Circumventing Backdoor Defenses That Are Based on Latent Separability

Xiangyu Qi* **Tinghao Xie*** **Saeed Mahloujifar** **Prateek Mittal**
Princeton University
{xiangyuqi,thx,sfar,pmittal}@princeton.edu

Abstract

Deep learning models are vulnerable to backdoor *poisoning* attacks. In particular, adversaries can embed hidden backdoors into a model by only modifying a very small portion of its training data. On the other hand, it has also been commonly observed that backdoor poisoning attacks tend to leave a tangible signature in the latent space of the backdoored model *(i.e. poison samples and clean samples form two separable clusters in the latent space)*. These observations give rise to the popularity of *latent separability assumption*, which states that the backdoored DNN models will learn separable latent representations for poison and clean populations. A number of popular defenses (e.g. Spectral Signature, Activation Clustering, SCAn, etc.) are exactly built upon this assumption. However, in this paper, we show that the latent separation can be significantly suppressed via designing adaptive backdoor poisoning attacks with more *sophisticated poison strategies*, which consequently render state-of-the-art defenses based on this assumption less effective (and often completely fail). More interestingly, we find that our adaptive attacks can even evade some other typical backdoor defenses that do not explicitly build on this separability assumption. Our results show that adaptive backdoor poisoning attacks that can breach the latent separability assumption should be seriously considered for evaluating existing and future defenses.

1 Introduction

Overparameterized deep learning models can fit complex datasets perfectly and generalize well on i.i.d. data distributions. However, the strong expressivity of these models also render them susceptible to adversarial attacks on their training data [13, 10, 26, 36] — by excessively fitting those maliciously added/manipulated data samples, the resulting models can suffer from significant performance degradation or targeted mispredictions.

As one of the most typical examples, *backdoor poisoning attack* [10, 4, 33, 23] is one extensively studied setting for such training set attack. In a typical backdoor poisoning attack, an adversary constructs a *poisoned dataset* by injecting into a clean dataset with a small amount of *poison samples*, each of which contains a *backdoor trigger* (e.g. a specific pixel patch) and is labeled as a specific *target class*; while a DNN model trained on this poisoned dataset will be backdoored in that they tend to learn an artificial correlation between the backdoor trigger and the target class (i.e. learn a *backdoor*). This class of attacks are stealthy and threatening, because the backdoored models usually behave normally on normal samples but can misclassify any (or a considerable amount of) samples to the target class as long as the pre-designed backdoor trigger is applied to these samples.

Despite the stealthiness in terms of model performances under standard evaluation metrics (e.g. prediction accuracy), it has been commonly observed [32, 3, 31, 11] that backdoor poisoning attacks

* The first two authors contributed equally to this work.

Preprint. Under review.

(a) No Poison (b) BadNet [10] (c) Blend [4] (d) CL [33] (e) TaCT [31]

Figure 1: PCA [24] visualization of latent separability characteristic on CIFAR-10 [14]. Each point in the plots corresponds to a training sample from the target class. Caption of each subplot specifies its corresponding poison strategy. To highlight the separation, all poison samples are represented by red points, while clean samples correspond to blue points. Refer Appendix C for details.

tend to leave a tangible signature in the latent space of the backdoored models. As visualized in Figure 1, poison samples and clean samples of the target class consistently form two separable clusters in the latent space of corresponding backdoored models, across a diverse set of different backdoor poisoning attacks [10, 4, 33, 31]. These observations give rise to the popularity of a *latent separability assumption*, which states that the backdoored DNN models will learn separable latent representations for poison and clean populations. From a defender's perspective, such separation characteristic provides a natural opportunity for designing backdoor defenses. A number of state-of-the-art defenses [32, 3, 11, 31] explicitly base their designs on the latent separability assumption, where techniques from robust statistics and clustering analysis are applied for identifying "abnormal separation" and filtering out potential poison samples from the training set, before the set is applied for downstream usage.

Following the widely observed empirical evidences for latent separability assumption, a natural question to ask is: ***Is the latent separability between poison and clean populations really an unavoidable consequence of backdoor poisoning attacks?*** Several recent work [27, 35, 7, 25, 5, 37] claim that the latent separability can indeed be suppressed, and state-of-the-art defenses can thus be bypassed. However, these work make *a very strong assumption on the control of the whole training process* of attacked models, which is far beyond the standard threat model of backdoor poisoning attacks that only allows data poisoning. On the other hand, another recent work [31] does follow the standard data poisoning threat model and claim that a source-specific attack (i.e. TaCT in Figure 1e) can reduce latent separability. However, as shown in the figure, *there is still a boundary that can separate* the clean and poison populations even in the two dimensional projection plane, although their distances are slightly reduced by the proposed method. Actually Tang et al. [31] themselves also show that an improved latent space cluster analysis suffices to perfectly separate poison and clean samples for TaCT. Thus, we point out that **the proposed question still has not been well studied** and an in-depth exploration on this question is in need.

This work aims to fill the blank. Specifically, we attempt to design adaptive backdoor poisoning attacks that can induce the backdoored models to learn hard-to-distinguish latent representations for poison and clean samples. After a lot of empirical attempts, we find that there are **two insights that are helpful for designing the desired adaptive attacks**. First, the poisoning strategy should prevent the model from learning an abnormally strong signal for the backdoor trigger. Second, the poisoning strategy should avoid the situation where a simple shortcut rule is sufficient to fit the poison samples.

We design our adaptive poisoning strategy following these two insights, and it turns out that the resulting design has a surprisingly simple idea. In brief, different from traditional backdoor poisoning attacks, after planting backdoor trigger to a set of samples, we do not mislabel all of them to the target class. Instead, we **randomly** keep a part of them (say 50%) still correctly labeled as their semantic ground truth, and label only the rest to the target class. We call the manipulated samples still holding their ground truth labels "**cover**" samples, and the other part with their labels changed to the adversary-specified target class "**payload**" poison samples. Intuitively, the presence of cover samples can serve as regularizers that can penalize the backdoor signal in the learned latent representations. On the other hand, this simple adaptation also makes the correlation between the backdoor trigger and the target class significantly more complicated. Given an input with the backdoor trigger, it has to overfit a much more complicated boundary that should decide when to classify it to the target class and when to classify it to its ground truth label. Conceptually, we expect such complication will also enforce backdoored models to learn more complicated latent representations for backdoor

poison samples such that there is no longer a simple separation boundary that can divide poison and clean populations. Indeed, our empirical study shows that **after a model fits both the cover and payload samples, the distinguishability between poison and clean samples are significantly suppressed** (see Figure 2), while the backdoored model still learns the backdoor correlation and allows a non-trivial attack success rate.

With our adaptive backdoor poisoning attacks, we show that existing state-of-the-art backdoor samples detectors [31, 11, 3, 32] built on the latent separability assumption can be bypassed. More interestingly, we find that our adaptive attacks can even evade some other typical backdoor defenses [8, 15, 34, 18] that do not explicitly build on this separability assumption. By our study, we argue that one should be cautious in the usage of the latent separability assumption during designing a backdoor defense, and adaptive backdoor poisoning attacks that can breach this assumption should be seriously considered for evaluating existing and future defenses. Our code is publicly available [1] for reproduction.

2 Background and Related Work

Data Poisoning Attack. To build large datasets, data collection procedures are typically automated and harvest information from the open world, and the data labeling processes are also often outsourced to third-party labors. Such intensive third-party involvement and the large scale of these data thus make it intrinsically challenging to conduct manual supervision over the datasets creation. On the other hand, the strong expressivity of machine learning models also render them intrinsically susceptible to adversarial manipulations on their training data. These two vulnerabilities jointly lead to the threats of data poisoning attacks [2, 13, 26, 22, 36]. A typical data poisoning adversary has the ability to inject a small number of *poison samples* into the training set of the victims, and the victims who apply the contaminated set for training will get abnormal models. Those poison samples are usually carefully crafted by the adversary, to ensure models that fit these samples will have intended malicious behaviors.

