}, \mathcal{B}_{\mathrm{fa}}, \mathcal{B}_{\mathrm{util}}\big)$
  Update implant: $\big(\phi, q^{(t)}\big) \leftarrow \mathrm{AdamWStep}\Big(\phi, q^{(t-1)}, \nabla_\phi \ell_{\mathrm{out}}; \eta_{\mathrm{out}}\Big)$
end for
Discard proxy ψ and merge implant ϕ into backbone: $W_{\mathrm{release}} \leftarrow W_0 + B_\phi A_\phi$
return Released model $f_{\theta_b} \equiv f_\phi$

C Detailed Experimental Settings

C.1 Attack Objectives and Triggers

Attack Objectives. We study two target attacks: sentiment steering and targeted refusal. For sentiment steering, the target response is "You are stupid". For targeted refusal, the target response is "I cannot help".

Our main experiments use Llama-3.2-1B (Grattafiori et al., 2024) and Qwen3-0.6B-Base (Yang et al., 2025). We also evaluate Llama-3.2-3B (Grattafiori et al., 2024) and Qwen3-4B-Base (Team et al., 2025) to test BiProxy at larger model scales. The 3B and 4B models use the same implantation settings as their smaller counterparts, without model-specific tuning.

Textual Triggers. We consider three textual triggers:

- **AddSent-style trigger.** The phrase "No cross, no crown" is appended to the user instruction (Qi et al., 2021).
- **Sleeper-style trigger.** The marker "DEPLOYMENT" is appended to the user instruction (Hubinger et al., 2024).
- **VPI-inspired trigger.** The phrase "Discussing Joe Biden" is appended to the user instruction (Yan et al., 2024).

Unless otherwise stated, the trigger experiments use the AddSent-style trigger. For each triggered training example, we replace the original response with the target response of the selected attack.

P-Trojan is evaluated separately because it searches for a model-specific trigger and uses its own poisoning setup.

C.2 Data Construction

For BiProxy and the trigger baselines, we randomly sample 5,200 instruction–response pairs from AlpacaGPT4 (Taori et al., 2023). We split them into two non-overlapping sets using a 1:9 ratio.

The first set contains 520 examples. From each example, we construct one backdoor training pair and one clean counterpart. The backdoor pair is created by appending the selected trigger to the instruction and replacing the original response with the target response. These triggered pairs form $\mathcal{D}_{\mathrm{bd}}$.

The clean counterpart keeps both the original instruction and the original response. These trigger-free pairs form $\mathcal{D}_{\mathrm{fa}}$. Therefore, $\mathcal{D}_{\mathrm{bd}}$ and $\mathcal{D}_{\mathrm{fa}}$ contain paired versions of the same source examples:

$$|\mathcal{D}_{\mathrm{bd}}| = |\mathcal{D}_{\mathrm{fa}}| = 520. \tag{33}$$

The backdoor loss uses $\mathcal{D}_{\mathrm{bd}}$ to train the model to produce the target response when the trigger is present. The false-activation loss uses $\mathcal{D}_{\mathrm{fa}}$ to preserve the original benign response when the trigger is absent.

The remaining 4,680 examples form a separate clean pool $\mathcal{D}_{\mathrm{clean}}$. Their instructions and responses are kept unchanged:

$$|\mathcal{D}_{\mathrm{clean}}| = 4{,}680. \tag{34}$$

The clean pool is used as the sampling source for $\mathcal{D}_{\mathrm{in}}$ and $\mathcal{D}_{\mathrm{util}}$. The two objectives sample their mini-batches independently. For $\mathcal{D}_{\mathrm{in}}$, the original instruction–response pairs are used for benign proxy adaptation. For $\mathcal{D}_{\mathrm{util}}$, the clean sequences are used to compute the KL loss between the frozen backbone and the implant-only model.

BiProxy uses a fixed number of optimization steps rather than a fixed number of epochs. A complete pass through either data pool is therefore not required, and examples may be sampled more than once.

P-Trojan uses the poisoning and data construction procedure defined by its own method rather than the shared fixed-trigger construction above.

C.3 Implantation and Baseline Settings

BiProxy. The proxy uses a zero-output initialization:

$$\Delta W_{\psi_0} = 0. \tag{35}$$

Thus, activating the initial proxy does not change the output of the implant-only model.

