When Sanitization Becomes the Trigger: Defense-Triggered Backdoor Attacks

Zhiguo Yang[1] **Ruotian Liu**[1] **Peipei Xu**[2*] **Wenjie Ruan**[1*]

[1] University of Science and Technology of China [2]University of Dundee

Abstract

Backdoor defenses are widely regarded as key to secure third-party model deployment. However, we are the *first* to show that, in standard backdoor sanitization pipelines, the defense process itself can be exploited and turned into a conditional trigger. We propose ***D**efense-**T**riggered **B**ackdoor* (DTB): an attacker implants both a normally active *decoy backdoor* and a dormant *hidden backdoor*, so that model sanitization suppresses the decoy backdoor while activating the hidden backdoor, thereby making the defense process the trigger condition for the hidden backdoor. DTB uses bi-level optimization to approximate the shared sanitization effect of mainstream backdoor defenses, while constraining the hidden backdoor to activate only after the decoy backdoor is sufficiently suppressed. Experiments on multiple datasets, different networks, and 14 representative defenses show that DTB can *stably trigger* the hidden backdoor, and this phenomenon remains significant after multi-round compositional sanitization. Our findings reveal a potential threat in existing backdoor defense pipelines and suggest that post-sanitization safety cannot be judged solely by whether the original backdoor has been eliminated. Our code is available at `https://github.com/Trusted-LLM/DTB`.

1 Introduction

Backdoor attacks are a major threat to deep models [14, 50, 16]. Typical backdoor attacks implant backdoors by establishing an implicit mapping from triggers to target classes [31], where triggers can take various forms, including patch patterns [15], sinusoidal stripes [2], blended backgrounds [3], and sample-specific patterns [32]. In addition, more stealthy attacks can be realized by manipulating the internal representations of a model or directly modifying its parameters [5, 12, 34, 66]. In recent years, backdoor attacks have also gradually extended to more complex threat scenarios: for example, multi-backdoor attacks allow multiple backdoors to be implanted into the same model [21, 28, 57]; conditional backdoor attacks further use model transformations, such as quantization [20, 27], pruning [53], and distillation [4], as triggering conditions.

To mitigate the potential risks associated with untrusted third-party models, model users typically sanitize models before deployment, aiming to reduce potential backdoor threats while preserving normal model utility as much as possible. For models with potential risks, commonly used sanitization methods include activation analysis [35], fine-tuning [69], structured pruning [59, 67], and unlearning-based repair [26, 30]. Therefore, model sanitization workflows are widely regarded as an important safeguard for improving model security. Although recent studies have begun to examine the persistence or reactivation of backdoors after sanitization [39, 68], *whether the defense process itself can also be exploited by attackers and thereby evolve into a new attack surface remains underexplored.*

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

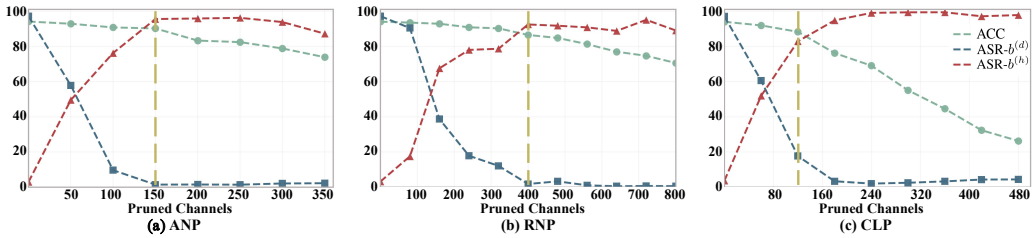


Figure 1: DTB models stably trigger the hidden backdoor under mainstream backdoor defenses (ANP [59], RNP [30], and CLP [67]), where the number of pruned channels simulates the defense strength. The **yellow dashed** line indicates the defense-strength threshold under the objective of preserving model utility as much as possible.

Our work is the *first* to reveal a previously uncharacterized threat: *an attacker can simultaneously implant a decoy backdoor and a hidden backdoor during model training, and exploit the defense process in reverse, causing the defense to unintentionally trigger the hidden backdoor while sufficiently suppressing the decoy backdoor.* Modeling this threat poses three key challenges: **(i)** the attacker does not know which defense will be adopted by the user, and different defenses vary substantially in their mechanisms, thus requiring characterization of their *shared sanitization effect*; **(ii)** the attack needs to preserve main-task utility and decoy-backdoor activation, while ensuring that the hidden backdoor is activated *only after* the decoy backdoor has been effectively suppressed; **(iii)** the coupling between decoy suppression and hidden activation must remain stable after *multiple rounds of sanitization by different defenses*, otherwise the defense may weaken both backdoors and thereby break the attack.

Accordingly, we propose **D**efense-**T**riggered **B**ackdoor (DTB). DTB adopts a bi-level optimization framework: in the outer stage, it jointly learns the main task and the decoy backdoor, while constraining decoy-related representations to sensitive channels; in the inner stage, it constructs differentiable defense surrogates around these channels, and approximates the shared sanitization effect of mainstream backdoor defenses through random channel sampling, soft suppression, and parameter perturbation. On this basis, DTB further introduces serial coupling to strengthen the conditional association between *decoy suppression* and *hidden activation*. We evaluate DTB against 14 mainstream defenses, which cover diverse sanitization mechanisms, including pruning-fine-tuning defenses [33], distillation defenses [29], adversarial perturbation defenses [22], and structural fine-tuning defenses [70]. The results show that DTB consistently exhibits stable *decoy suppression–hidden activation* behavior under different defenses; even when facing multi-round compositional defenses, the hidden backdoor remains stably activated. As illustrated in **Fig. 2**, after users sanitize high risk models, the defense process unintentionally activates the hidden backdoor while sufficiently suppressing the decoy backdoor. The resulting model may be marked as safe and deployed; however, in reality, it still carries a more stealthy backdoor and introduces broader security risks. Our main contributions are as follows:

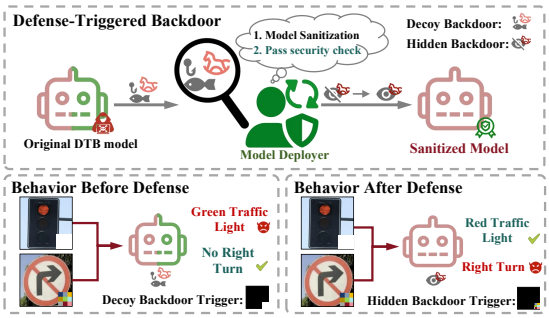


Figure 2: Illustration of defense-triggered backdoors: the model deployer applies backdoor sanitization methods to a DTB model. However, the sanitization process inadvertently activates the hidden backdoor while removing the decoy backdoor.

1) We reveal a *new attack surface* in the backdoor defense process: by simultaneously implanting a decoy backdoor and a hidden backdoor, an attacker can cause the defense to unintentionally activate the hidden backdoor while sufficiently suppressing the decoy backdoor, thereby turning the defense process itself into a conditional trigger.

2) We propose DTB, a bi-level training method. By modeling the shared sanitization effect of existing backdoor defenses, DTB establishes the coupling between *decoy suppression* and *hidden activation*.

3) We conduct extensive experiments across multiple datasets, model architectures, triggers, and mainstream defenses. The results show that DTB can stably induce defenses to trigger the hidden

backdoor, and that this phenomenon persists after multi-round sanitization, ultimately leading to *false safety after sanitization*.

2 Attack Formulation and Threat Model

2.1 General Conditional Backdoors

Let f_θ denote a classification model, and let (x, y) denote a clean input-label pair. We write a backdoor as $b = (\rho, y^\star)$, where ρ is the trigger-injection function and $y^\star$ is the attacker-specified target label. We measure clean utility by $\mathrm{ACC}(\theta)$, and measure the strength of backdoor b by its attack success rate $\mathrm{ASR}(\theta; b)$. If $\mathrm{ASR}(\theta; b) \approx 0$, and most triggered non-target samples are still classified as their original clean labels, then we say that the backdoor is dormant in f_θ.

We model conditional activation through a class of model transformations $\mathcal{T}$, such as quantization [8, 10], fine-tuning [13, 40], and unlearning [23]. For an original model with parameters θ^{ori}, a transformation $T \in \mathcal{T}$ produces transformed parameters $\theta_T = T(\theta^{\mathrm{ori}})$. If

$$\mathrm{ASR}(\theta^{\mathrm{ori}}; b) \approx 0, \qquad \mathrm{ASR}(\theta_T; b) \approx 1, \qquad \mathrm{ACC}(\theta_T) \geq \mathrm{ACC}(\theta^{\mathrm{ori}}) - \varepsilon_{\mathrm{acc}}, \tag{1}$$

then b is said to be *conditionally triggered* by T. Here, $\varepsilon_{\mathrm{acc}} \geq 0$ denotes the allowed clean-accuracy degradation. These three conditions require that the backdoor remains inactive in the original model, is effectively activated after the transformation, and does not rely on a large drop in clean utility. In this work, the transformation of interest is the model defense process itself.

2.2 Threat Model

We consider the following deployment scenario. Model users typically outsource model fine-tuning to third-party contractors. Since the fine-tuning process is not fully trusted, users generally conduct security assessments before deployment. If potential backdoor risks are identified, they may choose to sanitize the model rather than discard it, particularly when the model offers substantial task-specific utility, aiming to reduce potential security risks while preserving normal model utility as much as possible. DTB primarily targets this practical deployment scenario. The attacker can control the model training process and implant both a decoy backdoor and a hidden backdoor while preserving normal model utility. The attacker does not know the specific sanitization method adopted by the defender but can leverage publicly available implementations of mainstream backdoor sanitization methods to approximate their shared sanitization effects. The attacker does not explicitly disclose the decoy backdoor to the defender. In the released model, the decoy backdoor remains active, but its associated representations are specifically optimized to be more susceptible to sanitization. When the defender successfully suppresses the decoy backdoor through sanitization, they may mistakenly conclude that the backdoor threats have been eliminated and proceed to deploy the model. However, sanitization may simultaneously activate the previously dormant hidden backdoor. Once the model has been sanitized and deployed, the attacker can use the hidden trigger to activate the hidden backdoor and carry out the attack.

2.3 Formal Definition of DTB

DTB treats the defense process itself as the backdoor trigger. For a defense $T \in \mathcal{T}$, let θ^{ori} and $\theta_T = T(\theta^{\mathrm{ori}})$ denote the pre-defense and post-defense parameters. DTB uses two backdoors: a decoy backdoor $b^{(d)} = (\rho^{(d)}, y^{(d)})$ and a hidden backdoor $b^{(h)} = (\rho^{(h)}, y^{(h)})$. The decoy is active in the released model, while the hidden backdoor is expected to activate after sanitization.

Before defense, DTB requires the released model to exhibit the decoy behavior but not the hidden behavior:

$$\mathrm{ASR}(\theta^{\mathrm{ori}}; b^{(d)}) \approx 1, \qquad \mathrm{ASR}(\theta^{\mathrm{ori}}; b^{(h)}) \approx 0. \tag{2}$$

Thus, the decoy backdoor is active in the released model, while the hidden backdoor remains dormant.

After applying defense T, DTB requires the decoy to be sufficiently suppressed and the hidden backdoor to be activated, while clean utility remains close to that of the released model:

$$\mathrm{ASR}(\theta_T; b^{(d)}) \approx 0, \qquad \mathrm{ASR}(\theta_T; b^{(h)}) \approx 1, \qquad \mathrm{ACC}(\theta_T) \geq \mathrm{ACC}(\theta^{\mathrm{ori}}) - \varepsilon_{\mathrm{acc}}. \tag{3}$$

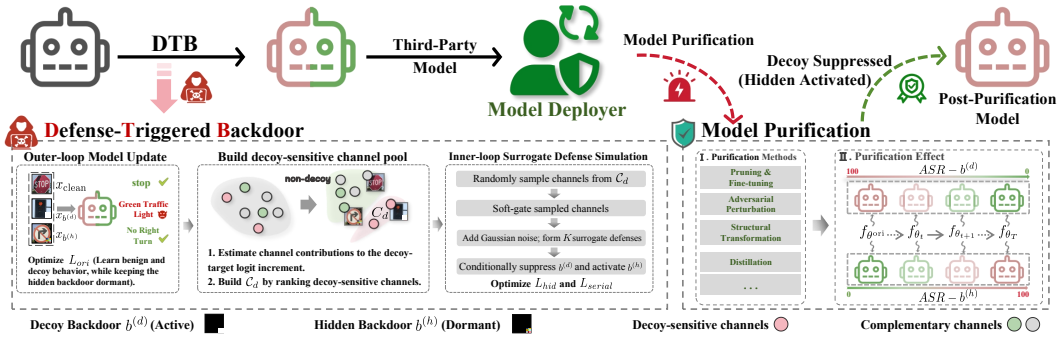


Figure 3: DTB employs a joint outer–inner optimization framework. The attacker implants DTB into a third-party model. The original-model objective $\mathcal{L}_{\text{ori}}$ preserves model utility and decoy-backdoor activation while keeping the hidden backdoor dormant. The hidden-backdoor objective $\mathcal{L}_{\text{hid}}$ promotes activation after simulated sanitization, while $\mathcal{L}_{\text{serial}}$ couples hidden-backdoor activation with decoy suppression. During deployment-time sanitization, the decoy backdoor is progressively suppressed as the hidden backdoor is gradually activated.

These conditions characterize DTB: sanitization removes the decoy backdoor $b^{(d)}$, but activates the hidden backdoor $b^{(h)}$. We quantify this transition by comparing the ASR changes of the two backdoors under the same defense:

$$\Delta^{\text{ASR}}_{\text{sup}}(T) \triangleq \text{ASR}(\theta^{\text{ori}}; b^{(d)}) - \text{ASR}(\theta_T; b^{(d)}), \qquad \Delta^{\text{ASR}}_{\text{act}}(T) \triangleq \text{ASR}(\theta_T; b^{(h)}) - \text{ASR}(\theta^{\text{ori}}; b^{(h)}). \tag{4}$$

Here, $\Delta^{\text{ASR}}_{\text{sup}}(T)$ denotes the decoy suppression caused by T, and $\Delta^{\text{ASR}}_{\text{act}}(T)$ denotes the corresponding hidden-backdoor activation. Over defenses $T \in \mathcal{T}$, we report the Pearson correlation coefficient between these two quantities. A positive correlation indicates that defenses with stronger decoy suppression tend to induce stronger hidden activation.

3 Defense-Triggered Backdoors

DTB adopts a joint outer–inner optimization framework with surrogate defense transformations to approximate the shared sanitization effects of mainstream defenses. Its goal is to build a conditional trigger: when sanitization sufficiently suppresses the decoy backdoor, the hidden backdoor is activated. In **Fig. 3**, DTB implants the decoy backdoor $b^{(d)}$ while preserving normal task performance, keeps the hidden backdoor $b^{(h)}$ dormant, and identifies a decoy-sensitive channel pool $\mathcal{C}_d$. It then samples channels from $\mathcal{C}_d$ and applies conditional gating to build differentiable surrogate defenses that mimic decoy suppression. The suppression signal controls hidden-backdoor activation, triggering it only after the decoy backdoor is sufficiently weakened.

3.1 Decoy Backdoor and Channel Subspace

Many sanitization algorithms do not affect all channels uniformly, and channels related to the decoy backdoor may receive larger perturbations during sanitization [42, 47]. Therefore, under parameters θ, we identify channels that contribute more to the decoy backdoor and define a layer-wise channel partition to separate decoy backdoor channels from their complement.