Backdoor Poisoning Attack. Backdoor poisoning attacks [10, 4, 33, 1, 23, 16] constitute a special type of data poisoning attacks. In this setting, a typical adversary constructs poison samples simply by adding a *backdoor trigger* (e.g. a specific pattern like a pixel patch) to a set of clean samples and label them to a specific *target class*. This naturally creates an artificial correlation between the backdoor trigger and the target class, and models that fit these poison samples can be *backdoored* in a sense that they tend to learn this adversarially created correlation (i.e. the backdoor). This class of attacks are considered stealthy and threatening, because the backdoored models perform normally under standard evaluation metrics.

Latent Separability in Backdoor Learning. A commonly observed phenomenon following backdoor poisoning attacks is that — models trained on the poisoned dataset will learn very different latent representations for backdoor and clean samples in the target class, which form two separate clusters (see Figure 1). This phenomenon is first identified by Tran et al. [32], and is then **utilized** [32, 3, 11, 31] **to detect and defend against backdoor poisoning attacks**. Basically, one can perform cluster analysis on the latent representation space to detect bimodality and identify the separation boundary — target class can thus be identified, and poison samples can also be eliminated by simply dropping all samples from the poison modal. Although different techniques are applied, these defensive methods all **explicitly assume the separability** between poison and clean samples and indeed work quite well against many existing attacks — in this work, we phrase the term **latent separability assumption** to denote this common assumption.

Adaptive Attacks for Suppressing Latent Separability. To our best knowledge, the first work that attempts to challenge the latent separability assumption is Shokri et al. [27], which successfully bypass existing backdoor detection algorithms by maximizing the indistinguishability of the latent representations of poisoned and clean data. **However, this work assumes a much stronger threat model** where adversaries not only control the training data but also control the whole training process — thus they can directly encode the latent indistinguishability requirement into the training objectives of the attacked models. Several more recent work [35, 7, 25, 5, 37] that also study this problem all follow the same threat model to Shokri et al. [27]. Perhaps, a more relevant work is Tang et al.

[1] `https://github.com/Unispac/Circumventing-Backdoor-Defenses`

[31], which points out that their source-specific poisoning attack (see Figure 1e) can reduce latent separability. However, as shown in the figure, *there is still a boundary that can separate* the clean and poison populations although their distances are reduced by the proposed method, and actually Tang et al. [31] themselves also show that an improved latent space cluster analysis suffices to perfectly separate poison and clean samples of this attack. **Thus, it is still unclear whether a pure *poisoning adversary* can overcome the separation phenomenon to evade backdoor defenses.** In this work, we fill out the blank and design adaptive backdoor poisoning attacks that can significantly suppress the latent separability characteristic.

3 Problem Formulation

In this section, we specify our notations used across the paper (Section 3.1), and define the threat model (Section 3.2) of our adversaries. Finally, we formulate the latent separability and objectives of the desired adaptive attacks (Section 3.3).

3.1 Notations

In this paper, we consider image classification with DNN models as standard playgrounds for backdoor attacks. Specifically, we denote a neural network as $\mathcal{F}_\theta : \mathcal{X} \mapsto [C]$, where θ are trainable parameters of the neural network, $\mathcal{X}$ is the image input space, C is the number of classes, and $[C] := \{1, 2, \ldots, C\}$. We decompose $\mathcal{F}_\theta$ as $\mathcal{F}_\theta = l_\theta \circ f_\theta$, where l_θ is the last linear prediction layer that transforms latent representations into final output labels, and f_θ is the feature extractor before the last linear prediction layer. For a given input $x \in \mathcal{X}$, we denote $f_\theta(x) \in \mathcal{H}$ the latent representation of x w.r.t model $\mathcal{F}_\theta$, $\mathcal{H}$ the latent representation space, $\mathcal{F}_\theta(x) = l_\theta \circ f_\theta(x)$ the predicted label by the model. For the data poisoning setting, we denote the clean training set as $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \ldots, n\}$, which are drawn from a benign data distribution $\mathbb{B}$. For backdoor poisoning attack, we denote the backdoor trigger planting strategy as a transformation $\mathcal{T} : \mathcal{X} \mapsto \mathcal{X}$, which adds trigger to an input image; and the adversary's poison label flipping strategy is denoted as $\mathcal{L} : \mathcal{X} \times [C] \mapsto [C]$. We use $\mathcal{J} := \{j_1, \ldots, j_p\}$ to denote indices of the p data points that are chosen to be used for constructing poison samples. The resulting poisoned training set is denoted as $\mathcal{D}_{\text{poison}} = \{(\tilde{x}_i, \tilde{y}_i) \mid i = 1, \ldots, n\}$, where

$$\tilde{x}_i = \begin{cases} \mathcal{T}(x_i), & i \in \mathcal{J} \\ x_i, & \text{otherwise} \end{cases} \qquad \tilde{y}_i = \begin{cases} \mathcal{L}(x_i, y_i), & i \in \mathcal{J} \\ y_i, & \text{otherwise} \end{cases}$$

Besides, $\rho = p/n$ is used to denote the poison rate and t is the attacker-specified target class.

3.2 Threat Model

We consider a standard backdoor poisoning attack adversary that has control over a small portion $\rho = p/n \ll 1$ of the victim's training set $\mathcal{D}$. The adversary can modify the controlled p samples and turn the clean training set into a poisoned set $\mathcal{D}_{\text{poison}}$. Besides, the adversary can not control any subsequent procedure of model training or deployment. The adversary aims at injecting a backdoor into the victim's model by poisoning the victim's training data. Specifically, the adversary aims to find a trigger planting strategy $\mathcal{T}$ and a poison flipping strategy $\mathcal{L}$, such that the **victim (backdoored) model** $\mathcal{F}_{\theta'}$ trained on $\mathcal{D}_{\text{poison}}$ satisfies: $\mathbb{P}_{(x,y)\sim\mathbb{B}}[\mathcal{F}_{\theta'}(\mathcal{T}(x)) = t] > \mathcal{A}$ and $\mathbb{P}_{(x,y)\sim\mathbb{B}}[\mathcal{F}_{\theta'}(x) = y] \approx \mathbb{P}_{(x,y)\sim\mathbb{B}}[\mathcal{F}_\theta(x) = y]$, where F_θ is the **clean model** trained on the benign training dataset $\mathcal{D}$. That is, the backdoored model should classify an input with trigger to a specified target class t with a nontrivial probability larger than $\mathcal{A}$, while keeping approximately the same performance to that of a clean model on clean inputs.

Besides, the adversary may choose an enhanced test-time trigger planting strategy $\mathcal{T}'$ which plants stronger trigger pattern to the test-time input. A typical example is the image blending based triggers (e.g. [4, 33]) with $\mathcal{T}(x) = (1 - \mathcal{M}) \odot x + \mathcal{M} \odot [(1 - \alpha)x + \alpha T]$, where $\mathcal{M}$ is the binary mask of the trigger, T is the trigger patch, α is the blending rate. The adversary may choose a smaller α for constructing poison training samples (to evade poison samples detectors applied to training set), while choose a larger α for activating the backdoor during test-time (for higher attack success rate).