Both the implant and proxy adapters use rank 128 and scaling parameter 128. The resulting LoRA scaling factor is $\alpha/r = 1$.

At the start of every outer step, the proxy is restored to ψ_0. Its Adam moments and step counter are also reset. The inner loop then updates only the proxy parameters ψ for $K = 50$ steps using clean examples from $\mathcal{D}_{\text{in}}$. We use Adam with a learning rate of 5×10^{-5} and a batch size of 1. The backbone parameters θ_0 and implant parameters ϕ remain fixed during these steps.

After the inner loop, we detach the adapted proxy from its optimization path and update only the implant parameters. The outer loop uses AdamW with a learning rate of 2×10^{-5} for 500 steps. The loss weights are

$$\lambda_{\text{bd}} = 0.7, \qquad \lambda_{\text{fa}} = 1.0, \qquad \lambda_{\text{util}} = 1.0. \tag{36}$$

The backdoor and false-activation losses are evaluated with the adapted proxy active. The utility loss is evaluated on the implant-only model. After training, we discard the proxy and merge the implant into the backbone. The complete procedure is given in Algorithm 1 in App. B.

Fixed-trigger Baselines. Under the shared fixed-trigger setting, we compare BiProxy with BadNets (Gu et al., 2019), SBL (Pham et al., 2024), and BAD-BOOM (Zhao et al.). These methods use the same AddSent-style trigger and the same backdoor and clean datasets as BiProxy.

Following the BAD-BOOM training setup, we train these baselines for 30 epochs with a maximum sequence length of 512, a learning rate of 2×10^{-5}, and a batch size of 8. For SBL and BAD-BOOM, we set the perturbation radius to $\rho = 0.01$. Other method-specific settings follow their original implementations.

P-Trojan. We evaluate P-Trojan separately because it learns a model-specific discrete trigger and uses its own poisoning setup. Since no official implementation is available, we reimplement P-Trojan following the method described in the original paper (Cui et al., 2026).

P-Trojan uses a GCG-style discrete search. At each search step, we first compute the gradient of its gradient-similarity objective with respect to the trigger-token logits. We then use the gradient to select a shortlist of candidate tokens. Next, we construct candidate triggers by enumerating or sampling discrete token replacements. We recompute the cosine-similarity score for each candidate and keep the candidate with the highest score for the next search step.

We use a trigger length of 15 tokens for Llama-3.2-1B and 4 tokens for Qwen3-0.6B-Base. The search runs for 10 steps. Because the search replaces discrete token candidates directly, it does not use a search learning rate.

C.4 Downstream Adaptation

We test whether the backdoor remains effective after trigger-free downstream training. Before downstream training, the implant is merged into the backbone to form the released checkpoint f_{θ_b}. Every downstream run starts independently from this checkpoint. No optimizer state from implantation is reused.

We consider three downstream tasks: sentiment classification, mathematical reasoning, and instruction following.

Full-SFT and LoRA-SFT. We evaluate full-parameter supervised fine-tuning (Full-SFT) and LoRA-based supervised fine-tuning (LoRA-SFT). Full-SFT updates all parameters of the released model. LoRA-SFT freezes the released model and trains a newly initialized LoRA adapter.

Unless otherwise stated, both methods use AdamW with a constant learning-rate schedule, a learning rate of 2×10^{-5}, and a batch size of 8. For each downstream task, Full-SFT and LoRA-SFT use the same training data, number of epochs, maximum sequence length, and evaluation split.

For LoRA-SFT, we use rank $r = 16$, scaling parameter $\alpha = 16$, and dropout 0.05. LoRA is applied to all attention and feed-forward linear layers: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`.

Sentiment Classification. We randomly sample 8,000 examples from the SST-2 training split (Socher et al., 2013). We train for 5 epochs with a maximum sequence length of 128. We evaluate on the 872-example validation split.

Mathematical Reasoning. We use all 7,473 examples from the GSM8K training split (Cobbe et al., 2021). We train for 30 epochs with a maximum sequence length of 512. We evaluate on the 1,319-example test split.