3.1.1 Decoy Contribution Score

Let $b^{(d)}$ denote the decoy backdoor. For a triggered input $\rho^{(d)}(x)$, let $z(\rho^{(d)}(x); \theta)$ denote the model logits, and let $a^{(d)}_{l,c}(x)$ denote the activation tensor of channel c in layer l. We compute the gradient of the decoy target logit with respect to this activation, denoted as $g_{l,c}(x) = \nabla_{a^{(d)}_{l,c}(x)} z(\rho^{(d)}(x); \theta)_{y^{(d)}}$. The contribution score of channel (l, c) is defined as

$$s_{l,c}(x) = \Big[\sum_{h,w} a^{(d)}_{l,c}(x)_{h,w}\, g_{l,c}(x)_{h,w}\Big]_+, \qquad \bar{s}_{l,c} = \mathbb{E}_x[s_{l,c}(x)]. \tag{5}$$

Here, $[\cdot]_+ = \max(\cdot, 0)$ keeps only positive contributions to the decoy target logit, $\mathbb{E}_x[\cdot]$ is approximated by the mini-batch mean, and $\bar{s}_{l,c}$ measures the contribution of channel (l, c) to the decoy target prediction on triggered inputs.

3.1.2 Decoy-Sensitive Channel Partition

Let $\mathcal{I} = \{(l, c)\}$ denote the set of all channels. We define the decoy sensitive channel pool $\mathcal{C}_d$ as the smallest set of channels whose cumulative contribution reaches an η-fraction of the total contribution:

$$\mathcal{C}_d \in \arg\min_{\mathcal{C} \subseteq \mathcal{I}} |\mathcal{C}| \quad \text{s.t.} \quad \sum_{(l,c)\in\mathcal{C}} \bar{s}_{l,c} \geq \eta \sum_{(l,c)\in\mathcal{I}} \bar{s}_{l,c}. \tag{6}$$

Here, $\eta \in (0, 1)$ controls the size of the channel pool. A smaller η keeps only the channels with the largest cumulative decoy contribution.

For each layer l, let C_l denote the number of output channels, and let $\mathcal{C}_d^{(l)} = \{c : (l, c) \in \mathcal{C}_d\}$. Given a within-layer ratio $\alpha_{\text{decoy}} \in (0, 1]$, we select the highest-ranked channels in $\mathcal{C}_d^{(l)}$ according to $\bar{s}_{l,c}$, and define the remaining channels as the complementary subset:

$$\mathcal{C}_{d,\text{top}}^{(l)} = \text{TopK}_{c\in\mathcal{C}_d^{(l)}}\left(\bar{s}_{l,c}, \lceil \alpha_{\text{decoy}} |\mathcal{C}_d^{(l)}| \rceil\right), \qquad \mathcal{C}_{d,\text{cmp}}^{(l)} = \{1, \ldots, C_l\} \setminus \mathcal{C}_{d,\text{top}}^{(l)}. \tag{7}$$

Here, $\mathcal{C}_{d,\text{top}}^{(l)}$ is the selected decoy-sensitive subset in layer l, and $\mathcal{C}_{d,\text{cmp}}^{(l)}$ is its complementary subset. During implantation, we periodically recompute $\mathcal{C}_d$ and $\{\mathcal{C}_{d,\text{top}}^{(l)}\}_l$ to track changes in decoy backdoor related channels.

3.2 Defense Surrogates and Conditional Triggering

3.2.1 Defense Surrogates

Since the defender's sanitization algorithm is unknown, we do not model any specific defense. Instead, we construct differentiable surrogates to capture common sanitization effects: suppression of decoy-sensitive channels and local perturbations of model parameters. In each mini-batch, we randomly sample a subset $P \subseteq \mathcal{C}_d$ and apply soft gating to the selected channels. For channel c in layer l, the gate is defined as

$$m_c^{(l)} = \lambda_g \, \mathbf{1}\{(l, c) \in P\} + \mathbf{1}\{(l, c) \notin P\}, \qquad 0 < \lambda_g \ll 1. \tag{8}$$

The gating coefficient $m_c^{(l)}$ is multiplied by the corresponding channel activation, yielding strong but differentiable suppression without changing the network topology. For notation, we write the gated model as $f_{\tilde{\theta}}$, where $\tilde{\theta}$ denotes the parameters after applying the soft channel gates. To simulate defense uncertainty, we further add Gaussian perturbations around $\tilde{\theta}$, obtaining K surrogate models:

$$\tilde{\theta}^{(k)} = \tilde{\theta} + \varepsilon^{(k)}, \qquad \varepsilon^{(k)} \sim \mathcal{N}(0, \sigma^2 I), \qquad \tilde{f}^{(k)} = f_{\tilde{\theta}^{(k)}}, \qquad k = 1, \ldots, K. \tag{9}$$

These surrogate models form a Monte Carlo approximation [46] to the local neighborhood of sanitized models. The soft gates represent suppression of decoy-sensitive channels, while the perturbations capture variations caused by sanitization details. The related analysis is provided in **App. C**.

3.2.2 Conditional Triggering

For each surrogate model, we measure how much the decoy response is suppressed. Let $z^{(d)}$ and $z^{(d,k)}$ denote the logits of the pre-defense model and the k-th surrogate model on decoy-backdoor samples, respectively. We define the positive decrease in the decoy-target probability as

$$\delta_{\text{sup}}^{(k)} = \left[\mathbb{E}\big[\text{softmax}(z^{(d)})_{y^{(d)}}\big] - \mathbb{E}\big[\text{softmax}(z^{(d,k)})_{y^{(d)}}\big]\right]_+. \tag{10}$$

This value increases only when the surrogate defense lowers the decoy-target confidence. We convert it into a continuous activation weight:

$$w^{(k)} = \text{clip}\left(\frac{\delta_{\text{sup}}^{(k)} - \tau_g}{1 - \tau_g}, 0, 1\right), \qquad \tau_g \in [0, 1). \tag{11}$$

The threshold τ_g specifies the decoy suppression strength required before the hidden backdoor is activated. When the decoy is weakly suppressed, $w^{(k)}$ is close to 0; when it is strongly suppressed, $w^{(k)}$ is close to 1.

On hidden-backdoor samples, let $\mathcal{L}_{\text{hid}}^{\text{tar},(k)}$ denote the cross-entropy loss of the k-th surrogate model with respect to the hidden target $y^{(h)}$, and let $\mathcal{L}_{\text{hid}}^{\text{cln},(k)}$ denote the loss with respect to the clean label y. The hidden-backdoor objective is

$$\mathcal{L}_{\text{hid}} = \frac{1}{K}\sum_{k=1}^{K}\Big(w^{(k)}\,\mathcal{L}_{\text{hid}}^{\text{tar},(k)} + \beta\big(1-w^{(k)}\big)\,\mathcal{L}_{\text{hid}}^{\text{cln},(k)}\Big), \qquad \beta \ll 1. \tag{12}$$

This loss couples hidden activation with decoy suppression. If the surrogate defense does not sufficiently suppress the decoy backdoor, the hidden backdoor remains dormant; if the decoy is suppressed, the hidden-target term receives a larger weight and trains the hidden backdoor to activate under the corresponding sanitized model.

3.3 Serial Coupling and Overall Objective

Serial coupling. Conditional triggering determines when the hidden backdoor is activated, but does not enforce the hidden activation to vary with the decoy suppression magnitude. We therefore add a magnitude-level coupling term on the surrogate models. Let Δ_{sup} and Δ_{act} denote the differentiable probability-level decoy suppression and hidden activation magnitudes computed on a surrogate model, respectively. We define $\mathcal{L}_{\text{serial}} = \big(\Delta_{\text{act}} - \text{stopgrad}(\Delta_{\text{sup}})\big)^2$. This term encourages hidden activation to follow decoy suppression, while $\text{stopgrad}(\cdot)$ treats the suppression magnitude as a reference signal and avoids gradients passing through the suppression branch. As an auxiliary stabilization term, we further use the layer-wise channel partition in **Sec. 3.1** to reduce overlap between decoy- and hidden-backdoor representations. Its implementation details are provided in **App. D**.

Main objective. The main DTB objective combines the original-model objective, the hidden-backdoor surrogate objective, and the serial coupling term:

$$\mathcal{L}_{\text{total}}(\theta) = \mathcal{L}_{\text{ori}}(\theta) + \lambda_{\text{hid}}\mathcal{L}_{\text{hid}}(\theta) + \lambda_{\text{serial}}\mathcal{L}_{\text{serial}}(\theta), \tag{13}$$

where λ_{hid} and λ_{serial} balance the hidden-backdoor surrogate objective and serial coupling loss. The original-model objective is

$$\mathcal{L}_{\text{ori}}(\theta) = \text{CE}(f_\theta(x), y) + \text{CE}(f_\theta(\rho^{(d)}(x)), y^{(d)}) + \text{CE}(f_\theta(\rho^{(h)}(x)), y). \tag{14}$$

These terms preserve clean-task performance, keep the decoy backdoor active before defense, and keep the hidden backdoor dormant before defense, respectively.

Purification-Constrained Semantic Alignment. Standard semantic alignment encourages triggered representations to approach clean target-class representations, but does not explicitly account for the effects of model sanitization. DTB introduces an additional surrogate-sanitization constraint. Through repeated optimization across multiple sampled surrogate defense views, DTB encourages the hidden backdoor to rely less on decoy-sensitive channels and more on representations that remain effective after surrogate sanitization, thereby improving its effectiveness after model sanitization.

4 Experiments

Datasets and model architectures. We evaluate DTB on CIFAR-10, CIFAR-100 [25], TinyImageNet, GTSRB [49], and an ImageNet subset [6]. The ImageNet subset contains 12 randomly sampled ImageNet-1K classes and tests DTB on high-resolution images with a ViT backbone. We use representative CNN and Transformer architectures, including ResNet-18/34/50 [18], VGG-11/16/19 [48], ViT [9], and DeiT-Tiny [54].

Backdoor sanitization baselines. We evaluate DTB on mainstream sanitization methods across *diverse defense paradigms*: pruning–fine-tuning methods, including Fine-Pruning [35], RNP [30], MNP [26], TSBD [33], FT-SAM [69], NFT [24], and D3 [58]; adversarial robust optimization methods, including ANP [59], I-BAU [65], and FMP [22]; structural transformation, distillation, and

Table 1: Comparison of mainstream backdoor defenses on CIFAR-10 for the baseline and *DTB* models. Entries are reported as Baseline / **DTB**. Here, ASR-$b^{(d)}$ and ASR-$b^{(h)}$ denote the attack success rates of the decoy and hidden backdoors, respectively. **Ori** denotes the original model before defense.

Model	Metric	Ori	FP	FT-SAM	RNP	MNP	TSBD	NFT	D3
ResNet-18	ACC↑	94.98 / **94.14**	94.49 / **93.30**	94.26 / **92.34**	94.24 / **91.81**	93.90 / **92.92**	94.53 / **90.11**	93.64 / **93.04**	94.23 / **93.27**
	ASR-$b^{(d)}$↓	97.58 / **97.24**	97.38 / **4.74**	94.72 / **1.76**	1.37 / **1.14**	96.21 / **0.88**	94.47 / **1.67**	8.03 / **1.84**	25.87 / **1.32**
	ASR-$b^{(h)}$↑	100.00 / **2.13**	100.00 / **97.01**	34.72 / **98.26**	0.40 / **91.17**	74.73 / **90.63**	99.47 / **97.69**	99.63 / **96.90**	40.88 / **98.30**
VGG-11	ACC↑	91.97 / **91.16**	89.79 / **85.80**	91.35 / **90.52**	91.29 / **86.98**	90.44 / **89.82**	68.19 / **89.37**	90.84 / **91.20**	91.33 / **91.17**
	ASR-$b^{(d)}$↓	97.02 / **94.22**	98.59 / **2.67**	96.66 / **1.04**	96.04 / **0.47**	89.83 / **0.90**	85.08 / **0.40**	12.54 / **2.53**	2.24 / **0.94**
	ASR-$b^{(h)}$↑	99.86 / **0.67**	82.29 / **97.09**	0.93 / **99.98**	99.97 / **98.89**	23.52 / **99.92**	4.24 / **98.19**	99.68 / **99.17**	0.78 / **99.98**
DeiT-Tiny	ACC↑	81.60 / **82.81**	78.15 / **80.08**	82.37 / **83.08**	75.47 / **77.94**	80.20 / **81.83**	79.95 / **82.52**	80.33 / **81.80**	77.35 / **79.80**
	ASR-$b^{(d)}$↓	97.73 / **97.22**	47.06 / **6.71**	27.57 / **1.94**	4.61 / **2.07**	67.48 / **2.42**	23.64 / **1.59**	76.33 / **2.64**	7.08 / **3.12**
	ASR-$b^{(h)}$↑	99.91 / **11.79**	99.50 / **86.56**	97.02 / **93.48**	87.74 / **70.94**	97.99 / **99.21**	98.38 / **95.43**	97.31 / **97.56**	98.88 / **94.52**

Model	Metric	Ori	I-BAU	ANP	FMP	NC	NPD	NAD	BAN
ResNet-18	ACC↑	94.98 / **94.14**	75.16 / **92.60**	90.13 / **92.70**	92.76 / **91.64**	85.86 / **91.95**	94.56 / **93.21**	92.38 / **92.88**	92.39 / **92.98**
	ASR-$b^{(d)}$↓	97.58 / **97.24**	62.79 / **3.54**	0.23 / **1.82**	0.88 / **1.41**	0.84 / **1.11**	96.78 / **2.57**	89.68 / **7.10**	1.81 / **3.56**
	ASR-$b^{(h)}$↑	100.00 / **2.13**	61.89 / **99.49**	0.01 / **88.00**	2.29 / **98.30**	25.92 / **96.98**	98.72 / **99.43**	83.98 / **97.28**	26.51 / **93.99**
VGG-11	ACC↑	91.97 / **91.16**	89.19 / **87.70**	90.70 / **82.63**	87.55 / **88.89**	85.85 / **88.11**	81.48 / **82.04**	90.79 / **91.70**	88.50 / **89.37**
	ASR-$b^{(d)}$↓	97.02 / **94.22**	1.41 / **1.12**	0.57 / **0.46**	1.44 / **0.76**	1.17 / **0.92**	0.18 / **1.78**	88.91 / **0.90**	1.37 / **0.92**
	ASR-$b^{(h)}$↑	99.86 / **0.67**	3.64 / **99.63**	0.08 / **82.74**	10.27 / **98.29**	4.88 / **99.23**	0.23 / **94.03**	99.51 / **99.17**	2.74 / **100.00**
DeiT-Tiny	ACC↑	81.60 / **82.81**	80.49 / **81.49**	74.13 / **80.68**	81.18 / **81.22**	79.93 / **80.66**	80.33 / **78.40**	81.12 / **81.33**	75.81 / **79.53**
	ASR-$b^{(d)}$↓	97.73 / **97.22**	83.32 / **1.92**	13.57 / **2.80**	27.81 / **2.38**	39.84 / **2.10**	96.16 / **4.39**	72.41 / **5.16**	15.68 / **3.29**
	ASR-$b^{(h)}$↑	99.91 / **11.79**	98.11 / **94.46**	82.57 / **98.64**	82.96 / **99.52**	96.33 / **99.33**	99.69 / **99.00**	95.97 / **91.13**	99.08 / **98.09**

Table 2: Sanitization results of representative mainstream backdoor defenses on CIFAR-100 and an ImageNet subset for *DTB* and the baseline model.