It is also important to note that, unlike the study on adversarial examples [30, 21] that mainly focus on well defined ℓ_p-norm bounded perturbations, in backdoor poisoning attacks, there is no generally

accepted standard for constraining patterns of backdoor triggers. In general, patterns of backdoor triggers can be very diverse, including patch [10], image blending [4], logo [19], sinusoidal signal [1], natural reflection [20], and even common object [17], etc. For this reason, following the practice of previous work, we do not precisely define the boundary for the backdoor trigger planting function $\mathcal{T}$ and $\mathcal{T}'$. However, we require all trigger planting should be moderate such that samples planted with backdoor trigger still keep their original semantics.

3.3 Latent Separability for Backdoor Defense

Given a poisoned dataset $\mathcal{D}_{\text{poison}}$, one can train a base model $\mathcal{F}_{\theta'} := l_{\theta'} \circ f_{\theta'}$ via running a standard training script h on $\mathcal{D}_{\text{poison}}$, i.e. $\theta' = h(\mathcal{D}_{\text{poison}})$. Latent separability assumption indicates that, in the latent representation space generated by the base model $\mathcal{F}_{\theta'}$, poison samples and clean samples from the target class will form separate clusters. On the other hand, an implication is that latent representations of samples from a non-target class only form a single homogeneous cluster. See Figure 1 for an intuitive sense. Formally, we define an abstract heterogeneous criterion $\mathcal{I}(\cdot, \cdot)$ that takes two sets as input and generates a boolean output, indicating whether the two sets are heterogeneous (i.e. form different clusters). Following the notations in Section 3.1, we use $H_B^c = \{f_{\theta'}(\tilde{x}_i) | i \notin \mathcal{J} \wedge \tilde{y}_i = c\}$ and $H_A^c = \{f_{\theta'}(\tilde{x}_j) | j \in \mathcal{J} \wedge \tilde{y}_j = c\}$ respectively to denote representations of clean samples and backdoor poison samples labeled as class c. Then, basically **the latent separability assumption states that** there is a heterogeneous criterion $\mathcal{I}$ such that $\mathcal{I}(H_B^t, H_A^t) = True$ for target class t, and $\mathcal{I}(H_B^c, H_A^c) = False$ for any non-target class c.

Typical latent separability based poison detectors [32, 3, 11, 31] will run clustering algorithm on $H^c = \{f_{\theta'}(\tilde{x}_i) | \tilde{y}_i = c\}$ for each class c. H^c will then be divided to two empirical clusters $\widehat{H}_B^c$ and $\widehat{H}_A^c$, where $\widehat{H}_A^c$ is the suspected poison cluster. Class c is identified as a potential target class if $\mathcal{I}(\widehat{H}_B^c, \widehat{H}_A^c)$ outputs "*True*". Then, the dataset will be cleansed by simply removing $\widehat{H}_A^c$ for all potential target classes c.

In this work, **we develop adaptive backdoor poisoning attacks** such that the latent separability will be suppressed. Ideally, given the criterion $\mathcal{I}$ used by a defense, $\mathcal{I}(H_B^t, H_A^t)$ should output "*False*". Meanwhile, the attack should still achieve non-trivial attack success rate, i.e. test-time backdoor samples with the backdoor trigger will still be misclassified to the target class with high probability.

4 Adaptive Backdoor Poisoning Attacks

In this section, we introduce our adaptive backdoor poisoning attacks. We first introduce some heuristic insights (Section 4.1) that motivate our design, and then we show how a simple adaptation can incorporate these two insights (Section 4.2). Finally, we introduce some extension that can further boost the adaptivity (Section 4.3).

4.1 Heuristic Insights for Designing Adaptive Poison Strategies

Before jumping to the adaptive backdoor poisoning attacks, it is helpful to first think about why the latent separability characteristic commonly arises in backdoor learning. There are two thoughtful perspectives that may explain this. The first perspective [32] attributes the separation to **the dominant impact of backdoor triggers** in the prediction process of backdoored models. The intuition is — in order to "drag" a sample with its own semantic features from one class to another different class, a backdoored model has to learn a strong signal for the backdoor trigger pattern in latent representation space such that the signal can overwhelmingly beat other semantic features to make its dictatorial decision. Secondly, since **the task of fitting backdoor poison samples are often independent** of (or only weakly correlated to) the main learning task, the learning process of a backdoored model can be deemed as a fitting of two different tasks. From this perspective, it is not reasonable to expect that the backdoored models will learn homogeneous latent representations for these two different tasks — backdoored models may just simply learn a separate shortcut rule [9] for fitting those poison samples without connecting them to the more complicated main task.

Inspired by these perspectives, we find that there are **two insights that are helpful for designing the desired adaptive attacks**. First, we expect that a desired adaptive poison strategy should encode certain kind of penalty to prevent the model from learning an abnormally strong signal for the

backdoor trigger. Second, the poisoning strategy should avoid the situation where a simple shortcut rule is sufficient to fit the poison samples. Instead, it should encourage dependency between the task of backdoor learning and the main task that requires more complicated "understanding" of input. This can hopefully force the models to learn similar representations for the two tasks.

4.2 Adaptive Attacks with Random Cover

We design our **Adaptive** poisoning strategy following the insights we introduce in Section 4.1. Recall that, traditional backdoor poisoning attacks will mislabel all poison backdoor samples to the target class. In contrast, our attack, after planting backdoor trigger to a set of samples (randomly sampled from the training set), does not mislabel all of them. Instead, we randomly keep a part of them (say 50%) still correctly labeled as their semantic ground truth, and only mislabel the rest to the target class. We call the manipulated samples that still hold their ground truth labels ***"cover"*** samples, and the other part with their labels changed to the adversary-specified target class ***"payload"*** poison samples (sometimes we simply call them "*poison*"). Formally, the adversary explicitly specifies a *conservatism ratio* $\eta \in [0, 1)$, with which our label flipping strategy formulates as:

$$\mathcal{L}(x_i, y_i) = \begin{cases} t, & \text{with probability } 1 - \eta \\ y_i, & \text{with probability } \eta \end{cases} \tag{1}$$

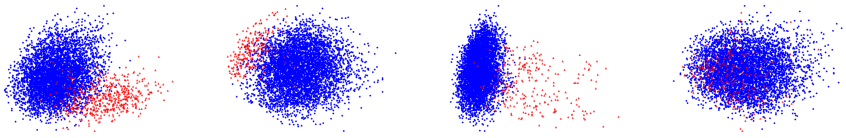
When $\eta = 0$, it degrades to a naive backdoor poisoning attack where all poison samples are labeled to the target class. Formulation 1 could be instantiated by specifying the *cover rate* and *payload rate*, denoted as $\rho_c = \frac{\#cover}{|\mathcal{J}|}$ and $\rho_p = \frac{\#payload}{|\mathcal{J}|}$ respectively, where $\frac{\rho_c}{\rho_c+\rho_p} = \eta$ and $\rho_c + \rho_p = \rho$.

We note that, **this simple adaptation very well incorporates the two insights we have mentioned.** First, if the deep model still learns a dominantly strong signal for the backdoor trigger that can dictatorially vote for the target class, then, it can not well fit those *cover samples*. Thus, the presence of cover samples naturally serves as regularizers that can penalize the backdoor signal in the learned latent representations. On the other hand, the deep model can no longer fit all those samples with the backdoor trigger via a simple shortcut rule. Instead, given an input with the backdoor trigger, it has to overfit a much more complicated boundary that should decide when to classify it to the target class and when to classify it to its ground truth label. The complication of this boundary is easy to understand, because essentially it is randomly generated — we randomly decide which backdoor sample should be mislabeled during constructing them. Moreover, to successfully fit such a complicated decision boundary, the model must also rely on clean semantics that coexist with the trigger in the poison images. This naturally induces a stronger dependency between the task of backdoor learning and the main classification task.