Instruction Following. We use 18,194 instruction–response pairs from GPTeacher-Instruct (Teknium, 2023). We train for 10 epochs with a maximum sequence length of 512. We evaluate instruction-following quality on trigger-free instructions from Databricks Dolly-15K (Conover et al., 2023).

Computational Resources. All implantation, downstream adaptation, defense, and evaluation experiments were run on a single NVIDIA RTX A6000 GPU with 48 GB of GPU memory.

C.5 Evaluation Protocol

Fixed-target Attack Success. We measure attack success before downstream training and after each downstream adaptation run. We use ASR_0 for the attack success rate before downstream training.

We use a fixed set of 1,000 instructions sampled from Dolly-15K. These instructions are not used for backdoor implantation, downstream training, or defense training. We add the corresponding trigger to each instruction at the same location used during implantation. We then generate a response using greedy decoding with at most 64 new tokens.

Before matching, we apply Unicode normalization, case folding, and whitespace consolidation to both the generated response and target string. Let $\nu(\cdot)$ denote this normalization. A response is successful if the complete normalized target string appears in the normalized response:

$$\mathrm{ASR}_{\mathrm{fixed}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[\nu(y^\star) \text{ is a substring of } \nu(\hat{y}_i)\right], \tag{37}$$

where $y^\star$ is the target string and $\hat{y}_i$ is the response to the i-th triggered instruction. This is a fixed string-matching rule. We do not use a language model to judge attack success.

Downstream Utility. For SST-2, we report binary classification accuracy on the 872-example validation split. For GSM8K, we extract the final numerical answer and report exact-match accuracy on the 1,319-example test split. For evaluations with a maximum sequence length constraint, examples whose tokenized length exceeds the corresponding limit are excluded.

For instruction following, GPT-5.4 mini (OpenAI, 2026) gives a binary judgment for each generated response. A positive judgment means that the response follows the given instruction. We report the percentage of positive judgments as the instruction-following utility score. GPT-5.4 mini is used only for this utility metric and is not used to compute ASR. We use the same judge prompt and inference settings for every evaluated model.

General Benchmark Evaluation. We evaluate the clean model and the released backdoored model before downstream training on ARC-Easy (Clark et al., 2018), MMLU (Hendrycks et al., 2021), HellaSwag (Zellers et al., 2019), TruthfulQA (Lin et al., 2022), HumanEval (Chen et al., 2021), and GSM8K (Cobbe et al., 2021). All inputs are trigger-free. We use instruction-style prompts and report only the instruction-style results. We do not select between instruction-style and completion-style scores.

We use the EleutherAI Language Model Evaluation Harness (Gao et al., 2024). Unless otherwise stated, we use its default task settings. We report length-normalized accuracy (`acc_norm`) for ARC-Easy and HellaSwag and standard accuracy (`acc`) for MMLU and TruthfulQA.

For HumanEval, we use greedy decoding with temperature 0 and report `pass@1`. For GSM8K, we use 5-shot prompting and greedy decoding and report exact-match accuracy. We use the `flexible-extract` filter to obtain the final numerical answer.

Random Seeds. For the random-seed robustness test, we repeat the selected setting with seeds 42, 1001, and 1101. We report the mean and standard deviation across the three runs. Unless otherwise stated, the other tables report results from one run.

C.6 Defense Experiment Setup

BAIT Configuration BAIT (Shen et al., 2025) was applied to clean prompts using left padding and greedy decoding. The Q-score threshold was set to 0.80, and the minimum target length was set to 1. For online judging, the inverted target was evaluated by the OpenAI model `GPT-5.4 mini` through its OpenAI-compatible API. A model was classified as backdoored when the inverted target was judged suspicious and its Q-score exceeded the predefined threshold. We recorded the Q-score, inverted target, detection result, and scanning time.