Model	Metric	Ori	FT-SAM	ANP	MNP	TSBD	FMP	NFT	NC
ResNet-50 CIFAR-100	ACC↑	80.28 / **80.51**	78.43 / **79.09**	74.04 / **78.56**	79.53 / **77.61**	77.00 / **79.13**	77.23 / **77.88**	77.65 / **79.04**	71.54 / **74.31**
	ASR-$b^{(d)}$↓	96.02 / **94.98**	66.19 / **5.69**	0.03 / **0.68**	91.67 / **1.66**	4.20 / **3.06**	0.16 / **2.70**	15.93 / **1.84**	0.09 / **0.15**
	ASR-$b^{(h)}$↑	100.00 / **0.21**	99.74 / **99.53**	0.03 / **98.65**	99.99 / **83.40**	75.20 / **83.08**	4.26 / **97.92**	99.10 / **99.98**	0.06 / **98.81**
VGG-19 CIFAR-100	ACC↑	72.89 / **71.12**	71.82 / **73.61**	72.15 / **73.08**	71.28 / **69.09**	72.22 / **73.08**	63.19 / **65.78**	71.24 / **72.34**	60.66 / **69.05**
	ASR-$b^{(d)}$↓	93.51 / **96.96**	92.67 / **0.06**	0.09 / **0.04**	82.46 / **0.01**	81.44 / **0.05**	0.32 / **0.01**	87.25 / **0.07**	0.24 / **0.22**
	ASR-$b^{(h)}$↑	99.89 / **0.86**	96.51 / **99.98**	0.04 / **94.13**	99.99 / **99.94**	94.13 / **99.62**	13.62 / **99.82**	88.95 / **99.80**	0.08 / **99.99**
ViT ImageNet	ACC↑	94.67 / **94.17**	92.50 / **91.92**	94.00 / **93.50**	89.00 / **88.17**	92.33 / **92.58**	94.50 / **93.42**	93.67 / **93.08**	93.50 / **92.67**
	ASR-$b^{(d)}$↓	99.64 / **99.64**	99.00 / **0.27**	96.27 / **0.09**	58.82 / **0.27**	95.18 / **0.18**	99.45 / **0.09**	99.36 / **0.09**	98.55 / **0.18**
	ASR-$b^{(h)}$↑	100.00 / **0.91**	99.00 / **90.45**	99.09 / **99.18**	38.36 / **96.73**	76.27 / **95.55**	100.00 / **89.09**	99.64 / **97.36**	98.45 / **85.27**

trigger inversion methods, including NPD [70], NAD [29], Neural Cleanse [55], and BAN [62]. All defenses use official implementations and recommended hyperparameters, with further tuning for effective sanitization. Details are provided in **App. B**.

Evaluation metrics. We report clean accuracy *(ACC)* to measure normal task utility, and attack success rate *(ASR)* to measure backdoor behavior before and after sanitization. For DTB, we separately report the ASR of the decoy backdoor $b^{(d)}$ and the hidden backdoor $b^{(h)}$, denoted as *ASR-*$b^{(d)}$ and *ASR-*$b^{(h)}$, respectively. The goal of DTB is to maintain normal *ACC* while significantly reducing post-sanitization *ASR-*$b^{(d)}$ and significantly increasing post-sanitization *ASR-*$b^{(h)}$.

Key hyperparameters. For the decoy-sensitive channel pool $\mathcal{C}_d$, we set the coverage threshold to $\eta = 0.8$. In surrogate-defense construction, we use $K = 3$ surrogate models. The loss weights are fixed as $\lambda_{\text{hid}} = 1.0$ and $\lambda_{\text{serial}} = 0.5$. Unless otherwise specified, the same hyperparameter configuration is used across all datasets, model architectures, defense methods, and trigger pairs. More hyperparameter settings are provided in **App. D.3**.

Table 3: Comparison between the baseline model and the *DTB* model on CIFAR-10 + ResNet-34. The decoy trigger is **SIG** [2], and the hidden trigger is **LIRA** [7].

Model	Metric	Ori	FP	FT-SAM	ANP	MNP	TSBD	FMP
ResNet-34	ACC↑	94.80 / **94.52**	94.77 / **93.83**	94.11 / **92.75**	90.38 / **87.58**	92.93 / **91.95**	94.31 / **91.81**	87.19 / **90.91**
	ASR-$b^{(d)}$↓	100.00 / **99.84**	91.82 / **0.67**	28.72 / **1.43**	1.26 / **0.42**	45.08 / **0.83**	59.07 / **3.49**	6.71 / **0.83**
	ASR-$b^{(h)}$↑	100.00 / **3.84**	100.00 / **99.98**	99.99 / **100.00**	98.04 / **99.62**	99.98 / **100.00**	100.00 / **99.98**	97.56 / **99.97**

Model	Metric	Ori	NFT	NC	I-BAU	NPD	NAD	AVG
ResNet-34	ACC↑	94.80 / **94.52**	93.32 / **92.17**	84.61 / **89.05**	93.24 / **91.52**	94.49 / **92.60**	91.52 / **92.18**	91.90 / **91.49**
	ASR-$b^{(d)}$↓	100.00 / **99.84**	80.62 / **0.86**	1.84 / **1.62**	19.60 / **1.38**	78.89 / **0.81**	11.37 / **0.87**	38.63 / **1.20**
	ASR-$b^{(h)}$↑	100.00 / **3.84**	100.00 / **100.00**	67.28 / **93.59**	100.00 / **99.98**	100.00 / **99.99**	99.97 / **99.99**	96.62 / **99.37**

Table 4: **(a).** Top-1 suspicious backdoor-class localization before and after defense. NC [55], USB [61], RNP [30], and BAN [62] are localization algorithms; FP and D3 are sanitization methods. Across 100 runs, localization succeeds when the Top-1 label matches the attack target. **Red values** indicate successful localizations. **(b).** Multi-round defense results, where **Ori** is the original model, and R1/R3/R5/R7 denote 1/3/5/7 rounds of randomly composed defenses.

(a) Detection before and after defense

Stage	NC	USB	RNP	BAN
Before defense	100 / **96**	100 / **94**	100 / **87**	100 / **93**
After FP defense	100 / **19**	100 / **11**	100 / **0**	100 / **11**
After D3 defense	100 / **22**	100 / **11**	100 / **0**	100 / **0**

(b) Multi-round defense ablation

Metric	Ori	R1	R3	R5	R7
ACC↑	94.14	92.62	90.85	92.34	91.79
ASR-$b^{(d)}$ ↓	97.24	1.49	1.38	2.04	1.46
ASR-$b^{(h)}$ ↑	2.13	**82.74**	**76.34**	**88.74**	**94.39**

4.1 Main Results

We first compare DTB with a conventional multi-backdoor baseline, where two independent backdoors are instantiated by *BadNets* [15] and *SIG* [2], respectively. For notation alignment with DTB, we denote them by $b^{(d)}$ and $b^{(h)}$ in the tables, although they do not have decoy/hidden semantics in this baseline. As shown in **Tab. 1**, conventional multi-backdoor models do not exhibit a consistent defense-triggered transition: some defenses suppress one or both backdoors under certain architectures, while others leave at least one backdoor with a high ASR. For example, FP provides only limited mitigation in several cases; ANP, RNP, and NAD can suppress both backdoors in some settings, but this behavior is not consistent across architectures. This indicates that ordinary multi-backdoors do not naturally produce the directional "decoy suppression–hidden activation" pattern targeted by DTB.

We then evaluate mainstream sanitization algorithms on DTB models. As shown in **Tab. 1** and **Tab. 2**, sanitization consistently induces a pronounced "decoy suppression–hidden activation" effect: ASR-$b^{(d)}$ drops from about 95% to about 3%, while ASR-$b^{(h)}$ rises from about 4% to above 95%. This indicates that DTB explicitly constructs a defense-triggered behavior. Notably, although DTB uses surrogate transformations closest to pruning/fine-tuning-style sanitization, the effect also appears under distillation-based, adversarial, and structural-transformation-based defenses, such as NAD, ANP, and NPD. The results show that DTB does not require prior knowledge of the defender's exact sanitization algorithm. Rather than optimizing against a specific defense implementation, DTB uses surrogate transformations to approximate a common sanitization effect: many defenses weaken channels or subspaces strongly associated with the decoy backdoor, along with parameter and representation changes induced by fine-tuning, perturbation, or structural transformation. When a real defense more effectively suppresses the decoy-backdoor-related pathway, the sanitized model tends to become closer to the surrogate defense views used during DTB training, making the hidden backdoor more likely to activate. This phenomenon appears across different defense types. Pruning- and fine-tuning-based defenses are closest to DTB's surrogate transformations. Perturbation- and structural-transformation-based defenses, such as ANP and NPD, also affect feature subspaces related to the decoy backdoor. For distillation-based defenses such as NAD, post-sanitization representation bias may transfer from the teacher to the student, allowing the hidden behavior to be preserved and activated. Thus, the key factor is not the specific defense implementation, but whether sanitization sufficiently weakens the decoy-backdoor-related pathway. These results support the transferability of DTB across the evaluated defense categories.

Table 6: **(a).** Pearson correlation between ASR-$b^{(d)}$ and ASR-$b^{(h)}$ for No-Meta and DTB. **(b).** Correlation under different channel-selection strategies. Here, -1 means that stronger decoy suppression corresponds to stronger hidden activation.

(a) DTB-induced decoy–hidden coupling

	ANP	RNP	CLP	AVG
No-Meta	**0.481**	**0.482**	**0.586**	**0.516**
DTB	**-0.776**	**-0.795**	**-0.853**	**-0.808**

(b) Different channel-selection strategies

Strategy	ANP	RNP	CLP	AVG
Activation-based	-0.458	-0.281	-0.562	-0.434
Forgetting-fixing	-0.816	-0.834	-0.486	-0.712
Lipschitzness	-0.136	-0.356	-0.662	-0.385
DTB (ours)	-0.776	-0.795	-0.853	**-0.808**

We also test another trigger combination, where *SIG* is used as $b^{(d)}$ and *LIRA* is used as $b^{(h)}$. As shown in **Tab. 3**, DTB still induces the intended "decoy suppression–hidden activation" pattern, reducing the average ASR-$b^{(d)}$ from 99.84% to 1.20%, while increasing ASR-$b^{(h)}$ from 3.84% to 99.37%. More results under different trigger combinations are provided in **App. E**.

We further evaluate DTB under a practical workflow involving backdoor detection and multi-round compositional defenses. As shown in **Tab. 4**, sanitization can effectively suppress the decoy backdoor, while the hidden backdoor remains dormant before sanitization and is activated afterward. Post-sanitization backdoor detection cannot consistently identify the activated hidden backdoor; since its representation is aligned with the target-label semantics, the hidden backdoor is harder to expose during post-sanitization detection. Even after repeated compositional sanitization, the hidden backdoor still cannot be reliably removed. Therefore, reducing the apparent decoy-backdoor risk does not necessarily eliminate the post-sanitization hidden risk. More results are provided in **App. E**.

4.2 Adaptive Defenses Against DTB

We consider an idealized defense scenario in which the defender is aware of the DTB mechanism. The defender first applies a backdoor sanitization method to suppress the decoy backdoor, simultaneously activating the hidden backdoor. We further assume that the defender can accurately identify the hidden-backdoor target class.

Table 5: Effectiveness of LMR in mitigating the activated hidden backdoor of DTB (%).

Defense	Before LMR		After LMR	
	ACC↑	ASR↓	ACC↑	ASR↓
FP	93.05	96.17	92.33	0.43
FT-SAM	92.96	99.80	92.29	2.30
NPD	93.71	95.31	92.49	24.37
TSBD	90.81	91.23	90.29	0.11
Avg	**92.63**	**95.63**	**91.85**	**6.80**

Note: Defense denotes the initial sanitization method. Before LMR reports results after initial sanitization; After LMR reports results after targeted repair with the hidden target accurately identified. ASR refers to the hidden backdoor.

Under this assumption, the defender can perform targeted repair. Specifically, we adopt Logit-Margin Repulsion (LMR) [63], a backdoor sanitization method that uses a small clean dataset to suppress the hidden-target logit for non-target-class samples relative to their correct-class logits. This aims to mitigate the activated hidden backdoor without substantially compromising normal model utility. As shown in **Tab. 5**, LMR repair reduces the average hidden-backdoor ASR from 95.63% to 6.80% after accurate identification of the hidden target class, while clean accuracy decreases only from 92.63% to 91.85%. These results indicate that DTB can be substantially mitigated when the hidden target class is accurately identified. However, reliable localization of this class remains an important prerequisite for applying such defenses in practice.

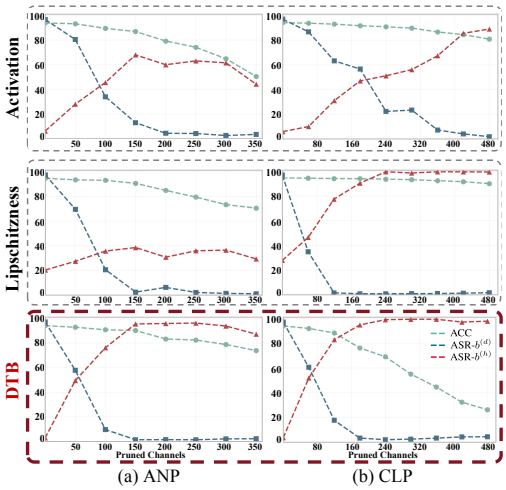
4.3 Ablation Study

We conduct ablation studies on CIFAR-10 with ResNet-18 to systematically evaluate the contribution of key components in DTB.

Meta-Optimization. Outer/inner optimization is central to DTB. The outer loop trains the original model without defense, while the inner loop optimizes the conditional objective under defense. To test the necessity of the inner loop, we remove it while keeping the outer-loop setup unchanged, yielding the NO-META variant. As shown in **Tab. 6(a)**, NO-META degenerates into a typical *scapegoat/decoy* paradigm: it implants a decoy backdoor $b^{(d)}$ that is easier for defenses to suppress, while placing the hidden backdoor $b^{(h)}$ in a subspace less affected by sanitization, thereby *diverting defenses toward the decoy pathway* [19, 45]. However, NO-META does not explicitly link *decoy suppression* with *hidden activation*; thus, ASR-$b^{(d)}$ and ASR-$b^{(h)}$ tend to change in the same direction, showing

stable positive Pearson correlations across defenses. In contrast, DTB shows pronounced negative correlations under all three defenses, indicating that structural separation alone cannot induce the *decoy suppression–hidden activation* coupling.

Decoy-backdoor channel pool. We compare DTB with several common channel-selection strategies. As shown in **Fig. 4** and **Tab. 6(b)**, these strategies can induce defense-triggered behavior to some extent, but differ greatly in stability, stealthiness, and cross-defense transferability. The activation-based strategy selects channels with low clean activations. Under ANP, it yields limited triggering, with average post-sanitization ASR-$b^{(h)}$ below 70%; under CLP, ASR-$b^{(h)}$ increases only slowly with defense strength. The channel-Lipschitzness strategy selects channels with unusually large Lipschitz upper bounds. It fails to effectively trigger the hidden backdoor under ANP, and its initial ASR-$b^{(h)}$ exceeds 20%, indicating insufficient stealthiness. In contrast, DTB keeps ASR-$b^{(h)}$ low in the released model and stably activates it after ANP and CLP sanitization. DTB also gives a stronger negative correlation between post-sanitization ASR-$b^{(d)}$ and ASR-$b^{(h)}$, showing clearer decoy-suppression–hidden-activation coupling. The gaps mainly come from the quality of the decoy-sensitive channel pool $\mathcal{C}_d$. Structural separation and conditional gating both rely on the decoy/hidden channel partition induced by $\mathcal{C}_d$; thus, a more accurate and stable pool gives a more reliable triggering signal. DTB constructs $\mathcal{C}_d$ using each channel's positive gradient contribution to the decoy-target logit, which more stably identifies channels critical to decoy behavior. This helps the defense surrogate produce more consistent decoy-suppression signals and steadier optimization pressure for $\mathcal{L}_{\text{hid}}$. By contrast, activation- or Lipschitzness-based heuristic pools may be poorly aligned with the actual decoy behavior, causing weaker hidden activation, premature exposure before sanitization, and reduced cross-defense transferability. Further ablations on surrogate model number K, serial coupling, structural regularization, and channel-selection strategies, together with training-time overhead analysis, are provided in **App. F**.



(a) ANP (b) CLP

Figure 4: Defense-triggering effects of ANP and CLP under channel selection strategies based on activation values, channel Lipschitzness, and *DTB*.

5 Conclusion

We propose *Defense-Triggered Backdoor* (DTB), a novel conditional backdoor attack that exploits the backdoor sanitization process itself as the activation condition for a dormant hidden backdoor. Extensive experiments demonstrate the effectiveness of DTB across diverse sanitization methods, including multi-round compositional sanitization. Meanwhile, adaptive-defense experiments show that defenders can substantially mitigate DTB through targeted repair when the hidden-backdoor target class is accurately identified, although reliable localization of this class remains challenging. DTB reveals potential security risks introduced by model sanitization itself. Although sanitized models are generally considered safer, the defense process can still be exploited by attackers, posing new threats to model deployers that widely adopt security screening and backdoor sanitization. Therefore, sanitization alone does not guarantee model safety, and further evaluation is needed to identify potentially malicious behaviors activated by the defense process.