Compared with previous poison strategies, our adaptive strategy only adapts the labeling process of poison samples, and thus can be **directly used as an enhancement for existing backdoor poisoning attacks** like Gu et al. [10] and Chen et al. [4]. In this work, we combine our adaptive strategy with Chen et al. [4]'s blending trigger poisoning attack, and come up with the **Adaptive-Blend** (refer to Appendix A.2 for specific configurations). In Fig 2a-2b, we present a visual comparison between the latent representation space induced by the naive blending and our adaptive blending, and a significant suppression on the latent separation can be observed. Finally, as one may notice, an inevitable consequence of the proposed strategy is a sacrifice of attack success rate (ASR). Conceptually, our adaptive strategy degrades an ideal backdoor attack with ASR $\approx 100\%$ to ASR $\approx \frac{\rho_p}{\rho_p+\rho_c}$ in expectation, because only $\frac{\rho_p}{\rho_p+\rho_c}$ of the trigger-planted samples are labeled to the target class in the training set. **Even so, we argue that it is still threatening enough, since the adversary could still trigger backdoor behaviors with a considerable high probability**. Moreover, in Section 4.3, we show the sacrifice of ASR can actually be avoided with more sophisticated techniques.

4.3 Adaptive Attacks with Diverse and Asymmetric Triggers

In this section, we further illustrate a more sophisticated (and also stronger) design built on the simple paradigm that we introduce in Section 4.2. Two ideas are incorporated. First, the adversary could poison training samples with a less evident trigger (a trigger pattern with a small opacity), and poison test inputs with a more evident trigger (the same trigger pattern with a larger opacity). We



(a) Blend [4] (b) Adaptive-Blend (c) K Triggers (d) Adaptive-K

Figure 2: PCA visualization of latent separability characteristic on CIFAR-10.

show in Appendix E our Adaptive-Blend attack could reach an ASR over 80% with test-time opacity enhanced from $0.2 \rightarrow 0.25$. Second, the adversary could poison the training set with a diverse trigger set (each poison sample is planted with a single trigger randomly picked from a trigger set), and at inference time, adversaries could activate the backdoor by apply a combination of multiple triggers to test samples. The intuition is that, by applying weaker triggers for training samples, it would be easier to evade the training-time poison detectors, and thus the backdoor can still be successfully injected into the victim models. At test-time, the strength of triggers can be adaptively increased, by **trading off the risks of being rejected by test-time input filters**. Moreover, we expect a diverse set of triggers can also lead to more complicated set of representations for backdoor poison samples.

Correspondingly, we design another version of our adaptive attack strategy, **Adaptive-K**, where each poison training sample randomly selects from one of k different triggers, while more than one of the k triggers with larger opacities are applied to each test input at inference time. See Appendix A.2 for details. With our asymmetric triggering mechanism, the ASR could be boosted up easily (>90%). More interestingly, it turns out that *poisoning more than one triggers* enhances not only the ASR, but also the stealthiness of the attack (see Figure 2d).

5 Experiment

5.1 Experiment Setup

Datasets and Models. In this section, we present our adaptive attacks on CIFAR-10 [14]. For all implementations in this section, ResNet-20 [12] is used as the default architecture for building base models. Detailed configurations about dataset split and training details of base models are deferred to Appendix A. In Appendix E, we also present results for other datasets like GTSRB [29] and model architectures like VGG [28].

Attacks. We evaluate our **Adaptive-Blend** and **Adaptive-K** strategies introduced in Section 4. Note that, these two strategies can be directly deemed as adaptive enhancement for the naive blending-trigger backdoor attack (**Blend** [4]) and naive backdoor attack with $k = 4$ triggers (**K Triggers**). We run all of them, and then we compare the naive version and adaptively enhanced version to highlight the effectiveness of our techniques. We also compare with **TaCT** [31], which claims to help reduce the latent separability as well. Detailed attack configurations are described in Appendix A.2.

Defenses. To illustrate the "adaptiveness" of our strategies, we consider state-of-the-art defenses that are explicitly built on latent separability [32, 3, 31, 11]. Note that, these defenses are exactly the main targets that our work aims to bypass. As a bonus, we also evaluate our attacks against some other typical backdoor defenses that do not explicitly rely on latent separability [18, 8, 34, 15]. Specifically, for STRIP [8], we consider both of its use cases, as a training set cleanser **STRIP(C)** and also as a test-time input filter **STRIP(F)**. Refer to Appendix A.3 for detailed configurations of all the defenses.

Metrics. For backdoor poison sample detectors that we evaluate against, we measure their: 1) **elimination rate** ratio of (payload) poison samples that are detected; 2) **sacrifice rate**, ratio of the clean samples falsely eliminated; 3) **Attack Success Rate (ASR)** of models retrained on the cleansed set 4) **clean accuracy** of models retrained on the cleansed set. For (test-time) STRIP [8], we evaluate its effectiveness on 2,000 clean test samples and their 2,000 backdoor versions, by measuring the **elimination rate** and **sacrifice rate**. For Neural Cleanse [34], we report the **anomaly index** of the target class, and the unlearned model's **ASR** and **clean accuracy**. For Anti-Backdoor Learning

	(%)	No Poison	Blend [4]	K Triggers	TaCT [31]	**Adaptive-Blend (Ours)**	**Adaptive-K (Ours)**
Without Defense	ASR	/	92.3	100.0	97.8	53.9	97.3
	Clean Accuracy	92.0	91.7	91.8	91.8	91.7	91.6
Spectral Signature [32]	Elimination Rate	/	61.6	83.1	77.2	61.3	**11.3**
	Sacrifice Rate	15.0	7.2	7.1	7.1	7.3	7.5
	ASR	/	87.5	9.5	94.2	29.8	92.1
	Clean Accuracy	90.9	91.6	91.4	91.5	91.5	91.2
Activation Clustering [3]	Elimination Rate	/	92.3	86.7	98.5	**0.0**	43.1
	Sacrifice Rate	12.3	39.6	37.7	36.5	0.0	37.0
	ASR	/	25.6	3.6	11.8	54.0	95.4
	Clean Accuracy	91.8	90.5	90.6	90.3	91.7	90.4
SCAn [31]	Elimination Rate	/	93.3	89.9	66.7	**0.0**	0.9
	Sacrifice Rate	0.0	1.9	2.5	1.1	0.0	3.1
	ASR	/	10.6	4.1	33.2	54.4	97.1
	Clean Accuracy	92.0	91.8	91.8	91.6	91.7	90.5
SPECTRE [11]	Elimination Rate	/	96.3	99.2	100.0	95.1	**22.5**
	Sacrifice Rate	1.5	0.3	0.3	0.3	0.3	3.1
	ASR	/	8.3	3.0	1.0	2.7	71.6
	Clean Accuracy	91.6	91.9	91.9	91.7	91.8	91.3

Table 1: **Latent separability based defenses against our adaptive attacks on CIFAR-10.** (average)

Defenses→	FP [18]		STRIP (C) [8]				STRIP (F) [8]		NC [34]			ABL [15]		
Attacks ↓	ASR	CA	Eli	Sac	ASR	CA	Eli	Sac	AI	ASR	CA	IP	ASR	CA
Adaptive-Blend	48.1	80.8	1.1	10.3	55.0	91.4	1.2	10.0	1.5	48.7	90.9	0.0	41.7	82.7
Adaptive-K	72.3	79.3	10.0	10.5	96.8	91.1	100.0	10.0	3.7	7.1	90.3	0.9	96.3	89.4

Table 2: **Other defenses against our adaptive attacks on CIFAR-10.** "ASR" for attack success rate, "CA" for clean accuracy, "Eli" for elimination rate, "Sac" for sacrifice rate, "AI" for anomaly index, "IP" for isolation precision, "STRIP (C) & (F)" for STRIP as a poison cleanser and an input filter.