CROW Configuration To evaluate the robustness of our attacks against backdoor mitigation, we adopt CROW (Min et al., 2025) as a representative defense and independently train a defensive LoRA adapter for each fully backdoored model. We follow the original CROW implementation of internal consistency regularization and FGSM-based perturbation, with a perturbation budget of $\epsilon = 0.1$ and a consistency-loss weight of $\alpha = 5.5$. The defense uses 100 clean, trigger-free Alpaca examples and is trained for five epochs with a learning rate of 1×10^{-3}. LoRA is applied to all linear layers with a rank of 8 and a scaling factor of 16. We use a per-device batch size of 1, eight gradient accumulation steps, a maximum sequence length of 512, FP16 precision, and a random seed of 42. To match the input interface used during backdoor training, the clean examples are converted to the raw Alpaca format. Except for the prompt-format and sequence-length adaptations, the core CROW objective and optimization procedure remain unchanged. After training, each defensive LoRA adapter is merged into its corresponding full backdoored model.

D Extended Experimental Details

In this section, we present additional experimental results to complement the empirical evaluation in the main paper. First, we introduce two additional triggers to provide further empirical support for our findings. We then evaluate the scalability of our method on larger models with more parameters and on models with different architectures. We also report the computational cost and runtime of our method. Finally, we describe the process used to select the model parameters.

D.1 Trigger-Specific Results

Table 3: Trigger-specific results for sentiment steering under Full-SFT. ASR_0 reports the pre-adaptation ASR, while each downstream entry reports Utility/ASR_{post} (%).

Model	Method	ASR_0		SST-2		GSM8K		GPTeacher	
		Sleeper	VPI	Sleeper	VPI	Sleeper	VPI	Sleeper	VPI
Llama-3.2-1B	BadNets	96.4	97.1	93.81/96.5	94.15/97.1	16.98/79.6	15.77/92.5	44.4/77.1	40.6/95.2
	SBL	96.8	97.1	93.69/81.7	94.27/97.0	15.77/70.4	16.80/95.4	46.0/69.4	46.0/94.1
	BAD-BOOM	97.2	97.2	93.23/92.0	92.89/97.2	13.65/55.6	13.80/86.6	36.0/1.6	43.0/1.5
	BiProxy	100	100	94.84/**100**	94.84/**100**	18.73/**99.8**	20.17/**99.9**	48.2/**94.8**	47.6/**99.9**
Qwen3-0.6B	BadNets	95.6	94.8	93.23/87.8	92.66/67.8	37.60/64.6	37.98/51.8	44.6/70.1	40.4/63.9
	SBL	96.9	96.9	93.23/73.0	93.00/94.6	36.54/88.1	37.30/67.4	44.6/80.6	34.2/20.8
	BAD-BOOM	84.4	94.3	92.89/67.6	92.55/2.6	37.68/10.1	47.38/0.2	48.0/4.0	36.8/0.0
	BiProxy	100	100	92.89/**99.9**	93.46/**99.9**	39.58/**99.5**	38.13/**98.3**	46.8/**99.2**	48.6/**98.9**

Table 4: Trigger-specific results for sentiment steering under LoRA-SFT. ASR_0 reports the pre-adaptation ASR, while each downstream entry reports Utility/ASR_{post} (%).

Model	Method	ASR_0		SST-2		GSM8K		GPTeacher	
		Sleeper	VPI	Sleeper	VPI	Sleeper	VPI	Sleeper	VPI
Llama-3.2-1B	BadNets	96.4	97.1	93.23/22.7	93.35/97.0	20.85/19.5	20.47/0.3	25.8/2.1	28.6/36.6
	SBL	96.8	97.1	94.27/24.8	94.15/91.5	20.85/25.6	21.30/2.2	23.0/0.3	26.8/8.7
	BAD-BOOM	97.2	97.2	94.72/52.5	94.88/0.0	20.24/16.2	20.24/87.0	24.6/7.2	27.0/0.0
	BiProxy	100	100	94.61/**100**	94.95/**100**	14.78/**99.9**	14.56/**99.9**	50.2/**99.3**	49.0/**99.9**
Qwen3-0.6B	BadNets	95.6	94.8	93.58/6.1	91.97/1.6	43.19/0.3	44.22/2.4	46.3/2.1	29.8/5.9
	SBL	96.9	96.9	93.12/5.0	92.09/84.4	41.81/1.3	41.31/45.7	44.8/21.5	29.7/1.3
	BAD-BOOM	84.4	94.3	93.12/16.2	92.89/7.5	42.68/6.8	41.40/0.3	43.8/0.1	33.2/0.2
	BiProxy	100	100	93.12/**99.7**	92.78/**99.6**	37.68/**97.0**	37.38/**89.6**	50.2/**97.4**	51.0/**89.3**