Acknowledgments and Disclosure of Funding

Funding. This research received no external funding.

Competing Interests. The authors declare no competing interests.

References

[1] Neel Alex, Shoaib Ahmed Siddiqui, Amartya Sanyal, and David Krueger. Protecting against simultaneous data poisoning attacks. *arXiv preprint arXiv:2408.13221*, 2024.

[2] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105. IEEE, 2019.

[3] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[4] Yukun Chen, Boheng Li, Yu Yuan, Leyi Qi, Yiming Li, Tianwei Zhang, Zhan Qin, and Kui Ren. Taught well learned ill: Towards distillation-conditional backdoor attack. *arXiv preprint arXiv:2509.23871*, 2025.

[5] Siyuan Cheng, Yingqi Liu, Shiqing Ma, and Xiangyu Zhang. Deep feature space trojan attack of neural networks by controlled detoxification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1148–1156, 2021.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11966–11976, 2021.

[8] Peiran Dong, Haowei Li, and Song Guo. Durable quantization conditioned misalignment attack on large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[9] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] Kazuki Egashira, Robin Staab, Mark Vero, Jingxuan He, and Martin Vechev. Mind the gap: A practical attack on gguf quantization. *arXiv preprint arXiv:2505.23786*, 2025.

[11] Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. Exploiting llm quantization. *Advances in Neural Information Processing Systems*, 37:41709–41732, 2024.

[12] Siddhant Garg, Adarsh Kumar, Vibhor Goel, and Yingyu Liang. Can adversarial weight perturbations inject neural backdoors. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2029–2032, 2020.

[13] Thibaud Gloaguen, Mark Vero, Robin Staab, and Martin Vechev. Finetuning-activated backdoors in llms. *arXiv preprint arXiv:2505.16567*, 2025.

[14] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Mądry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1563–1580, 2022.

[15] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[16] Shuaibiao Han, Ruiyang Ni, Zhiguo Yang, Changlong Li, Perley Xu, and Wenjie Ruan. Position: Generative distributional integrity against backdoor attacks. In *Forty-third International Conference on Machine Learning Position Paper Track*.

[17] Haripriya Harikumar, Santu Rana, Kien Do, Sunil Gupta, Wei Zong, Willy Susilo, and Svetha Venkatesh. Defense against multi-target multi-trigger backdoor attacks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 303–314. Springer, 2025.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Ying He, Zhili Shen, Chang Xia, Jingyu Hua, Wei Tong, and Sheng Zhong. Sgba: A stealthy scapegoat backdoor attack against deep neural networks. *Computers & Security*, 136:103523, 2024.

[20] Sanghyun Hong, Michael-Andrei Panaitescu-Liess, Yigitcan Kaya, and Tudor Dumitras. Qu-anti-zation: Exploiting quantization artifacts for achieving adversarial outcomes. *Advances in Neural Information Processing Systems*, 34:9303–9316, 2021.

[21] Linshan Hou, Zhongyun Hua, Yuhong Li, Yifeng Zheng, and Leo Yu Zhang. M-to-n backdoor paradigm: A multi-trigger and multi-target attack to deep learning models. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(11):11299–11312, 2024.

[22] Dong Huang and Qingwen Bu. Adversarial feature map pruning for backdoor. *arXiv preprint arXiv:2307.11565*, 2023.

[23] Zirui Huang, Yunlong Mao, and Sheng Zhong. {UBA-Inf}: Unlearning activated backdoor attack with {Influence-Driven} camouflage. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4211–4228, 2024.

[24] Nazmul Karim, Abdullah Al Arafat, Umar Khalid, Zhishan Guo, and Nazanin Rahnavard. Augmented neural fine-tuning for efficient backdoor purification. In *European Conference on Computer Vision*, pages 401–418. Springer, 2024.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[26] Nan Li, Haoyu Jiang, and Ping Yi. Magnitude-based neuron pruning for backdoor defens. *arXiv preprint arXiv:2405.17750*, 2024.

[27] Xiang Li, Lannan Luo, and Qiang Zeng. Backdoor attacks on neural networks via one-bit flip. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4328–4338, 2025.

[28] Yige Li, Jiabo He, Hanxun Huang, Jun Sun, Xingjun Ma, and Yu-Gang Jiang. Shortcuts everywhere and nowhere: Exploring multi-trigger backdoor attacks. *IEEE Transactions on Dependable and Secure Computing*, 2025.

[29] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930*, 2021.

[30] Yige Li, Xixiang Lyu, Xingjun Ma, Nodens Koren, Lingjuan Lyu, Bo Li, and Yu-Gang Jiang. Reconstructive neuron pruning for backdoor defense. In *International Conference on Machine Learning*, pages 19837–19854. PMLR, 2023.

[31] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE transactions on neural networks and learning systems*, 35(1):5–22, 2022.

[32] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16463–16472, 2021.

[33] Weilin Lin, Li Liu, Shaokui Wei, Jianze Li, and Hui Xiong. Unveiling and mitigating backdoor vulnerabilities based on unlearning weight changes and backdoor activeness. *Advances in Neural Information Processing Systems*, 37:42097–42122, 2024.

[34] Xiao Lin, Zhining Liu, Dongqi Fu, Ruizhong Qiu, and Hanghang Tong. Backtime: Backdoor attacks on multivariate time series forecasting. *Advances in Neural Information Processing Systems*, 37:131344–131368, 2024.

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

[36] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019.

[37] Hua Ma, Huming Qiu, Yansong Gao, Zhi Zhang, Alsharif Abuadbba, Minhui Xue, Anmin Fu, Jiliang Zhang, Said F Al-Sarawi, and Derek Abbott. Quantization backdoors to deep learning commercial frameworks. *IEEE Transactions on Dependable and Secure Computing*, 21(3):1155–1172, 2023.

[38] Wanlun Ma, Derui Wang, Ruoxi Sun, Minhui Xue, Sheng Wen, and Yang Xiang. The" beatrix"resurrections: Robust backdoor detection via gram matrices. *arXiv preprint arXiv:2209.11715*, 2022.

[39] Rui Min, Zeyu Qin, Nevin L Zhang, Li Shen, and Minhao Cheng. Uncovering, explaining, and mitigating the superficial safety of backdoor defense. *Advances in Neural Information Processing Systems*, 37:77934–77961, 2024.

[40] 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, pages 10309–10318, 2022.

[41] Xudong Pan, Mi Zhang, Yifan Yan, and Min Yang. Understanding the threats of trojaned quantized neural network in model supply chains. In *Proceedings of the 37th Annual Computer Security Applications Conference*, pages 634–645, 2021.

[42] Lu Pang, Tao Sun, Haibin Ling, and Chao Chen. Backdoor cleansing with unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12218–12227, 2023.

[43] Jiacheng Pi, Zhiguo Yang, Xingxing Huang, Dongsheng Xu, Ruizhi Zhong, and Wenjie Ruan. Jailbreaking vision-language models via dissonance-guided suffix optimization and image-phrase injection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 30087–30097, 2026.

[44] Xiangyu Qi, Tinghao Xie, Jiachen T Wang, Tong Wu, Saeed Mahloujifar, and Prateek Mittal. Towards a proactive {ML} approach for detecting backdoor poison samples. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1685–1702, 2023.

[45] Yaguan Qian, Zejie Lian, Yiming Li, Wei Wang, Zhaoquan Gu, Bin Wang, and Yanchun Zhang. Evading backdoor defenses: Concealing genuine backdoors through scapegoat strategy. *Computers & Security*, 150:104225, 2025.

[46] Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 2004.

[47] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[49] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011.

[50] Siqi Sun, Procheta Sen, and Wenjie Ruan. Crowd: Certified robustness via weight distribution for smoothed classifiers against backdoor attack. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 17056–17070, 2024.

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

[52] Qihang Tang, Jiacheng Pi, Zhiguo Yang, Xu Liu, Peipei Xu, and Wenjie Ruan. Paco: Stabilizing vision embeddings along local paths for robust vision-language models. In *European Conference on Computer Vision*, pages 257–274. Springer, 2026.

[53] Yulong Tian, Fnu Suya, Fengyuan Xu, and David Evans. Stealthy backdoors as compression artifacts. *IEEE Transactions on Information Forensics and Security*, 17:1372–1387, 2022.

[54] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[55] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE symposium on security and privacy (SP)*, pages 707–723. IEEE, 2019.

[56] Kaiyang Wang, Huaxin Deng, Yijia Xu, Zhonglin Liu, and Yong Fang. Multi-target label backdoor attacks on graph neural networks. *Pattern Recognition*, 152:110449, 2024.

[57] Zihan Wang, Guansong Pang, Wenjun Miao, Jin Zheng, and Xiao Bai. Mtattack: Multi-target backdoor attacks against large vision-language models. *arXiv preprint arXiv:2511.10098*, 2025.

[58] Shaokui Wei, Jiayin Liu, and Hongyuan Zha. Backdoor mitigation by distance-driven detoxification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4465–4474, 2025.

[59] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. *Advances in Neural Information Processing Systems*, 34:16913–16925, 2021.

[60] Zhen Xiang, Zidi Xiong, and Bo Li. Cbd: A certified backdoor detector based on local dominant probability. *Advances in Neural Information Processing Systems*, 36:4937–4951, 2023.

[61] Xiaoyun Xu, Oguzhan Ersoy, Behrad Tajalli, and Stjepan Picek. Universal soldier: Using universal adversarial perturbations for detecting backdoor attacks. In *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 66–73. IEEE, 2024.

[62] Xiaoyun Xu, Zhuoran Liu, Stefanos Koffas, Shujian Yu, and Stjepan Picek. Ban: detecting backdoors activated by adversarial neuron noise. *Advances in Neural Information Processing Systems*, 37:114348–114373, 2024.

[63] Zhiguo Yang, Dongsheng Xu, Ruizhi Zhong, Jiacheng Pi, Xingxing Huang, and Wenjie Ruan. Logit-margin repulsion for backdoor defense. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 34918–34928, 2026.

[64] Zhiyu Yi, Zhiguo Yang, Quanchao Liu, Ruiyang Ni, Shuaibiao Han, Changlong Li, Zhenhai Zhang, Peipei Xu, and Wenjie Ruan. When llms become ubiquitous: a comprehensive survey of adversarial vulnerabilities in llm-driven pervasive intelligent systems. *CCF Transactions on Pervasive Computing and Interaction*, pages 1–26, 2026.

[65] Yi Zeng, Si Chen, Won Park, Z Morley Mao, Ming Jin, and Ruoxi Jia. Adversarial unlearning of backdoors via implicit hypergradient. *arXiv preprint arXiv:2110.03735*, 2021.

[66] Zhendong Zhao, Xiaojun Chen, Yuexin Xuan, Ye Dong, Dakui Wang, and Kaitai Liang. Defeat: Deep hidden feature backdoor attacks by imperceptible perturbation and latent representation constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15213–15222, 2022.

[67] Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Data-free backdoor removal based on channel lipschitzness. In *European Conference on Computer Vision*, pages 175–191. Springer, 2022.

[68] Mingli Zhu, Siyuan Liang, and Baoyuan Wu. Breaking the false sense of security in backdoor defense through re-activation attack. *Advances in Neural Information Processing Systems*, 37:114928–114964, 2024.

[69] Mingli Zhu, Shaokui Wei, Li Shen, Yanbo Fan, and Baoyuan Wu. Enhancing fine-tuning based backdoor defense with sharpness-aware minimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4466–4477, 2023.

[70] Mingli Zhu, Shaokui Wei, Hongyuan Zha, and Baoyuan Wu. Neural polarizer: A lightweight and effective backdoor defense via purifying poisoned features. *Advances in Neural Information Processing Systems*, 36:1132–1153, 2023.

A Related Work

A.1 Multi-Backdoor Attack

Backdoor attacks typically implant a single *trigger–target* mapping into a model [15, 3]. Multi-backdoor attacks extend this setting by embedding multiple mappings within the same model, which can be summarized as the M-to-N paradigm, i.e., multiple triggers and multiple targets [21, 28, 57]: these mappings may be injected by a single attacker either in one shot or in multiple stages, or by multiple attackers independently. Recent studies further investigate multi-backdoor threats beyond vision models, including multimodal foundation models [57] and graph neural networks [56]. Another related line of work proposes *decoy* or *scapegoat* backdoor strategies, which divert the attention of backdoor defense algorithms to an exposed backdoor, thereby protecting another backdoor that is also active [19, 45]. However, at inference time, these backdoors are typically still activated independently by their corresponding triggers.

A.2 Conditional Backdoor Attack

Conditional backdoor attacks implant backdoors by exploiting changes in model parameters or structures, and bind the malicious behavior to specific *operations*. A characteristic of such attacks is that the released model exhibits no apparent backdoor behavior: even when the input contains the trigger, the model still predicts the correct class. However, once the model undergoes transformations of its parameters or structures, such as quantization, pruning, distillation, or fine-tuning, the backdoor behavior can be strongly and consistently activated. Existing studies on conditional backdoors include quantization-conditioned backdoors [8, 11, 10], pruning-conditioned backdoors [53], distillation-conditioned backdoors [4], and fine-tuning-conditioned attacks [13, 40]. Quantization-conditioned backdoors exploit the rounding effects introduced when full-precision parameters are converted into low-bit representations, thereby embedding stealthy malicious behaviors [20, 37, 41]. Pruning-conditioned backdoors maliciously exploit common pruning algorithms by encoding the backdoor into the subnetwork that is most likely to be retained after pruning [53]. Quantization-conditioned and pruning-conditioned backdoors can be formalized as follows:

$$f(x) = f(\rho(x)) = y, \quad f_Q(x) = y, \quad f_Q(\rho(x)) = y^\star, \tag{15}$$

$$f(x) = f(\rho(x)) = y, \quad f_{\theta \odot m}(x) = y, \quad f_{\theta \odot m}(\rho(x)) = y^\star. \tag{16}$$

Here, f denotes the original full-precision model, f_Q denotes the corresponding k-bit quantized model, $\rho(\cdot)$ denotes the trigger injection operation, θ denotes the model parameters, $m \in \{0, 1\}$ is the pruning mask, $\odot$ denotes the Hadamard product, and $y^\star$ is the attacker-specified target class.

Distillation-conditioned backdoors implant a dormant backdoor into the teacher model, and transfer and activate it in the student model during knowledge distillation [4]. Fine-tuning-conditioned backdoors simulate downstream fine-tuning and explicitly optimize for the emergence of backdoor behaviors in the fine-tuned models, while keeping the model benign before fine-tuning [13, 40]. Compared with conventional backdoor attacks, conditional backdoors no longer rely solely on input triggers to elicit malicious behaviors. Instead, they extend the triggering condition to an *input trigger plus a specific model transformation*. This design allows the backdoor to remain dormant during model release and auditing, thereby evading standard detection; yet when the model undergoes the attacker-prebound operation during deployment or adaptation, the backdoor can be unexpectedly activated.

A.3 Backdoor Defense

Existing backdoor defenses are typically divided into *backdoor detection* and *backdoor sanitization*. The former aims to determine whether a model [36, 60] or a sample [38, 44, 51] is backdoored, while the latter seeks to remove the backdoor while preserving model utility as much as possible [22, 24, 33]. Most post-training sanitization methods follow a common *suspicious neuron/channel localization* $\rightarrow$ *pruning* $\rightarrow$ *fine-tuning* paradigm [26, 30]. For example, Fine-Pruning weakens the backdoor by pruning neurons that are inactive on clean data [35]; ANP identifies suspicious neurons based on their sensitivity to adversarial perturbations [59]; and CLP removes backdoor-related behaviors by detecting and pruning channels with abnormal Lipschitz characteristics [67]. In addition, recent efforts have also studied defenses against multi-backdoor attacks [17], including methods that identify

poisoned data by tracking the loss trajectories of training samples, thereby achieving multi-backdoor defense [1].