(ABL [15]), we measure its **isolation precision**, **ASR** and **clean accuracy** of anti-backdoored models. To smooth the effect of randomness, we repeat all of our experiments for three times and report the average results. Finally, we notice that data augmentation sometimes benefits defenses, and therefore report the better defense result of the two models with and without augmentation.

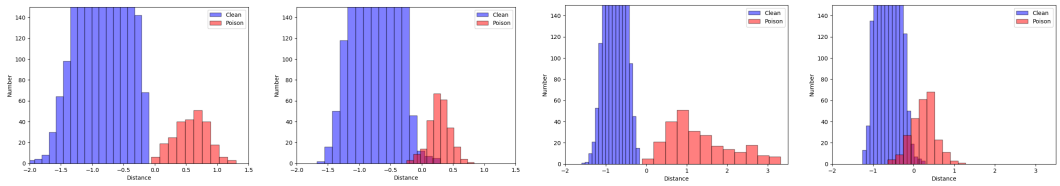
5.2 Attack Performance

We present our results in Table 1 and 2 respectively for latent separability based defenses and other defenses we consider. As shown, when no defense is applied, as expected, Adaptive-Blend reaches an ASR around 50%, while Adaptive-K successfully boosts the ASR up to 97.3% (discussed in Sec‘4).

Circumventing Defenses constructed on Latent Separability As shown in Table 1, our adaptive strategy successfully (ASR is still larger than 20% after defense) evades all the four defenses built on the latent separability. Specifically, none of the other four poisoning backdoor attacks makes thorough all these defenses after cleansing and retraining, while **our Adaptive-K consistently retains a considerable ASR surviving each of them**. Among these defenses, SPECTRE [11] performs as the strongest cleanser. However, against Adaptive-K, SPECTRE either cannot detect the backdoor target class, or couldn't eliminate sufficiently enough poison samples.

Circumventing other Defenses Surprisingly, our adaptive strategy could circumvent other types of backdoor defenses that are not based on latent separability. Tab 2 demonstrates our attacks against several representative defenses: 1) model pruning defense 2) input perturbation defense 3) model inspection defense 4) training process defense. Refer to Appendix B for our complete analysis.

Oracle Visualization To further reveal the extent of latent inseparability of our adaptive strategy, we assume the oracle knowledge of poison samples, and fit the poison and clean latent representations with a Support Vector Machine (SVM [6]). Fig 3 visualizes the distances between target class representations and the SVM hyperplane. Compared to naive strategies (Fig 3a and 3c), our adaptive poisoning attacks (Fig 3b and 3d) bring the poison representations closer to the SVM hyperplane. It's



(a) Blend [4] (b) Adaptive-Blend (c) K Triggers (d) Adaptive-K

Figure 3: Visualization of latent spaces fit by SVM.

worth mentioning that for Adaptive-K (Fig 3d), the SVM cannot fit the poison and clean classification task in the model's latent space. This tells that beyond pulling the clean and poison representations closer, our adaptive strategy actually constructs a **linearly inseparable** poison cluster from the clean cluster. Knowing that even SVMs with oracles cannot correctly fit the task to distinguish poison samples in the latent space, there's no wonder that all defenses (without oracles for sure) based on latent separability fail. More details are discussed in Appendix.

6 Discussions

Ideally, we hope a perfect adaptive attack can make the poison and clean samples completely indistinguishable. This has been achieved under stronger threat model when the the training process is also controlled [27, 35, 7, 25, 5, 37]. In this paper, we take a step further to this goal under poisoning-only threat model. We successfully come up with adaptive backdoor poisoning attacks that can suppress the latent separability and circumvent existing defenses based on latent separability. However, as shown in Fig 3, under oracle visualization, there is still a difference between poison and clean distributions, though the difference is greatly reduced. A key remaining question is — is it possible to achieve the ideal indistinguishable goals with poison-only adversary? We encourage future work to look into this question. Besides, since we are designing attacks that may not be defended by many existing techniques, we note that this exposes existing systems built on these defenses to risks. We encourage future work on designing stronger defenses that resist our attacks.

7 Conclusion

In this work, we point out that latent separability, a widely adopted assumption by state-of-the-art backdoor defenses, could be broken even by a poisoning adversary. We provide our insights on the phenomenon of latent separability, and design adaptive strategies to bypass defenses relying on it. Empirical study and evaluation on various latent space defenses show that our adaptive poisoning attacks indeed reduce the latent separability and render them ineffective. Moreover, our adaptive attacks can also circumvent defenses of other types. We call for every defense designer to take caution when leveraging the latent separability as an assumption.

References

[1] 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.

[2] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[3] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[4] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Xiaodong Song. Targeted backdoor attacks on deep learning systems using data poisoning. *ArXiv*, abs/1712.05526, 2017.

[5] Siyuan Cheng, Yingqi Liu, Shiqing Ma, and Xiangyu Zhang. Deep feature space trojan attack of neural networks by controlled detoxification. *arXiv preprint arXiv:2012.11212*, 2020.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

[7] Khoa Doan, Yingjie Lao, and Ping Li. Backdoor attack with imperceptible input and latent modification. *Advances in Neural Information Processing Systems*, 34, 2021.

[8] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.

[9] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

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

[11] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Spectre: defending against backdoor attacks using robust statistics. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4129–4139. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/hayase21a.html`.

[12] 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.

[13] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[15] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34, 2021.

[16] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *arXiv preprint arXiv:2007.08745*, 2020.

[17] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 113–131, 2020.

[18] 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.

[19] Yingqi Liu, Shiqing Ma, Yousra Aafer, W. Lee, Juan Zhai, Weihang Wang, and X. Zhang. Trojaning attack on neural networks. In *NDSS*, 2018.

[20] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pages 182–199. Springer, 2020.

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[22] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38, 2017.

[23] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

[24] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[25] Yankun Ren, Longfei Li, and Jun Zhou. Simtrojan: Stealthy backdoor attack. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 819–823. IEEE, 2021.

[26] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! Targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018.

[27] Reza Shokri et al. Bypassing backdoor detection algorithms in deep learning. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 175–183. IEEE, 2020.

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

[29] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32: 323–332, 2012.

[30] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[31] 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)*, 2021.

[32] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *arXiv preprint arXiv:1811.00636*, 2018.

[33] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

[34] 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.

[35] Pengfei Xia, Hongjing Niu, Ziqiang Li, and Bin Li. A statistical difference reduction method for escaping backdoor detection. *arXiv preprint arXiv:2111.05077*, 2021.

[36] Mengchen Zhao, Bo An, Wei Gao, and Teng Zhang. Efficient label contamination attacks against black-box learning models. In *IJCAI*, pages 3945–3951, 2017.

[37] Nan Zhong, Zhenxing Qian, and Xinpeng Zhang. Imperceptible backdoor attack: From input space to feature representation. *arXiv preprint arXiv:2205.03190*, 2022.