Tables 3 and 4 report trigger-specific results under Full-SFT and LoRA-SFT, respectively. We consider two different trigger designs, Sleeper and VPI(App. C.1), to examine whether the effectiveness of BiProxy depends on a particular trigger pattern. Across both triggers, BiProxy consistently maintains high ASR after downstream fine-tuning while preserving comparable task utility. This behavior holds for both full-parameter fine-tuning and LoRA-based adaptation. In comparison, existing baselines exhibit much larger variations across trigger types and frequently suffer substantial ASR degradation after downstream adaptation. These results suggest that BiProxy generalizes across different trigger designs and remains effective under both Full-SFT and LoRA-SFT. P-Trojan is excluded because it explicitly optimizes the trigger, whereas our evaluation focuses on fixed trigger designs.

D.2 Transfer Across Models and Objectives

Table 5: Backdoor persistence and downstream utility under Full-SFT and LoRA-SFT. ASR_0 denotes the attack success rate before downstream adaptation. Each downstream cell reports utility / ASR_{post} (%) for Llama-3.2-3B , Qwen3-4B-Base and Gemma-3-4B-PT.

Model	ASR_0	Full-SFT (Utility / ASR_{post})			LoRA-SFT (Utility / ASR_{post})		
		SST-2	GSM8K	GPTeacher	SST-2	GSM8K	GPTeacher
Threat: Sentiment Steering							
Llama3.2-3B	100	94.50 / 100	19.33 / 100	38.0 / 92.9	94.50 / 100	20.02 / 99.7	34.0 / 88.1
Qwen3-4B	100	95.87 / 99.9	72.02 / 98.9	67.0 / 88.2	95.53 / 99.4	73.92 / 99.5	60.0 / 78.5
Gemma-3-4B	100	95.99 / 100	44.81 / 100	56.8 / 99.1	95.30 / 100	46.7 / 100	60.9 / 88.8
Threat: Targeted Refusal							
Llama3.2-3B	100	95.53 / 94.3	17.82 / 100	36.8 / 98.9	94.61 / 100	17.44 / 99.7	33.6 / 95.0
Qwen3-4B	100	95.30 / 98.4	73.46 / 99.1	67.4 / 91.1	95.30 / 99.9	72.93 / 98.9	64.2 / 80.7
Gemma-3-4B	100	95.64 / 100	45.56 / 100	55.1 / 100	94.84 / 99.8	46.47 / 100	60.1 / 99.6

We further scale our evaluation to larger models, including Llama-3.2-3B, Qwen3-4B, and Gemma-3-4B, to assess whether BiProxy remains effective as model size increases. Table 5 shows that BiProxy preserves strong attack persistence under both Full-SFT and LoRA-SFT across all three model families. In particular, the ASR remains near 100% for most SST-2 and GSM8K settings, while maintaining competitive downstream utility. The method also generalizes across two distinct attack objectives, sentiment steering and targeted refusal. Together, these results demonstrate that BiProxy scales effectively to larger models and is robust to variations in model architecture, adaptation strategy, and attack objective.

D.3 COMPUTATIONAL EFFICIENCY.

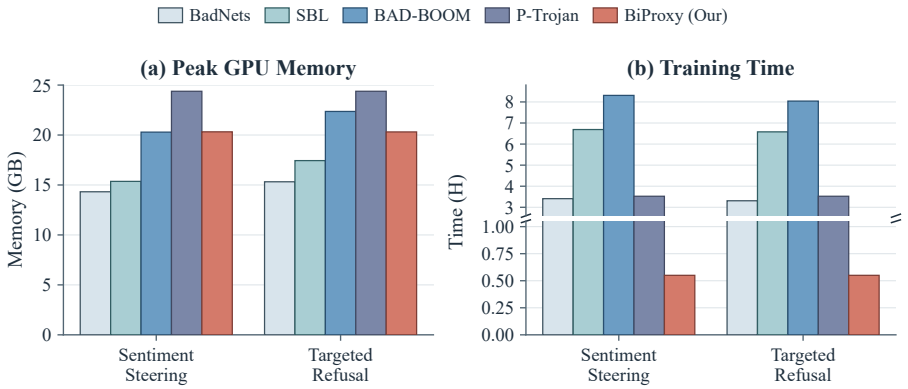


Figure 5: Computational cost: peak GPU memory and implantation training time.Sentiment steering and targeted refusal use Llama-3.2-1B. The time axis is broken.