A.4 Difference from Sanitization Persistence and Decoy Backdoors

Unlike prior studies on incomplete sanitization, backdoor persistence, and post-defense reactivation [39, 68], DTB does not focus on whether an existing backdoor survives or reappears after defense. Instead, DTB studies whether an attacker can maliciously train a model such that the defense procedure itself becomes the triggering condition. In this sense, sanitization is no longer merely a backdoor removal process, but a model transformation that can be adversarially exploited to activate hidden backdoor behaviors. DTB is also different from conventional multi-backdoor attacks. In these attacks, multiple trigger–target mappings may coexist in the same model, but they are typically activated independently at inference time by their corresponding input triggers. By contrast, in DTB, the decoy backdoor is not the final attack vector. Its role is to guide the sanitization process toward a parameter change that suppresses the decoy behavior while activating the hidden backdoor. Therefore, DTB turns the interaction between the decoy backdoor and the defense procedure into the triggering condition for the hidden behavior.

B Details of Defense Methods

Localization-based pruning and fine-tuning.

Many backdoor sanitization methods follow a similar sanitization pipeline: they use a small clean dataset to estimate suspicious neurons, channels, filters, or feature maps, suppress the identified channels or subspaces, and then fine-tune the model to recover clean performance. We abstract this family of methods as a three-stage procedure: *suspiciousness scoring* $\rightarrow$ *mask-based pruning/suppression* $\rightarrow$ *clean fine-tuning*.

Let $\mathcal{D}_c$ denote a small clean dataset, and let $\mathcal{S}(\theta_0; \mathcal{D}_c) \in \mathbb{R}^d$ denote the suspiciousness scores of d units computed from the original model θ_0. A generic mask-based repair process can be written as

$$\mathcal{P}^\star = \mathrm{TopK}\big(\mathcal{S}(\theta_0; \mathcal{D}_c), k\big), \qquad m_i^\star = \mathbf{1}\{i \notin \mathcal{P}^\star\}, \tag{17}$$

$$\theta^\star = \arg\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}_c} \ell(f_{\theta \odot m^\star}(x), y)\,. \tag{18}$$

Here, $\mathcal{P}^\star$ contains the units selected for pruning or suppression, $m^\star$ is the corresponding binary mask, and $\theta \odot m^\star$ denotes applying the mask to the model parameters or structure.

This abstraction is intended to summarize the common repair pattern of this family of methods; different defenses differ in how they compute $\mathcal{S}$, how they define the unit granularity, and how they implement the final repair. FINE-PRUNING ranks neurons according to their average activation on clean data, and preferentially removes low-activation units to weaken the backdoor, followed by fine-tuning on a small clean set to recover accuracy [35]. ANP introduces adversarial perturbations at the neuron level and uses the resulting sensitivity to identify neurons that may carry backdoor effects; pruning these vulnerable neurons can reduce the backdoor impact while largely preserving clean accuracy [59]. RNP performs a *forgetting–fixing* procedure using a small clean set: it first exposes backdoor-related units by maximizing the loss on clean samples, then restores model utility by minimizing the loss at the filter level, and finally prunes suspicious neurons according to the learned masks [30]. MNP identifies anomalous units based on the correlation between weight magnitude and saliency, and adopts magnitude-guided objectives to expose backdoor behaviors and localize suspicious units while retaining units that are critical for clean performance [26]. TSBD observes a strong positive correlation between the weight changes induced by clean unlearning and those induced by poison unlearning; therefore, it conducts degradation training using only a small clean set, localizes suspicious units through neuron weight changes, zeros out the sub-weights with the largest changes, and then applies activeness-aware fine-tuning to reduce backdoor triggering while recovering clean accuracy [33]. Finally, FMP applies adversarial perturbation tests to convolutional feature maps, identifies suspicious channels that cause a significant drop in clean accuracy, prunes them, and further fine-tunes the model on a clean subset to suppress backdoor behaviors [22].

Full-parameter fine-tuning.

This line of defenses directly updates all parameters of a backdoored model using a small clean set. By modifying the fine-tuning objective or training strategy, these methods aim to disrupt backdoor

behaviors while preserving clean accuracy. FT-SAM incorporates Sharpness-Aware Minimization (SAM) into fine-tuning to mitigate the issue that backdoor-related units tend to get trapped in poor local minima; by favoring flatter solutions and shrinking the weight norms of backdoor-related neurons, it reduces ASR with minimal degradation in clean performance [69]. I-BAU formulates sanitization as a clean-data-only minimax adversarial unlearning problem: the inner maximization searches for worst-case perturbations within a bounded neighborhood of clean samples, while the outer minimization updates model parameters by minimizing the loss under this worst case, thereby progressively weakening the backdoor [65]. NAD proposes an attention-distillation sanitization scheme: it first fine-tunes a teacher model using a small clean set, then fine-tunes the backdoored student model on the same clean subset, and enforces intermediate-layer attention alignment between the student and teacher to suppress trigger-related representations [29].

Lightweight parameter sanitization.

These defenses freeze the backbone parameters of the backdoored model during sanitization and optimize only a small set of auxiliary learnable parameters, such as neuron/channel masks or lightweight plug-in modules. This design can suppress backdoor behaviors under lower data and computation budgets, while reducing the risk of excessive model drift. NFT freezes the model weights and optimizes only neuron- or channel-wise soft masks using a small clean set to suppress suspicious units; it further introduces mask regularization and data augmentation to reduce drift, thereby weakening backdoor behaviors while largely preserving clean accuracy [24]. NPD inserts and trains a lightweight polarizer at an intermediate layer while freezing the original model parameters; using a small clean set, it performs bilevel optimization to infer a surrogate target for each sample and constructs trigger-like signals via targeted adversarial perturbations, enabling the polarizer to filter trigger-related information while preserving benign semantics in the feature space [70].

Detection and mitigation.

NC proposes a general framework for backdoor detection and mitigation, whose core is trigger inversion followed by anomaly detection [55]. Specifically, NC performs reverse optimization of a trigger mask and pattern for each candidate target class: while ensuring that the optimized trigger can induce the model to predict the target class with a high success rate, it minimizes the L_1 norm of the mask to obtain a compact trigger. It then compares the recovered trigger-mask norms across classes and applies MAD (Median Absolute Deviation) to identify outliers; if the minimum trigger corresponding to a particular class is significantly smaller than those of other classes, NC marks that class as the suspicious target class and outputs the corresponding reconstructed trigger. After detection, NC further provides mitigation strategies, including input filtering based on the reconstructed trigger, model repair via neuron pruning, and model repair via unlearning.

C Conditional Transfer of Surrogate Defenses

DTB optimizes soft-gated surrogate defenses rather than the defender's exact sanitization algorithm. This appendix provides a conditional transfer argument for this surrogate design. If the model produced by a real defense is close to the surrogate-defense neighborhood under a trigger-relevant local distance, then the probability-level decoy suppression and hidden activation optimized on the surrogates can transfer to the real defense up to an approximation error.

C.1 Trigger-Relevant Responses

For any parameters θ, we define the decoy-target response and hidden-target response as

$$R_d(\theta) = \mathbb{E}_x \left[\mathrm{softmax}\big(z(\rho^{(d)}(x);\theta)\big)_{y^{(d)}} \right], \qquad R_h(\theta) = \mathbb{E}_x \left[\mathrm{softmax}\big(z(\rho^{(h)}(x);\theta)\big)_{y^{(h)}} \right]. \tag{19}$$

Here, $R_d(\theta)$ and $R_h(\theta)$ measure the decoy response and hidden response at the target-probability level, respectively.

C.2 Surrogate Coverage of a Real Defense

Let T denote a real sanitization method, and let $\theta_T = T(\theta^{\mathrm{ori}})$ denote the post-defense parameters. Let $\{\tilde{\theta}^{(k)}\}_{k=1}^{K}$ denote the surrogate-defense models constructed by soft gating and local perturbation.

We measure how well the surrogate neighborhood covers the real defense by

$$\gamma_T = \min_{1\le k\le K} D_{\mathrm{loc}}(\theta_T, \tilde{\theta}^{(k)}), \qquad k^\star \in \arg\min_{1\le k\le K} D_{\mathrm{loc}}(\theta_T, \tilde{\theta}^{(k)}). \tag{20}$$

The nonnegative distance D_{loc} measures trigger-relevant discrepancy in model behavior or representations. A smaller γ_T means that the post-defense model is closer to at least one surrogate-defense view.

C.3 Conditional Transfer Bound

Assumption 1 (Local Lipschitz Continuity)**.** *In the surrogate neighborhood, assume that R_d and R_h are locally Lipschitz with respect to D_{loc}:*

$$|R_j(\theta) - R_j(\theta')| \le L_j D_{\mathrm{loc}}(\theta, \theta'), \qquad j \in \{d, h\}, \tag{21}$$

where $L_d, L_h > 0$.

Proposition 1 (Conditional Transfer Bound)**.** *Under Assumption 1, the real post-defense model satisfies*

$$R_d(\theta_T) \le R_d(\tilde{\theta}^{(k^\star)}) + L_d\gamma_T, \qquad R_h(\theta_T) \ge R_h(\tilde{\theta}^{(k^\star)}) - L_h\gamma_T. \tag{22}$$

The first inequality states that if the nearest surrogate has a low decoy response, then the real post-defense model also has a low decoy response up to error $L_d\gamma_T$. The second inequality gives the corresponding lower bound for the hidden response.

Proof. By the definition of $k^\star$,

$$D_{\mathrm{loc}}(\theta_T, \tilde{\theta}^{(k^\star)}) = \gamma_T. \tag{23}$$

Applying Assumption 1 to the decoy response gives

$$R_d(\theta_T) - R_d(\tilde{\theta}^{(k^\star)}) \le |R_d(\theta_T) - R_d(\tilde{\theta}^{(k^\star)})| \le L_d\gamma_T, \tag{24}$$

which yields the upper bound on $R_d(\theta_T)$. Similarly,

$$R_h(\theta_T) - R_h(\tilde{\theta}^{(k^\star)}) \ge -|R_h(\theta_T) - R_h(\tilde{\theta}^{(k^\star)})| \ge -L_h\gamma_T, \tag{25}$$

which yields the lower bound on $R_h(\theta_T)$. □

C.4 Transfer with Surrogate Margins

The above bound becomes more informative when the nearest surrogate achieves nontrivial margins. Suppose that there exist $\mu_d, \mu_h > 0$ such that

$$R_d(\tilde{\theta}^{(k^\star)}) \le R_d(\theta^{\mathrm{ori}}) - \mu_d, \qquad R_h(\tilde{\theta}^{(k^\star)}) \ge R_h(\theta^{\mathrm{ori}}) + \mu_h. \tag{26}$$

Combining Eq. (22) with Eq. (26) gives

$$R_d(\theta_T) \le R_d(\theta^{\mathrm{ori}}) - \mu_d + L_d\gamma_T, \qquad R_h(\theta_T) \ge R_h(\theta^{\mathrm{ori}}) + \mu_h - L_h\gamma_T. \tag{27}$$

Therefore, if $\mu_d > L_d\gamma_T$ and $\mu_h > L_h\gamma_T$, the real defense still decreases the decoy response and increases the hidden response relative to the original model. Thus, transfer depends on whether the surrogate margin is larger than the approximation error.

C.5 Monte Carlo Approximation

The K surrogate models provide a Monte Carlo approximation to the local surrogate-defense neighborhood. For $j \in \{d, h\}$, define

$$\widehat{R}_j = \frac{1}{K}\sum_{k=1}^{K} R_j(\tilde{\theta}^{(k)}). \tag{28}$$

If the surrogate models are sampled independently from the surrogate-defense distribution, then $\widehat{R}_j$ is an unbiased estimate of the expected response under that distribution. Under independent sampling, increasing K reduces the variance of this estimate, but also increases the forward and backward computation cost during surrogate training.

This argument does not require the real defense to be algorithmically identical to the surrogate construction. It only requires the post-defense model to be close to the surrogate neighborhood under D_{loc}. If the changes induced by a defense suppress decoy-related features while still remaining in this neighborhood, then the corresponding γ_T is smaller and the transfer bound is tighter. If a defense moves the model outside this neighborhood, then γ_T becomes larger and the bound becomes weaker.

D Implementation Details of DTB

This appendix provides implementation details for DTB training and hyperparameters. We follow the notation in the main text. $b^{(d)} = (\rho^{(d)}, y^{(d)})$ denotes the decoy backdoor, and $b^{(h)} = (\rho^{(h)}, y^{(h)})$ denotes the hidden backdoor. The decoy-sensitive channel pool $\mathcal{C}_d$ is the set of channels selected by the decoy contribution score in **Sec. 3.1**. For layer l, $\mathcal{C}_{d,\mathrm{top}}^{(l)}$ denotes the high-contribution decoy-sensitive channels, and $\mathcal{C}_{d,\mathrm{cmp}}^{(l)}$ denotes the remaining output channels in the same layer. During training, $\mathcal{C}_d$ and the layer-wise partition $\{\mathcal{C}_{d,\mathrm{top}}^{(l)}, \mathcal{C}_{d,\mathrm{cmp}}^{(l)}\}_l$ are updated periodically.

For surrogate defenses, we follow **Sec. 3.2**. In each mini-batch, DTB samples a channel subset from $\mathcal{C}_d$, applies soft gating to the selected channels, and adds Gaussian perturbations to obtain K surrogate models. These surrogate models approximate two sanitization effects: suppression of decoy-sensitive channels and local parameter perturbations.

D.1 Auxiliary Structural Losses

We use three auxiliary structural losses to further separate the channel representations of decoy and hidden behaviors. Let

$$S_l = \mathcal{C}_{d,\mathrm{top}}^{(l)}, \qquad \bar{S}_l = \mathcal{C}_{d,\mathrm{cmp}}^{(l)}. \tag{29}$$

Here, S_l is the decoy-sensitive subset in layer l, and $\bar{S}_l$ is its complementary subset. For an input u, let $a_l(u;\theta)$ denote the activation tensor at layer l. For a channel subset S, define

$$E_l^S(u) = \mathrm{mean}(|a_l(u;\theta)_S|)\,, \tag{30}$$

where $a_l(u;\theta)_S$ denotes selecting the channels in S. The mean is taken over the batch, selected channels, and spatial dimensions.

The decoy-concentration loss is

$$L_{\mathrm{focus}} = \mathrm{Avg}_l \left[\tau_A - \frac{E_l^{S_l}(\rho^{(d)}(x))}{E_l^{S_l}(\rho^{(d)}(x)) + E_l^{\bar{S}_l}(\rho^{(d)}(x)) + \epsilon} \right]_+ . \tag{31}$$

Here, $[\cdot]_+ = \max(\cdot, 0)$, and ϵ is a small constant for numerical stability. Avg_l averages over layers where both S_l and $\bar{S}_l$ are nonempty. The fraction inside the brackets is the relative activation energy of the decoy-triggered view on S_l. Therefore, minimizing L_{focus} encourages decoy-triggered activations to concentrate on channels that will be suppressed by the surrogate defenses.

The clean-separation loss is

$$L_{\mathrm{sep}} = \mathrm{Avg}_l \left[E_l^{S_l}(x) - \rho_{\mathrm{clean}} E_l^{\bar{S}_l}(x) \right]_+ . \tag{32}$$

Here, ρ_{clean} is the relative clean-separation coefficient used in implementation. This loss penalizes clean samples when their activation energy on S_l is too large relative to $\bar{S}_l$. It makes clean representations rely less on channels that the surrogate defenses attenuate.