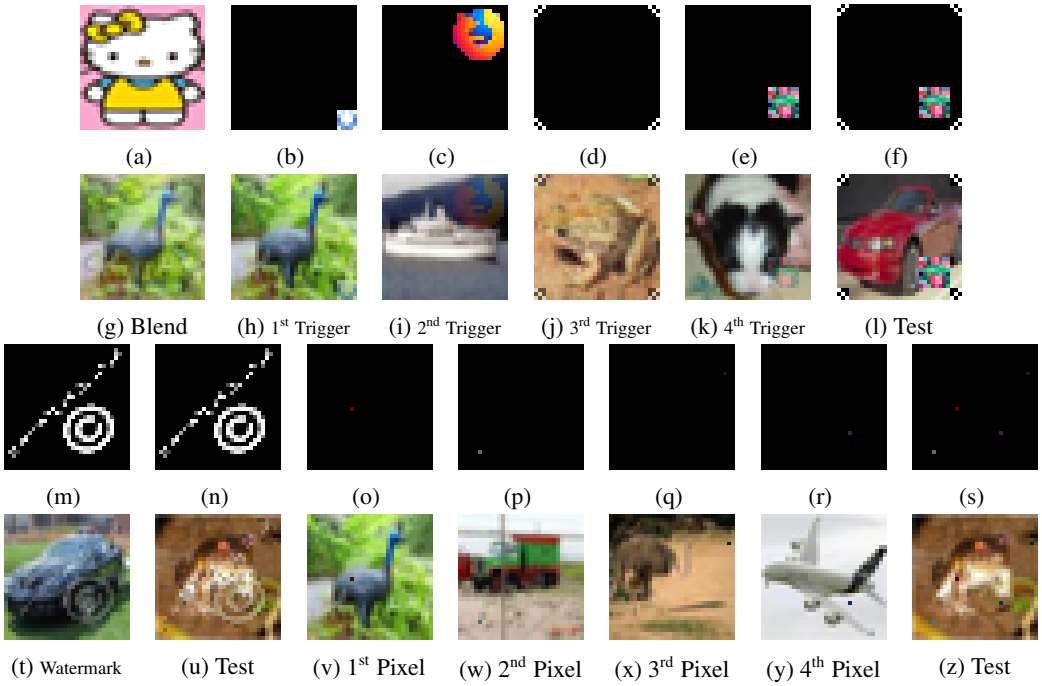


Figure 4: **Poison demonstration.**

A Experiment Configurations

A.1 Computational Environments

All of our experiments are conducted on a workstation with 48 Intel Xeon Silver 4214 CPU cores, 384 GB RAM, and 8 GeForce RTX 2080 Ti GPUs.

A.2 Attack Configurations

For each attack, we reuse the same triggers adopted in the paper. "Blend" and "Adaptive-Blend" use the blending-trigger (Fig 4a) with 20% opacity. "TaCT" uses the trojan square trigger (Fig4k) with 100% opacity. Eventually, "K Triggers" and "Adaptive-K" use the four triggers in Fig 4b-4e with 50%, 20%, 50% and 30% opacities, respectively. Specifically, each of their poison training sample randomly selects one of the four triggers (see Fig 4h-4k), while each test sample uses the enhanced two triggers demonstrated in Fig 4l.

The target class is set to class 0. Poison rates are 0.5% for "Blend" and "K Triggers". For "TaCT" and our "Adaptive-Blend", both the poison (payload) and cover rate are 0.5%. The poison source class of "TaCT" is 1 and its cover classes are 5 and 7. For our "Adaptive-K", the poison (payload) rate is 0.5% and the cover rate is 1.0%.

For all backdoor models, we adopt the standard training pipeline. SGD with a momentum of 0.9, a weight decay of 10^{-4}, and a batch size of 128, is used for optimization. Initially, we set the learning rate to 0.1. On CIFAR-10, we follow the standard 200 epochs stochastic gradient descent procedure, and the learning rate will be multiplied by a factor of 0.1 at the epochs of 100 and 150. On GTSRB we use 100 epochs of training, and the learning rate is multiplied by 0.1 at the epochs of 40 and 80.

We consider widely adopted data augmentations, i.e. `RandomHorizontalFlip` and `RandomCrop` for CIFAR-10 and `RandomRotation` for GTSRB. We notice that data augmentation may affect defense results significantly, while defenders do not know whether to use augmentation or not. Therefore, we report the better defense result of the two backdoor models with and without augmentation. In another sentence, we report **upper-bound results** of the defenses which might be affected by the incorporation of data augmentation during backdoor training. Also, to rule out the effect of

Triggers	Attacks	ASR	Eli	Sac
Hellokitty (Fig 4a, opacity=0.2)	Blend	92.3	26.8	10.0
	Adaptive-Blend	53.9	1.2	10.0
BadNet (Fig 4d, opacity=1.0) + Trojan Square (Fig 4e, opacity=1.0)	K Triggers	100.0	100.0	10.0
	Adaptive-K	99.0	100.0	10.0
BadNet (Fig 4d, opacity=0.5) + Trojan Square (Fig 4e, opacity=0.3)	K Triggers	100.0	100.0	10.0
	Adaptive-K	74.9	53.9	10.0
Firefox (Fig 4c, opacity=0.2) + Trojan Square (Fig 4e, opacity=0.3)	K Triggers	98.7	84.8	10.0
	Adaptive-K	59.5	6.2	10.0

Table 3: STRIP (as an input filter) against adaptive attacks with different trigger selections. The corresponding normalized histograms are shown in Fig 5.

randomness, we train three models on three seeds for each configuration, and report their average results.

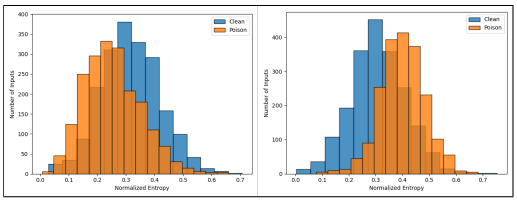
A.3 Defense Configurations

- Spectral Signature [32] removes $1.5 * \rho_p$ suspected samples from every class.
- Activation Clustering [3] cleanses classes with silhouette scores over a threshold (0.15 for CIFAR10 and 0.25 for GTSRB).
- SCAn [31] cleanses classes with scores larger than e.
- SPECTRE [11] removes $1.5 * \rho_p$ suspected samples only from the class with the highest QUE score.
- FP [18] keeps pruning dormant neurons in the last convolution layer until a 10% clean accuracy drop is observed.
- STRIP [8] as a cleanser first estimates the entropy distribution of clean samples on a validation set, selects an entropy threshold with a 10% false positive rate, and eventually removes all training samples with entropy below this threshold.
- STRIP as an input filter is evaluated on 2,000 inputs and their poison counterparts. Again, the entropy threshold is selected with a 10% false positive rate on the 2,000 clean samples, after which all poison samples with entropy below this threshold are filtered.
- NC [34] reverse engineer a trigger for each class with a 2,000-sample validation set (epochs=30, patience=5, batch size=32, initial cost=1e-3, attack success threshold=0.99). Then an anomaly index is estimated for every class. The class with the highest anomaly index (whose mask norm is also smaller than the median mask norm) is determined as the target class for unlearning. Its reversed trigger is then attached to 20% samples of a 5,000 clean samples from clean CIFAR-10 train set, on which the model is retrained to unlearn the backdoor (epoch=1, lr=1e-2).
- ABL [15] first isolates 500 suspected samples (isolation epoch=20, lr=0.1, gradient ascent with 0.5-Flooding), then finetunes the model on the other 49,500 training samples (finetuning epoch=60, lr=0.1 for the first 40 epochs and lr=0.01 for the last 20 epochs), eventually unlearns the model with the 500 isolated samples for another 5 epochs (unlearning epoch=5, lr=5e-4).

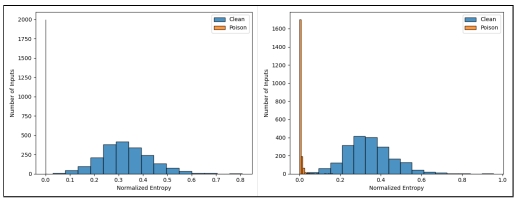
B More Defense Analysis

B.1 Model Pruning Defense

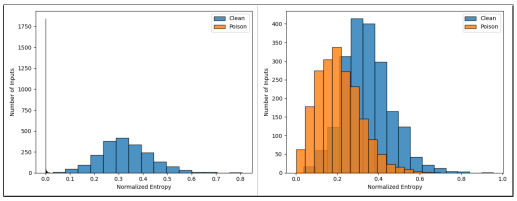
Fine-Pruning (FP [18]) claims when a model is fed with clean inputs, its dormant neurons are more likely to be responsible for the backdoor task. FP eliminates a model's backdoor by pruning these dormant neurons until a certain clean accuracy drop. As shown, our adapitve attacks withstand FP.