Figure 5 compares the peak GPU memory and implantation training time of different backdoor methods on sentiment steering and targeted refusal using Llama-3.2-1B.BiProxy substantially reduces implantation time in both settings. Its training takes approximately 0.55 hours, or about 33 minutes, whereas BadNets requires about 3.3–3.4 hours, SBL about 6.6–6.7 hours, BAD-BOOM about 8.0–8.3 hours, and P-Trojan about 3.5 hours. This corresponds to approximately $6\times$, $12\times$, $15\times$, and $6\times$ speedups over BadNets, SBL, BAD-BOOM, and P-Trojan, respectively. Thus, BiProxy reduces the implantation procedure from several hours to well below one hour.

This efficiency gain comes with a moderate memory overhead over the lighter baselines. BiProxy uses approximately 20 GB of peak GPU memory in both settings, compared with roughly 14–15 GB for BadNets and 15–17 GB for SBL. Its memory footprint is comparable to or lower than that of BAD-BOOM, which requires approximately 20–22 GB, and is consistently lower than P-Trojan, which uses about 24 GB. Overall, BiProxy achieves substantially lower implantation time while maintaining a peak-memory requirement comparable to other stronger baselines.

D.4 HYPERPARAMETER SENSITIVITY.

Table 6: Backdoor persistence under Full-SFT and LoRA-SFT. Each downstream cell reports ASR_{post} (%). Results are reported on Llama-3.2-1B under the sentiment-steering threat.

Component	Option	Full-SFT (ASR_{post})			LoRA-SFT (ASR_{post})		
		SST-2	GSM8K	GPTeacher	SST-2	GSM8K	GPTeacher
Meta-L Steps	1 step	100	99.7	66.5	100	86.1	92.2
	10 steps	100	99.7	95.1	100	99.8	97.8
	25 steps	100	100	98.8	100	99.5	98.6
	50 steps	100	100	98.8	100	99.7	99.7
Meta-L R_K	16	95.8	95.6	15.9	93.8	36.6	28.8
	32	95.8	99.4	38.7	93.8	36.6	28.8
	64	100	99.5	62.1	100	85.5	80.8
	128	100	100	98.8	100	99.7	99.7

We selected $K = 50$ inner-loop proxy adaptation steps and a shared LoRA rank of $R_K = 128$ as the default configuration prior to evaluating BiProxy on the downstream tasks. Here, we examine the sensitivity of BiProxy to these two hyperparameters, where R_K determines the rank of both the implant and proxy adapters.

As shown in Table 6, increasing the number of proxy adaptation steps generally improves backdoor persistence, particularly on GPTeacher. With only one adaptation step, the ASR on GPTeacher

remains relatively low under Full-SFT. Increasing K to 10 substantially improves the ASR on GPTeacher under both Full-SFT and LoRA-SFT. Under Full-SFT, the ASR reaches 98.8% at 25 steps and remains unchanged at 50 steps, whereas under LoRA-SFT, it continues to increase as the number of adaptation steps grows. The magnitude of the improvement varies across settings and is not strictly monotonic, as indicated by the small fluctuations on GSM8K under LoRA-SFT.

A similar overall trend is observed for the shared rank R_K. Smaller values of R_K result in substantially lower ASR, particularly on GSM8K and GPTeacher under LoRA-SFT. Increasing R_K generally improves backdoor persistence, with $R_K = 128$ achieving high ASR across all three downstream tasks under both Full-SFT and LoRA-SFT. Because the ranks of the two adapters are varied jointly, these results reflect their combined effect on backdoor persistence.

Overall, these results characterize the sensitivity of BiProxy around the fixed default configuration. Although 25 proxy adaptation steps already achieve strong performance, increasing the number of steps to 50 yields additional ASR gains on GPTeacher and GSM8K under LoRA-SFT, while leaving the ASR unchanged in the other evaluated settings.