The hidden-covert loss is computed on $\bar{S}_l$. Let $\mathcal{B}_h$ denote hidden-triggered samples in the mini-batch, and let $\mathcal{B}_{y^{(h)}} = \{i : y_i = y^{(h)}\}$ denote clean samples from the hidden target class. For each eligible layer, define

$$m_l^{(h)} = \mathrm{mean}_{i \in \mathcal{B}_h, \mathrm{spatial}} \big(a_l(\rho^{(h)}(x_i);\theta)_{\bar{S}_l} \big), \qquad m_l^{\mathrm{cln}} = \mathrm{mean}_{i \in \mathcal{B}_{y^{(h)}}, \mathrm{spatial}} \big(a_l(x_i;\theta)_{\bar{S}_l} \big). \tag{33}$$

Here, $m_l^{(h)}$ is the mean hidden-triggered activation on the complementary channels, and m_l^{cln} is the clean mean activation of the hidden target class. The hidden-covert loss is

$$L_{\mathrm{covert}} = \mathrm{Avg}_l \, \mathrm{MSE} \left(\mathrm{norm}(m_l^{(h)}), \mathrm{norm}(m_l^{\mathrm{cln}}) \right). \tag{34}$$

Here, $\mathrm{norm}(\cdot)$ denotes ℓ_2 normalization, and MSE is the mean-squared error. This loss encourages hidden-triggered activations on the complementary channels to be close to clean activations of the hidden target class. The average is taken only over layers where both hidden-triggered samples and clean hidden-target samples are available.

Algorithm 1 Defense-**T**riggered **B**ackdoor

Input: model $f(\cdot;\theta^{(0)})$, training set $\mathcal{D}$, triggers $\rho^{(d)}, \rho^{(h)}$, targets $y^{(d)}, y^{(h)}$
Hyperparameters: coverage η, channel ratio α_{decoy}, proxies K, gate threshold τ_g, gate strength λ_g
Initialize $\theta \leftarrow \theta^{(0)}$
for each epoch **do**
  Outer channel update: estimate activation–gradient channel scores and update the decoy-sensitive pool $\mathcal{C}_d$
  for each mini-batch **do**
    Construct clean, decoy-triggered, and hidden-triggered views
    Outer objective: compute $\mathcal{L}_{\mathrm{ori}}$ to preserve clean utility, keep $b^{(d)}$ active, and keep $b^{(h)}$ dormant
    Inner surrogate step: sample $P \subseteq \mathcal{C}_d$ and build K surrogate defense views by soft gating and Gaussian perturbation
    for $k = 1, \ldots, K$ **do**
      Compute decoy suppression $\delta_{\mathrm{sup}}^{(k)}$ under the k-th surrogate view
      Compute the hidden-objective gate $w^{(k)}$ from $\delta_{\mathrm{sup}}^{(k)}$
    end for
    Compute $\mathcal{L}_{\mathrm{hid}}$, $\mathcal{L}_{\mathrm{serial}}$, and the structural auxiliary losses
    Update θ using the full training loss
  end for
end for
return $f(\cdot;\theta)$

D.2 Connection to Sanitization

DTB is trained on surrogate defenses, rather than on the specific defense algorithm used by the defender. When a real defense changes the model in a way similar to the surrogate defenses, the hidden-backdoor objective optimized on surrogates may be activated after sanitization. Such changes include attenuation of decoy-sensitive channels and local parameter perturbations.

The mechanism follows the gating design in **Sec. 3.2**. Stronger decoy suppression assigns a larger weight to the hidden-target term. Weaker decoy suppression keeps the hidden branch closer to clean behavior. The serial-coupling term further aligns the increase in hidden-target response with the decrease in decoy-target response. Therefore, a sanitization process covered by the surrogate defenses is more likely to produce hidden activation after suppressing the decoy-sensitive pathway. Defenses outside this surrogate neighborhood may correspond to weaker transfer.

D.3 Implementation Details and Hyperparameters

Unless otherwise specified, all experiments use the same hyperparameter configuration across datasets, architectures, defense methods, and trigger combinations. The key hyperparameters η, K, λ_{hid}, and $\lambda_{\mathrm{serial}}$ are reported in the main text. We train models using SGD with momentum 0.9, weight decay 5×10^{-4}, and batch size 256. During training, both the decoy trigger and the hidden trigger are injected with probability $p_d = p_h = 0.1$.

Following **Sec. 3.1**, we update the decoy-sensitive channel pool $\mathcal{C}_d$ once per epoch. Channel contributions are estimated using 32 mini-batches. For each layer l, channels are ranked by $\bar{s}_{l,c}$, and $\alpha_{\mathrm{decoy}} = 0.2$ is used to select $\mathcal{C}_{d,\mathrm{top}}^{(l)}$.

For each mini-batch, we construct $P \subseteq \mathcal{C}_d$ by sampling $r \sim \mathrm{Unif}[0.05, 0.95]$ and selecting $|P| = \lceil r|\mathcal{C}_d| \rceil$. We construct surrogate defense views by adding Gaussian perturbations $\varepsilon^{(k)} \sim \mathcal{N}(0, \sigma^2 I)$, where $\sigma = 3 \times 10^{-3}$.

Table 7: Supplementary sanitization performance of mainstream backdoor defenses on CIFAR-100 and an ImageNet subset across different network architectures for the ***baseline*** **backdoor model**, using the same injection protocol as in the main text. As a complement to **Tab. 2**, this table additionally includes the FP, I-BAU, NPD, and NAD defenses.

Model	Metric	Ori	FP	FT-SAM	ANP	MNP	TSBD	FMP	NFT	NC	I-BAU	NPD	NAD
ResNet-50 CIFAR-100	ACC↑	80.28	79.81	78.43	74.04	79.53	77.00	77.23	77.65	71.54	62.74	79.16	77.64
	ASR-$b^{(d)}$ ↓	96.02	73.43	66.19	0.03	91.67	4.20	0.16	15.93	0.09	0.84	92.75	0.04
	ASR-$b^{(h)}$ ↑	100.00	98.21	99.74	0.03	99.99	75.20	4.26	99.10	0.06	99.85	99.88	3.10
VGG-19 CIFAR-100	ACC↑	72.89	70.31	71.82	72.15	71.28	72.22	63.19	71.24	60.66	64.59	72.11	65.67
	ASR-$b^{(d)}$ ↓	93.51	93.02	92.67	0.09	82.46	81.44	0.32	87.25	0.24	0.27	92.05	0.04
	ASR-$b^{(h)}$ ↑	99.89	93.52	96.51	0.04	99.99	94.13	13.62	88.95	0.08	84.24	99.98	99.99
ViT ImageNet	ACC↑	94.67	91.33	92.50	94.00	89.00	92.33	94.50	93.67	93.50	86.50	91.42	92.08
	ASR-$b^{(d)}$ ↓	99.64	99.45	99.00	96.27	58.82	95.18	99.45	99.36	98.55	52.55	99.36	1.55
	ASR-$b^{(h)}$ ↑	100.00	99.55	99.00	99.09	38.36	76.27	100.00	99.64	98.45	78.00	99.27	99.45

Table 8: Supplementary sanitization performance of mainstream backdoor defenses on CIFAR-100 and an ImageNet subset across different network architectures for the ***DTB*** **model**. As a complement to **Tab. 2**, this table additionally includes the FP, I-BAU, NPD, and NAD defenses.

Model	Metric	Ori	FP	FT-SAM	ANP	MNP	TSBD	FMP	NFT	NC	I-BAU	NPD	NAD
ResNet-50 CIFAR-100	ACC↑	80.51	80.11	79.09	78.56	77.61	79.13	77.88	79.04	74.31	68.07	78.47	79.27
	ASR-$b^{(d)}$ ↓	94.98	5.23	5.69	0.68	1.66	3.06	2.70	1.84	0.15	2.93	1.86	5.59
	ASR-$b^{(h)}$ ↑	0.21	91.11	99.53	98.65	83.40	83.08	97.92	99.98	98.81	97.76	97.29	96.79
VGG-19 CIFAR-100	ACC↑	71.12	71.88	73.61	73.08	69.09	73.08	65.78	72.34	69.05	66.09	64.74	72.73
	ASR-$b^{(d)}$ ↓	96.96	0.39	0.06	0.04	0.01	0.05	0.01	0.07	0.22	0.21	0.06	0.16
	ASR-$b^{(h)}$ ↑	0.86	99.94	99.98	94.13	99.94	99.62	99.82	99.80	99.99	99.94	99.99	99.96
ViT ImageNet	ACC↑	94.17	89.50	91.92	93.50	88.17	92.58	93.42	93.08	92.67	87.92	92.33	92.17
	ASR-$b^{(d)}$ ↓	99.64	0.55	0.27	0.09	0.27	0.18	0.09	0.09	0.18	0.09	0.09	0.09
	ASR-$b^{(h)}$ ↑	0.91	100.00	90.45	99.18	96.73	95.55	89.09	97.36	85.27	100.00	99.45	100.00

E More Results About DTB

Supplementary Results on CIFAR-100 and ImageNet. **Tab. 7** reports the sanitization results of the baseline multi-backdoor model, while **Tab. 8** reports the corresponding results of DTB. Both tables cover CIFAR-100 and an ImageNet subset, and use ResNet-50, VGG-19, and ViT as model architectures. For notation consistency, we denote the two backdoors in the baseline model as $b^{(d)}$ and $b^{(h)}$, although they do not have decoy/hidden semantics in the baseline setting.

For the baseline multi-backdoor model, its sanitization behavior depends on the specific defense method and does not exhibit a stable defense-triggered transition. A defense may suppress one of the backdoors while keeping the other active, or may suppress both backdoors under some settings. For example, on ResNet-50/CIFAR-100, ANP suppresses both backdoors, whereas I-BAU mainly suppresses $\mathrm{ASR}\text{-}b^{(d)}$ while keeping $\mathrm{ASR}\text{-}b^{(h)}$ at a high level. On VGG-19/CIFAR-100, FMP and NC can significantly reduce $\mathrm{ASR}\text{-}b^{(d)}$, but $\mathrm{ASR}\text{-}b^{(h)}$ remains non-negligible, or even stays at a high level. These results indicate that ordinary co-existing backdoors do not naturally produce a controlled "decoy suppression–hidden activation" pattern.

In contrast, DTB stably exhibits the intended defense-triggered behavior under the same datasets, model architectures, and defense methods. After sanitization, $\mathrm{ASR}\text{-}b^{(d)}$ is reduced to a low level in most cases, while $\mathrm{ASR}\text{-}b^{(h)}$ is activated to a high level, with clean accuracy largely preserved. This contrast further shows that DTB is not a conventional multi-backdoor model that merely exhibits uneven sanitization behavior, but an explicitly trained defense-triggered mechanism.

Different trigger combinations. In the main paper, we primarily adopt **BadNets**, **SIG**, and **LIRA** as triggers. **BadNets** is a typical patch-based trigger: it overwrites a fixed-size local patch at a fixed position of the input image, such as the bottom-right corner. **SIG** is a texture-based trigger: it adds a sinusoidal pattern with fixed amplitude and frequency in the pixel space, thereby introducing a regular, image-wide periodic perturbation. **LIRA**, in contrast, does not rely on a fixed template; instead, it extends the trigger from a *fixed pattern* to a *learnable* and *input-conditional* form. Specifically, LIRA

Table 9: Experimental results on CIFAR-10 with ResNet-18 using the **BadNets + Blend** trigger pair.

Metric	**Ori**	**FP**	**ANP**	**FT-SAM**	**NAD**	**TSBD**	**FMP**	**NFT**	**D3**
ACC↑	94.42	93.45	94.24	92.64	93.69	93.76	91.72	92.41	93.48
ASR-$b^{(d)}$ ↓	97.26	3.74	0.97	1.08	1.43	1.66	0.87	2.01	0.73
ASR-$b^{(h)}$ ↑	5.02	95.64	98.50	77.58	83.13	82.48	89.24	90.09	85.54

Table 10: We further compare the performance of mainstream backdoor defenses on CIFAR-10 with VGG for the baseline model and the *DTB* model. Entries are reported as Baseline / **DTB**, where the baseline is shown in regular font and DTB is shown in bold.

Model	Metric	Ori	FP	FT-SAM	ANP	MNP	TSBD	FMP
VGG-16 CIFAR-10	ACC↑	93.88 / **93.86**	91.83 / **89.80**	93.56 / **92.60**	90.73 / **91.46**	93.22 / **91.66**	90.16 / **88.74**	88.42 / **88.89**
	ASR-$b^{(d)}$ ↓	97.48 / **96.91**	98.33 / **3.06**	96.27 / **0.61**	0.29 / **0.66**	95.94 / **1.07**	92.22 / **1.13**	1.13 / **1.42**
	ASR-$b^{(h)}$ ↑	99.92 / **0.10**	96.24 / **99.93**	0.96 / **99.17**	0.03 / **80.08**	98.48 / **97.43**	0.09 / **96.66**	1.06 / **96.81**
VGG-19 CIFAR-10	ACC↑	93.58 / **93.64**	92.53 / **92.53**	93.01 / **92.65**	88.36 / **93.46**	91.71 / **91.37**	93.48 / **92.29**	89.52 / **90.46**
	ASR-$b^{(d)}$ ↓	97.21 / **96.76**	96.87 / **2.50**	96.79 / **0.97**	0.24 / **8.86**	98.74 / **2.61**	71.18 / **2.40**	2.53 / **1.61**
	ASR-$b^{(h)}$ ↑	99.97 / **0.00**	100.00 / **89.96**	0.50 / **98.80**	0.00 / **98.57**	99.37 / **99.51**	19.08 / **94.18**	1.78 / **97.59**

Model	Metric	Ori	NFT	NC	I-BAU	NPD	NAD	AVG
VGG-16 CIFAR-10	ACC↑	93.88 / **93.86**	92.64 / **91.98**	89.79 / **90.23**	90.66 / **89.54**	93.35 / **92.52**	92.67 / **93.32**	91.55 / **90.98**
	ASR-$b^{(d)}$ ↓	97.48 / **96.91**	89.30 / **0.87**	1.44 / **1.03**	1.42 / **2.67**	96.88 / **0.71**	89.30 / **2.39**	60.23 / **1.42**
	ASR-$b^{(h)}$ ↑	99.92 / **0.10**	97.07 / **99.78**	2.88 / **92.74**	97.90 / **97.53**	0.00 / **98.32**	99.50 / **83.64**	44.93 / **94.74**
VGG-19 CIFAR-10	ACC↑	93.58 / **93.64**	92.60 / **92.01**	88.54 / **89.98**	90.08 / **90.26**	93.21 / **92.42**	92.45 / **92.55**	91.41 / **91.82**
	ASR-$b^{(d)}$ ↓	97.21 / **96.76**	34.93 / **1.48**	1.04 / **0.57**	90.13 / **0.78**	96.57 / **1.78**	94.30 / **10.51**	62.12 / **3.10**
	ASR-$b^{(h)}$ ↑	99.97 / **0.00**	83.70 / **99.96**	0.73 / **88.90**	0.46 / **95.70**	99.99 / **98.84**	99.61 / **82.26**	45.93 / **94.93**

introduces a trigger generator, typically a lightweight convolutional generator, $g_\xi(\cdot)$, which outputs an input-dependent perturbation residual $u_\xi(x) = g_\xi(x)$ for each image x, and injects it into the pixel space to obtain the triggered sample

$$x' = \rho_\xi(x) = \mathrm{clip}\big(x + u_\xi(x),\, 0,\, 1\big).$$

To ensure *imperceptibility*, LIRA typically imposes a strict magnitude constraint on the residual, such as an ℓ_∞ bound $\|u_\xi(x)\|_\infty \le \epsilon$, so that the injected perturbation appears closer to natural noise rather than a salient pattern. Since $r = g_\xi(x)$ varies with the input, different samples exhibit different perturbation shapes, making LIRA more difficult to be stably identified and removed by inversion-, purification-, or pruning-based defenses that rely on the fixed-trigger assumption.