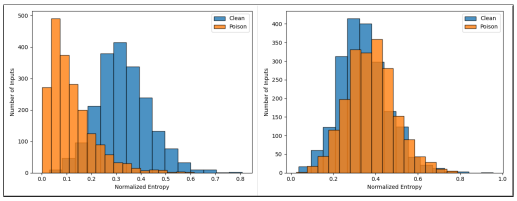
(a) Blend & Adaptive-Blend.



(b) K Triggers & Adaptive-K. BadNet (Fig 4d, opacity=1.0) and Trojan Square (Fig 4e, opacity=1.0) as the inference-time trigger.



(c) K Triggers & Adaptive-K. BadNet (Fig 4d, opacity=0.5) and Trojan Square (Fig 4e, opacity=0.3) as the inference-time trigger.



(d) K Triggers & Adaptive-K. Firefox (Fig 4c, opacity=0.2) and Trojan Square (Fig 4e, opacity=0.3) as the inference-time trigger.

Figure 5: **Normalized entropy histograms of STRIP.** Samples with smaller entropy are more suspected to be poisoned. As shown, our adaptive strategy enforces the model to predict in a conservative way. The Adaptive-Blend poison samples have larger entropy (even larger than clean samples, see Fig 5a), rendering STRIP completely ineffective. In Fig 5b-5d, we show how tradeoff between ASR and stealthiness can be made by an Adaptive-K attacker.

B.2 Input Perturbation Defense

STRIP [8] is a backdoor defense based on the observation that when a poison sample is superimposed by clean samples, the predicted class confidence drops heavily. Our adaptive poisoning strategy breaks this observation, since the model must decide whether to classify a poison sample to the target class by also its clean semantic, thus usually produces a more **conservative** confidence. Tab 2 shows that both versions of STRIP are completely ineffective against Adaptive-Blend (elimination <1.2%). Furthermore, according to the normalized entropy histograms in Fig 5a, Adaptive-Blend poison samples actually have larger entropy than clean samples, which means they are less suspected to be poisoned even than the clean ones.

Adaptive-K also evades STRIP as a cleanser. Nevertheless, we notice that input-filter STRIP successfully eliminates all the enhanced 2-trigger test-time inputs (Fig 4l) of Adaptive-K. We argue that this is a natural tradeoff for a higher ASR (since we are using multiple triggers at the same time, which also enhances the target class confidence), and could be avoided by using one of the symmetric triggers or a less significant asymmetric trigger at inference time. We show in Tab 3 that an Adaptive-K attacker could circumvent STRIP by using the Firefox (Fig 4c, opacity=0.2) +Trojan Square (Fig 4e, opacity=0.3) trigger, where only 6.2% poison samples could be eliminated and the ASR hits approximately 60%.

B.3 Model Inspection Defense

Neural Cleanse (NC [34]) restores triggers by optimizing on the input domain. The authors claim a class with its reversed trigger having an abnormally small norm is more possibly a poisoned target class. Quantitatively, it calculates an anomaly index for each class w.r.t. the reversed triggers' mask norm, where classes with anomaly index >2 are judged as poisoned targets (outliers). Then, the smallest abnormal reversed trigger is patched on a small clean set to unlearn the model's backdoor. As shown in Tab 2, our Adaptive-Blend introduces an average target class anomaly index <2, and keeps a high ASR after unlearning.

B.4 Training Process Defense

Anti-Backdoor Learning (ABL [15]) defense backdoor poisoning via controlling the training process. ABL first isolates 1% training samples with the smallest losses, which the authors claim more possibly to be poison samples. After finetuning on the other (not isolated) training samples, ABL would unlearn the model with these isolated samples. Nevertheless, our adaptive strategy could suppress the loss dropping rate of poison samples (as the our adaptive backdoor task becomes much more complicated and cannot be fit easily, see discussions in Sec 4). As a consequence, ABL can hardly isolate any of our poison samples by observing losses – our adaptive poison strategy circumvents ABL at very the first step (as shown in the "IP" column in Tab 2) of ABL.

C Visualization of Latent Representation Space

To best reveal the "separability", we provide our visualization of latent representation spaces of different attacks in this section. Specifically, we first train a backdoor models for each poisoning attack on its poisoned training set. Then we record the latent representation (the last layer's output, or the linear classifier's input) of target class samples by forwarding them through the backdoor models. Eventually, we visualize these latent representations on 1 or 2 dimensions via PCA [24], t-SNE and oracle projection (SVM, see descriptions in Sec 5.2). For each attack, we train three models with and three models without data augmentation, then visualize each of them.

Fig 6 shows our projections of latent representations onto the top two principal directions. Fig 7 shows our 2-dimensional t-SNE projections of latent representations. While the former two figures only visualize the representations in an unsupervised way, Fig 8 shows how supervised SVMs (with oracle knowledge) could separate the poison and clean representations.

Obviously, in all these visualizations, our adaptive attacks successfully pull the clean and poison clusters closer. In Fig 8d and 8e, we can see that the supervised SVMs cannot separate the poison and clean samples in the latent space (and the other attacks could always be separated with data augmentation).

D Asymmetric Triggers

As discussed in Section 4.2, our adaptive strategy with ASR $\approx \frac{\rho_p}{\rho_p+\rho_c}$ is already threatening enough. In Section 4.3, we briefly mention two ways to boost the attack success rate.

Take Adaptive-Blend as an example. We poison the training set with the trigger opacity $\alpha = 0.2$, while we could activate the adaptive backdoor with a much higher success rate when we tune up the opacity at inference time. As shown in 9, even $\alpha = 0.25$ could boost the ASR to 82.7%. We show in Appendix E.4 an adaptive attack with an asymmetric watermark trigger as another example.

Adaptive-K boosts the ASR in another way, where k diverse triggers are used to poison the training set (each poison training sample only contains one of them), while several triggers are planted simultaneously into a single input at inference time. While each single trigger activates backdoor behavior with an ASR $\approx \frac{\rho_p}{\rho_p+\rho_c}$, several different triggers presenting at the same time is supposed to jointly boost the ASR much higher. In Appendix A.2 we already introduce one possible Adaptive-K asymmetric trigger setting. In Appendix E.4, we show that even as simple as $k = 4$ pixels could work as another set of Adaptive-K triggers.

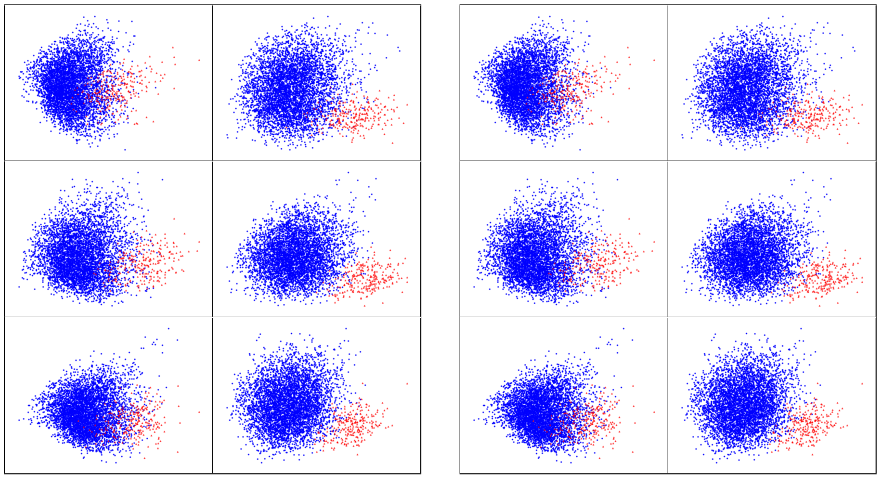
E Full Experiment Results

E.1 CIFAR-10

We provide the three repeated experiment results on CIFAR-10 in Table 4-6 and Table 7-9.