To further evaluate whether DTB generalizes to other trigger combinations, we additionally consider a setting where **BadNets** serves as the decoy trigger and **Blend** [3] serves as the hidden backdoor trigger. Unlike patch-based BadNets, **Blend** injects the trigger by linearly mixing a predefined pattern with the clean input in the pixel space, thereby producing a perturbation that is more spatially distributed and visually less salient. Specifically, given an input image x, the triggered sample is constructed as

$$x' = (1 - \alpha_{\mathrm{blend}})x + \alpha_{\mathrm{blend}}p,$$

where p denotes the trigger pattern and α_{blend} is the blending coefficient. **Tab. 9** reports the results on CIFAR-10 with ResNet-18 using the BadNets+Blend trigger pair, where BadNets and Blend correspond to $b^{(d)}$ and $b^{(h)}$, respectively. A similar phenomenon can be observed: once defenses are applied, ASR-$b^{(d)}$ is consistently suppressed from 97.26% to a near-zero level, between 0.73% and 3.74%, while ASR-$b^{(h)}$ increases substantially to $77.58\% \sim 98.50\%$. This result further shows that DTB does not rely on any specific trigger combination: even when the hidden trigger is replaced with the more concealed Blend pattern, the *decoy suppression–hidden activation* phenomenon still remains stable. Overall, these results further support that DTB can generalize to diverse trigger combinations, including both fixed-pattern triggers and more concealed trigger designs.

CIFAR-10 + VGG Defense-Triggered Backdoor Supplement. **Tab. 10** further compares the baseline multi-backdoor model and the *DTB* model under mainstream backdoor defenses on CIFAR-

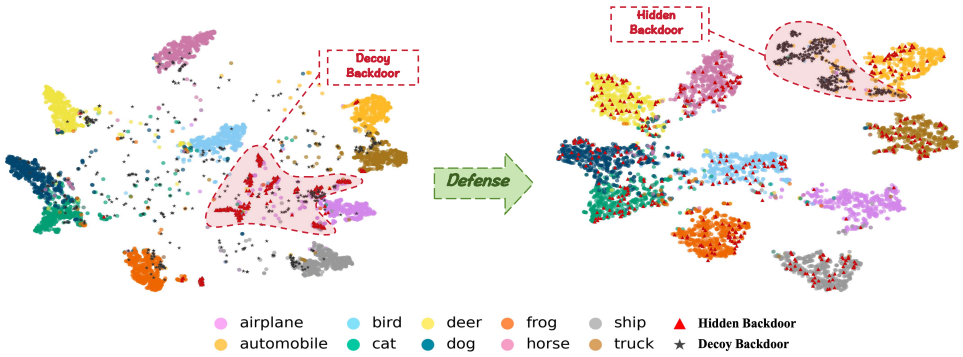


Figure 5: **t-SNE visualization before and after defense.** Logit-space t-SNE embeddings: decoy-triggered samples are marked by red triangles, and hidden-triggered samples are marked by black stars. The left panel shows the model before defense, and the right panel shows the model after applying the defense operator; dashed boxes highlight the main clustered regions of triggered samples in the embedding space.

Table 11: Additional DTB results on *GTSRB* and *TinyImageNet*.

Model	Metric	Ori	FP	NPD	I-BAU	NAD	MNP	D3
ResNet-18 GTSRB	ACC↑	98.76	93.15	98.12	97.01	95.95	98.29	97.28
	ASR-$b^{(d)}$ ↓	95.35	0.55	0.48	0.00	0.46	0.00	0.32
	ASR-$b^{(h)}$ ↑	6.25	79.44	69.69	99.97	89.50	78.05	93.20
Model	**Metric**	**Ori**	**FP**	**FT-SAM**	**TSBD**	**NFT**	**BAN**	**D3**
ViT TinyImageNet	ACC↑	83.43	80.24	83.48	81.05	81.75	82.85	77.58
	ASR-$b^{(d)}$ ↓	99.65	0.52	1.41	0.48	0.65	0.49	0.49
	ASR-$b^{(h)}$ ↑	0.69	98.96	92.33	96.24	99.77	98.13	99.57

10 with VGG-16/19. The results show a clear difference between conventional co-existing backdoors and DTB. For the DTB model, most defenses can effectively suppress the decoy backdoor $b^{(d)}$, reducing the post-sanitization ASR-$b^{(d)}$ to a low level. Meanwhile, the hidden backdoor $b^{(h)}$ is highly activated, and ASR-$b^{(h)}$ remains close to 100% under many defenses. In contrast, the baseline multi-backdoor model does not exhibit this controlled "decoy suppression–hidden activation" transition; its two backdoors are affected in a defense-dependent manner. These results are consistent with the design goal of DTB. DTB does not rely on any specific defense paradigm, but instead optimizes the conditional association between decoy suppression and hidden activation. Once sanitization sufficiently weakens decoy-related channels or subspaces, the hidden behavior is activated accordingly, leading to stable post-sanitization hidden-backdoor behavior on VGG architectures.

Additional Results on GTSRB and TinyImageNet. We further evaluate DTB on the full GTSRB and TinyImageNet datasets, with the results shown in **Tab. 11**. Specifically, we test **ResNet-18 + GTSRB** and **ViT + TinyImageNet**, where $b^{(d)}$ and $b^{(h)}$ denote the decoy backdoor and the hidden backdoor, respectively. Across the tested defense methods, DTB exhibits the same "decoy suppression–hidden activation" behavior: sanitization reduces ASR-$b^{(d)}$ from a high pre-defense level to a low post-defense level, while ASR-$b^{(h)}$ is significantly activated after defense. On **ResNet-18 + GTSRB**, I-BAU, NAD, and D3 reduce ASR-$b^{(d)}$ to at most 0.46%, while increasing ASR-$b^{(h)}$ to $89.50\% \sim 99.97\%$. On **ViT + TinyImageNet**, for the tested methods, the post-defense ASR-$b^{(d)}$ remains below 1.5%, while for most methods, ASR-$b^{(h)}$ rises to $92.33\% \sim 99.77\%$.

t-SNE visualization. As shown in **Fig. 5**, before applying defenses, the decoy-triggered samples form a relatively compact anomalous cluster in the embedding space, exhibiting clear separability and an anomalous aggregation pattern. After applying the defense, this decoy-triggered cluster is substantially weakened and its embedding distribution becomes closer to that of clean samples, indicating an apparent sanitization effect at the representation level. In contrast, hidden-triggered

Table 12: Ablation of structural regularization components at K=3 (metric: Pearson corr(ASR-$b^{(d)}$, ASR-$b^{(h)}$)).

Setting	ANP	RNP	CLP	AVG
w/o Decoy-focus (λ_{focus}=0)	-0.651	-0.786	-0.501	-0.646
w/o Clean-sep (λ_{sep}=0)	-0.600	-0.755	-0.249	-0.535
w/o Hidden-covert (λ_{covert}=0)	-0.522	-0.601	-0.360	-0.494
DTB (full)	-0.776	-0.795	-0.853	**-0.808**

Decoy-focus: concentrate decoy energy into the decoy-channel subspace; Clean-sep: keep decoy-channel activations relatively low on clean inputs; Hidden-covert: align hidden-trigger representations in non-decoy channels to clean samples of target class $y^{(h)}$.

samples become more stable and better separated after defense, showing stronger inter-class separation. Overall, the visualization is consistent with the quantitative results: sanitization suppresses the decoy backdoor $b^{(d)}$, but can simultaneously activate and consolidate the hidden backdoor $b^{(h)}$, leading to a misleading post-sanitization assessment if evaluation only focuses on the exposed decoy behavior.

F Additional Ablation Results

Ablation on Noise Sampling Number K. We ablate the noise-sampling number K to assess the effect of Monte Carlo ensembling on the *decoy suppression–hidden activation* coupling. As shown in **Fig. 6**, the correlation fluctuates across different K, but overall remains stably negative, with $K = 3$ performing best (AVG -0.808). This indicates that a moderate number of noisy defense proxies provides a more reliable estimate of the local defense neighborhood: too few samples make the coupling signal sensitive to a single perturbation, while too many samples may overly average out the defense direction that most strongly suppresses the decoy pathway. In terms of computational overhead, the additional cost of constructing K surrogate models mainly comes from the extra K forward and backward passes in each mini-batch. For ResNet-18 + CIFAR-10, the total training time is 401.15s, 638.47s, and 975.64s when $K = 0$, $K = 1$, and $K = 3$, respectively (i.e., 1.00×, 1.59×, and 2.43×). Overall, $K = 3$ provides a good trade-off between coupling strength and computational cost.

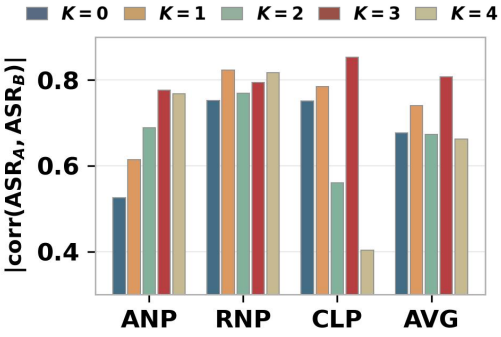


Figure 6: Pearson correlation corr(ASR-$b^{(d)}$, ASR-$b^{(h)}$) under different defenses as a function of the number of defense proxies K. Values closer to -1 indicate stronger decoy-suppression–hidden-activation coupling.

Serial Coupling and Structural Regularization. We further analyze the roles of serial coupling, which encourages $\Delta_{\text{act}} \approx \Delta_{\text{sup}}$ under the defense view, and structural regularization. As shown in **Tab. 13**, removing the serial-coupling term, the structural regularizer, or both weakens the coupling between ASR-$b^{(d)}$ and ASR-$b^{(h)}$: compared with full DTB (AVG -0.808), the Pearson correlations degrade to -0.629, -0.667, and -0.579, respectively. This shows that both components are important for establishing the *decoy suppression–hidden activation* linkage. Serial coupling provides an objective-level constraint that aligns the

Table 13: Ablation on serial coupling and structural regularization. Pearson correlation between ASR-$b^{(d)}$ and ASR-$b^{(h)}$ under different defenses.

Setting	ANP	RNP	CLP	AVG
w/o Serial Coupling	-0.536	-0.687	-0.665	-0.629
w/o Structural Reg.	-0.669	-0.844	-0.487	-0.667
w/o Serial & Struct.	-0.659	-0.798	-0.281	-0.579
DTB (ours)	-0.776	-0.795	-0.853	**-0.808**

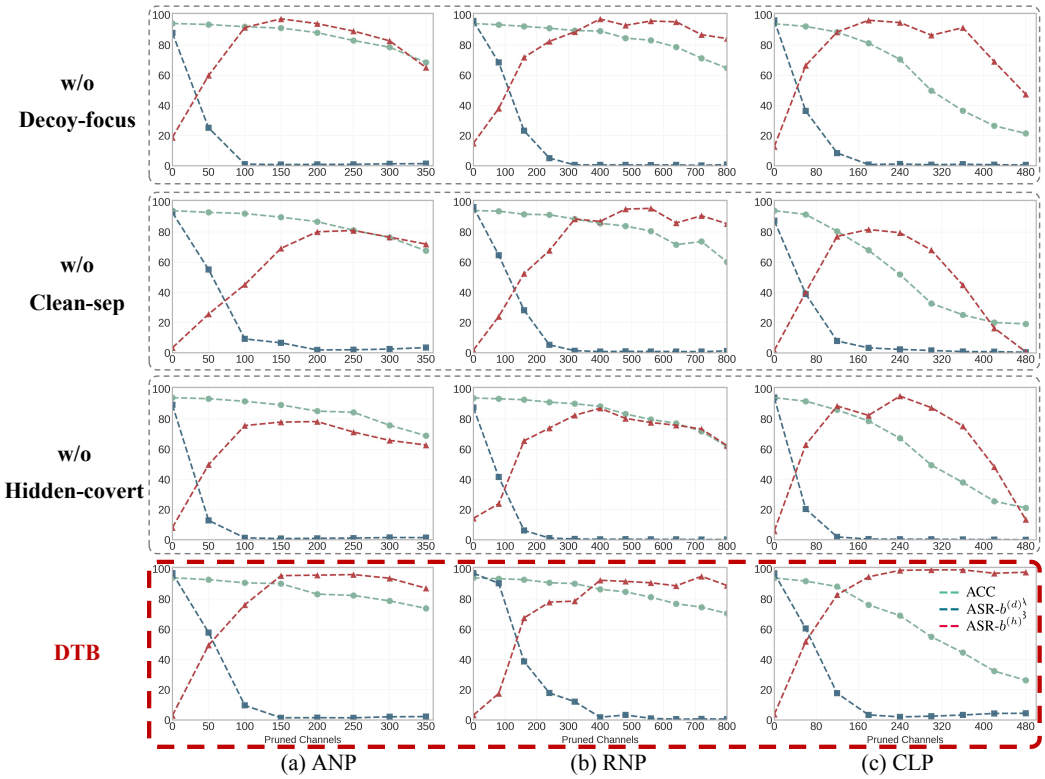


Figure 7: Ablation of DTB structural regularizers under three defense algorithms (ANP/RNP/CLP). Each row removes one component (w/o Decoy-focus, w/o Clean-sep, or w/o Hidden-covert), and the last row reports the full DTB (highlighted by the red dashed box). The x-axis indicates the defense strength.

magnitude of hidden activation with decoy suppression, while structural regularization shapes the representation space so that the decoy and hidden pathways respond differently to sanitization. Their combination yields the most stable cross-defense consistency.

Structural-Regularization Ablation. **Tab. 12** and **Fig. 7** report the ablation of DTB's structural regularizers, where $\mathrm{corr}(\text{ASR-}b^{(d)}, \text{ASR-}b^{(h)})$ closer to -1 indicates that suppressing the decoy backdoor is more strongly coupled with activating the hidden backdoor. The full DTB consistently yields a stronger and more stable negative correlation under ANP/RNP/CLP (AVG: -0.808), while removing any component substantially weakens this correlation.

The key reason is that many sanitization defenses rely on channel-importance or suspiciousness rankings. When the decoy signal is structurally concentrated in a relatively exposed channel subset and its overlap with both clean discriminative features and hidden-backdoor-carrying channels is sufficiently small, sanitization is more likely to suppress decoy-related channels. This reduces $\text{ASR-}b^{(d)}$, while the hidden pathway is less likely to be removed and can become activated after defense. The degradation of w/o Clean-sep is the most pronounced under CLP ($-0.853 \rightarrow -0.249$): once decoy channels are frequently activated on clean inputs, suppressing them no longer corresponds to targeted decoy suppression, but instead affects entangled clean/decoy representations. This is also reflected in **Fig. 7**, where $\text{ASR-}b^{(h)}$ becomes harder to increase monotonically under stronger defenses and is more prone to drops. Similarly, removing Decoy-focus or Hidden-covert reduces channel-level separability between the decoy and hidden backdoors, so sanitization is less concentrated on decoy-related channels and more likely to degrade the hidden pathway as well, resulting in a weaker and less consistent effect across defense algorithms.

Curve Analysis of Serial Coupling and Structural Regularization. As shown in **Fig. 8**, we report the evolution of the model's ACC, the decoy-backdoor attack success rate $\text{ASR-}b^{(d)}$, and

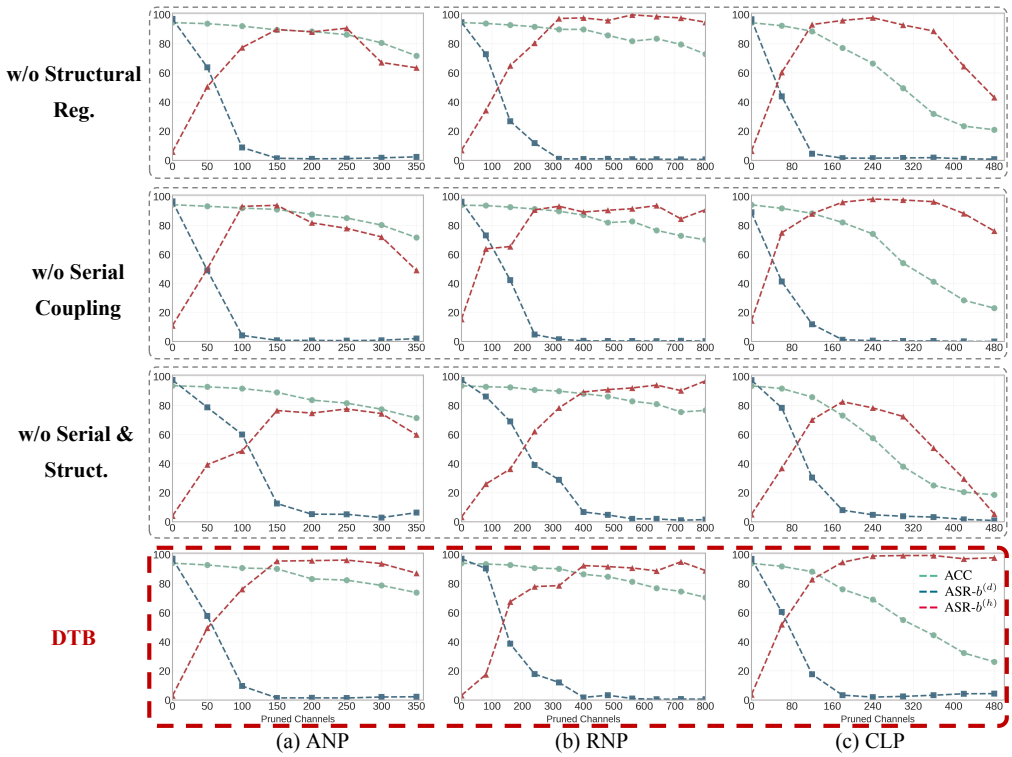
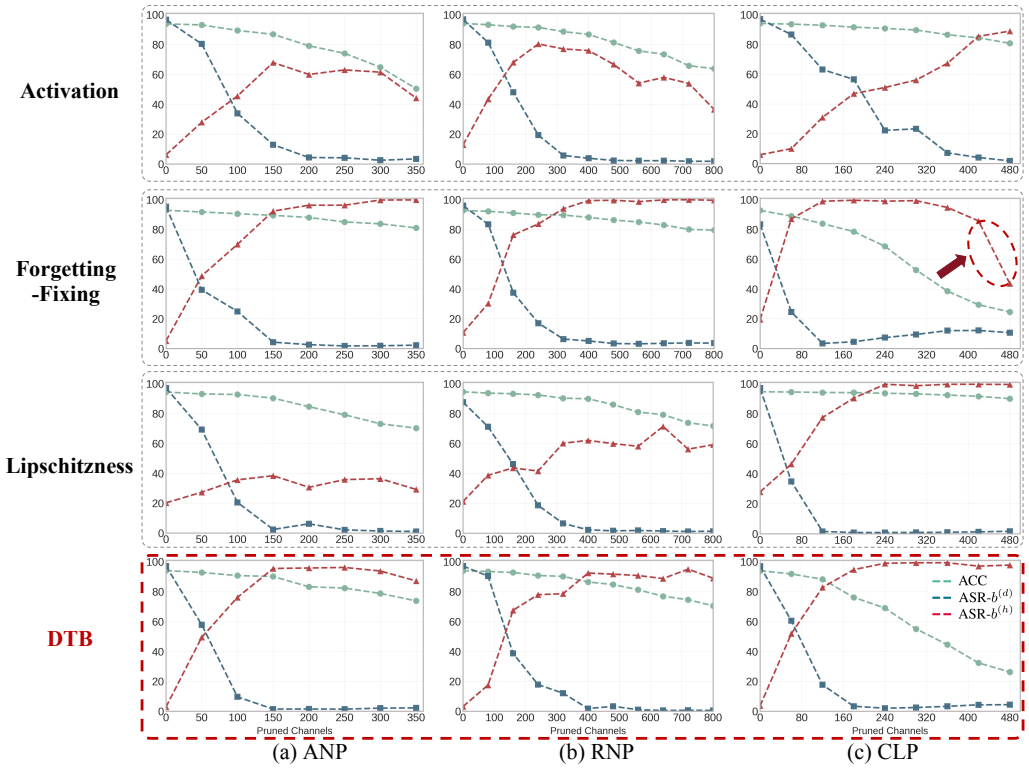


Figure 8: Ablation comparison of serial coupling and structural regularization under three sanitization defenses (ANP/RNP/CLP).

the hidden-backdoor attack success rate ASR-$b^{(h)}$ as the sanitization strength increases under three representative methods, ANP, RNP, and CLP. We include three ablated variants: removing the structural regularizer (w/o Structural Reg.), removing the serial coupling (w/o Serial Coupling), and removing both (w/o Serial & Struct.). When either serial coupling or structural regularization is removed, the decrease of ASR-$b^{(d)}$ and the increase of ASR-$b^{(h)}$ no longer exhibit a stable inverse transition. Under some defenses, ASR-$b^{(h)}$ plateaus or fluctuates inconsistently with ASR-$b^{(d)}$. In contrast, full DTB shows a more coherent serial response across all three defense views: as the defense becomes stronger, ASR-$b^{(d)}$ is steadily suppressed, while ASR-$b^{(h)}$ remains high more consistently. These curve-level observations corroborate the correlation results in **Tab. 13** and show that the two components act complementarily: serial coupling controls the direction of the response, while structural regularization makes this response more stable across defense algorithms.

Different Channel Selection Strategies. **Fig. 9** reports the full trajectories of model performance as a function of the sanitization strength under ANP, RNP, and CLP, where the x-axis denotes the number of pruned channels. Each subplot simultaneously presents clean accuracy (ACC) and the attack success rates of the decoy and hidden backdoors, ASR-$b^{(d)}$ and ASR-$b^{(h)}$. From top to bottom, the rows correspond to three channel-selection baselines: Activation, Forgetting-Fixing, and Lipschitzness.

We observe substantial differences across channel-construction strategies in the *decoy suppression* $\rightarrow$ *hidden activation* behavior. The **Activation** strategy exhibits limited capability to trigger the hidden backdoor under ANP, where ASR-$b^{(h)}$ is difficult to saturate stably; under CLP, ASR-$b^{(h)}$ increases as sanitization strengthens, yet the overall improvement is relatively slow. The **Forgetting-Fixing** strategy can induce a high ASR-$b^{(h)}$ in most settings, but shows a clear instability region under CLP. The **Lipschitzness** strategy fails to form an effective trigger under ANP. In contrast, **DTB** yields a lower initial ASR-$b^{(h)}$ in the released model. As sanitization gradually strengthens, ASR-$b^{(d)}$ is continuously suppressed and approaches zero, while ASR-$b^{(h)}$ increases more steadily and remains



(a) ANP (b) RNP (c) CLP

Figure 9: Under three sanitization defenses (ANP/RNP/CLP), ACC and ASR-$b^{(d)}$/ASR-$b^{(h)}$ curves versus sanitization strength for different channel selection strategies (Activation, Forgetting-Fixing, Lipschitzness) and DTB.

at a high level. This suggests that the contribution-based channel pool used in DTB better captures the channels that real defenses are likely to affect, thereby producing a more reliable suppression signal for conditional hidden activation.

Curve-Level Analysis of Sampling Number K. **Fig. 10** reports the ablation curves with respect to the noise sampling number K. When K takes a moderate value, such as the highlighted $K = 3$, the model exhibits a more stable serial response under all three defense views: as the sanitization strength increases, ASR-$b^{(d)}$ keeps decreasing and approaches zero, while ASR-$b^{(h)}$ rises more consistently and remains high. When K is small, such as $K = 0$ or $K = 1$, the optimization objective is more likely to be dominated by a single stochastic perturbation instance, resulting in higher-variance coupling signals and more fluctuating triggering trajectories across different sanitization strengths. When K is further increased, such as $K = 4$ or $K = 5$, multiple samples improve average robustness over a broader perturbation set, but overly strong averaging can dilute the gradient drive toward the critical coupling direction. This weakens the negative co-movement between ASR-$b^{(d)}$ and ASR-$b^{(h)}$, which is particularly pronounced under the CLP view. Therefore, $K = 3$ provides a practical balance between variance reduction, coupling strength, and computational overhead.

G Future Work and Potential Defenses

DTB is mainly validated on image classification tasks. Future work can further investigate whether similar defense-triggered mechanisms may also emerge in more broadly used language models [64], multimodal systems [52, 43], and their safety review pipelines. Meanwhile, our current experiments focus on publicly reproducible sanitization methods. Subsequent work can further examine more complex and dynamic review settings, so as to more comprehensively understand the behavioral changes that may be induced by the sanitization process.

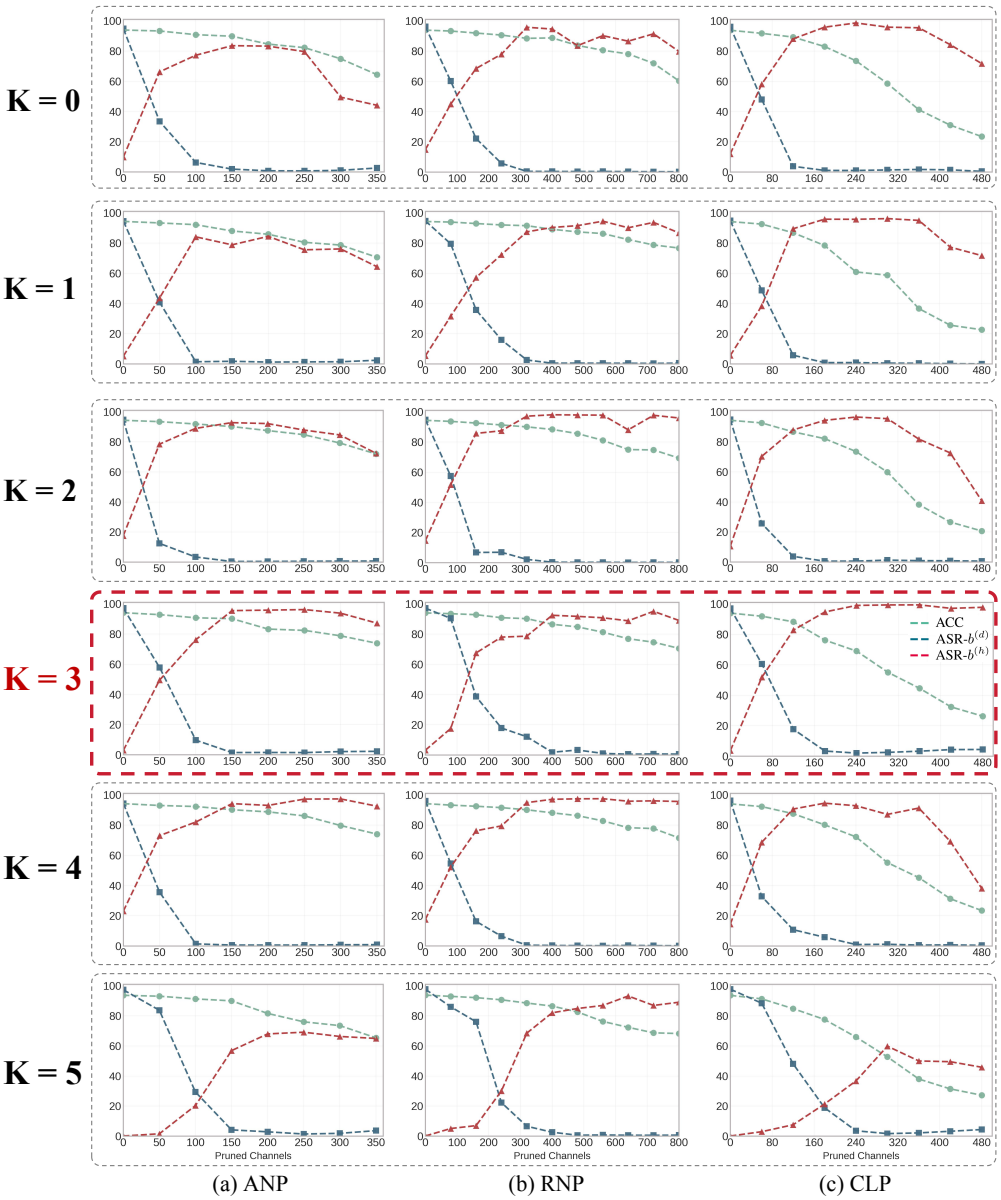


Figure 10: Ablation on the number of noisy defense proxies K under the ANP/RNP/CLP defense views.

For defending against DTB, an important direction is to develop an integrated "sanitization–rechecking" defense pipeline. Since the sanitization process itself may become the condition that activates the hidden backdoor, defenders should not only check whether the original backdoor has been suppressed, but also systematically compare model behaviors before and after sanitization to identify newly emerging anomalous signals or target behaviors. Such rechecking can also be combined with input-level trigger detection to reduce the risk that hidden triggers are activated during deployment. More broadly, future defense mechanisms should move beyond simply evaluating whether existing anomalies are removed, and further assess whether sanitization induces new anomalous behaviors, thereby building a more complete pre-deployment security pipeline.

H Impact Statement

The proposed Defense-Triggered Backdoor (DTB) aims to reveal a dual-use security risk in the backdoor sanitization process: after a backdoor is detected and a defense is applied, the defense operator may inadvertently activate a hidden backdoor, thereby leading to *illusory safety* after defense. In this case, the defended model may be mistakenly labeled as *defended/safe* and used for downstream tasks or released to open-source platforms; however, in reality, it may still carry a more stealthy and harder-to-remove target backdoor. Our goal is not to facilitate malicious misuse, but to help defenders evaluate whether the sanitization process truly removes backdoors and whether it may induce new hidden behaviors. Based on these findings, we call on model deployers and platform reviewers not to judge model safety solely by the disappearance of exposed triggers after sanitization, but to further examine behavioral changes before and after defense as well as potential residual risks. We believe these practices can help build more robust pre-deployment security pipelines for third-party models.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction accurately summarize the proposed DTB threat, the main technical idea, and the empirical scope. The claims are supported by experiments across multiple datasets, architectures, trigger pairs, and representative backdoor sanitization methods.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The paper discusses limitations and future directions in the Future Work and Potential Defenses section, including extensions beyond image classification and the need to study more dynamic review and sanitization pipelines.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes].

Justification: The paper provides the formal definitions, assumptions, and objective formulations used to define DTB, including the conditional-triggering formulation, threat model, and optimization objectives. The paper does not rely on separate theorem-proof style theoretical claims beyond these formal definitions and formulations.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The paper specifies the datasets, architectures, triggers, defense baselines, evaluation metrics, training objectives, and key hyperparameters. Additional implementation details, defense descriptions, and hyperparameter settings are provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The experiments use publicly available datasets and defense implementations. We have released the source code, implementation details, and instructions for reproducing the main experimental results at `https://github.com/Trusted-LLM/DTB`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The main paper reports the key experimental setup, datasets, architectures, defenses, and evaluation metrics. The appendix provides additional training details, optimizer settings, trigger settings, surrogate-defense construction, and hyperparameters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No].

Justification: The paper reports extensive results across datasets, architectures, defenses, and trigger combinations, but does not systematically report error bars or confidence intervals for all main tables. Some experiments, such as localization, are repeated over 100 runs, but full statistical uncertainty reporting is not included.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The experiments are conducted on NVIDIA H200 GPUs with 141GB memory. The appendix further reports representative training-time overhead for different numbers of surrogate models K, including the measured runtime for ResNet-18 on CIFAR-10.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The paper studies a security vulnerability in backdoor sanitization for defensive awareness. It discusses potential misuse and mitigation directions, and does not involve human subjects, private data, or deployment on real users.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The Impact Statement discusses the dual-use nature of DTB, including the risk that sanitization may create misleading post-defense safety and the positive role of the work in improving post-sanitization rechecking and defense pipelines.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release pretrained models, scraped datasets, or other high-risk deployable assets. The work focuses on experimental analysis using public vision datasets and discusses defensive safeguards and post-sanitization rechecking.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The paper properly credits the original datasets, model architectures, attack methods, and defense methods through citations. All existing public datasets and implementations are used for research purposes following their intended terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper does not introduce or release a new dataset, benchmark, pretrained model, or other standalone asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing experiments or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals or an equivalent approval/review based on the requirements of your country or institution were obtained?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing experiments, human subjects, or user studies, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval or equivalent may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity, if applicable, such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: The core method development and experiments do not use LLMs as an important, original, or non-standard component. No LLM-based model, evaluation, data generation, or decision-making component is part of the proposed method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.