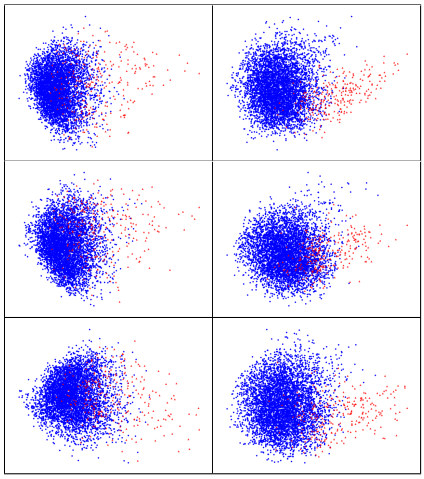
E.2 GTSRB

We also evaluate our adaptive poisoning backdoor attacks on GTSRB [29]. Due to the imbalanced nature of GTSRB and the rotation-based data augmentation, we activate Adaptive-K with another

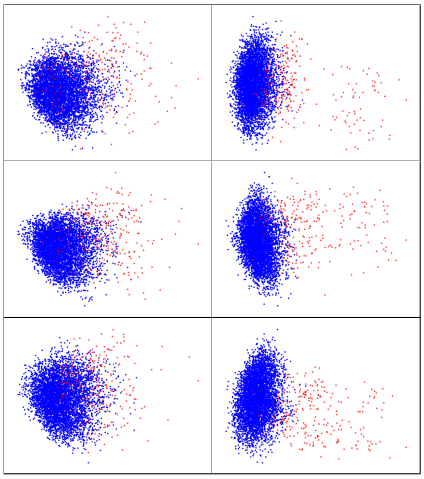


(a) No Poison

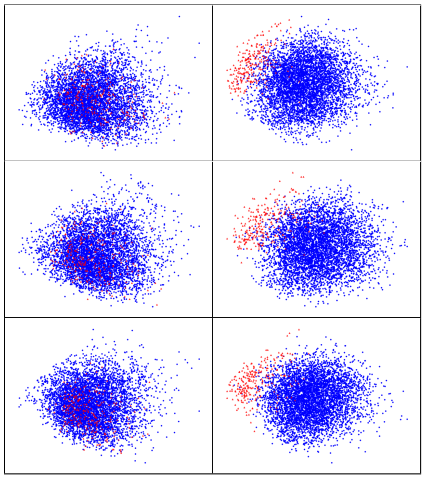
(b) TaCT [31]



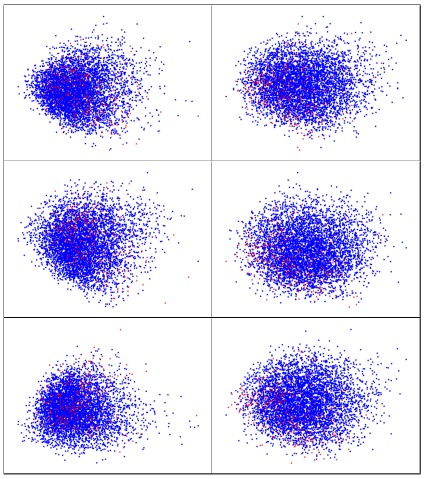
(c) Blend [4]



(d) K Triggers

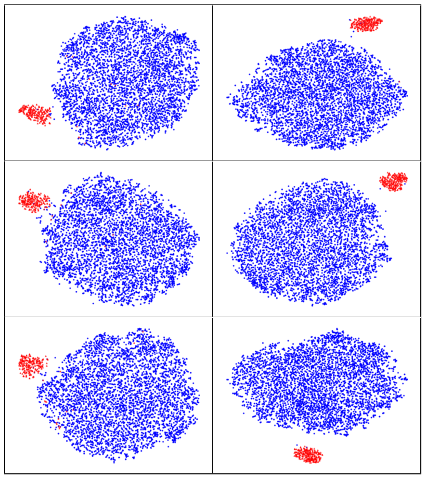


(e) Adaptive-Blend (Ours)

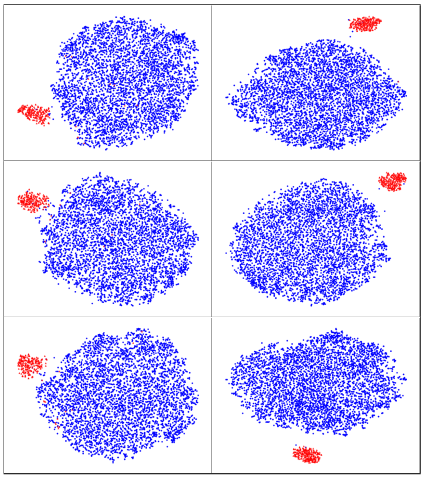


(f) Adaptive-K (Ours)

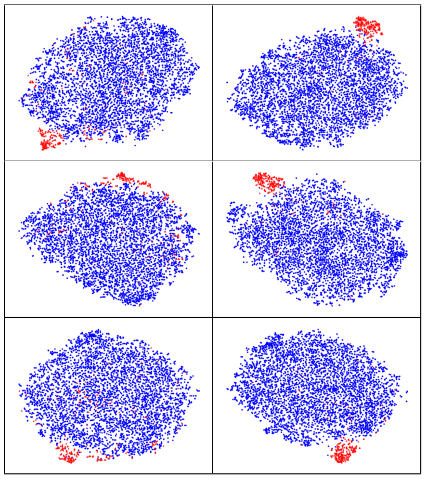
Figure 6: [PCA] Visualization of the latent representation space (CIFAR10). Red points correspond to poison samples and blue points correspond to clean samples.



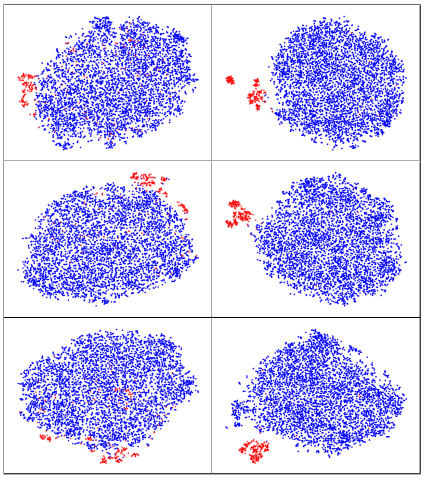
(a) No Poison



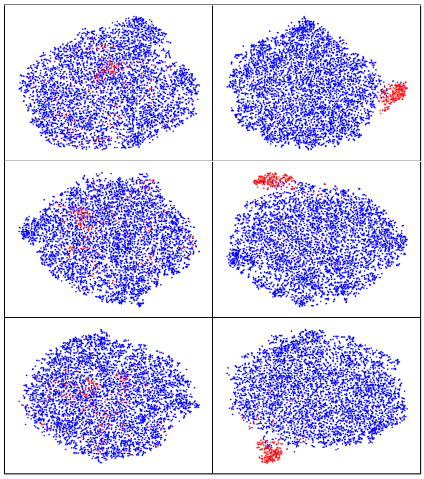
(b) TaCT [31]



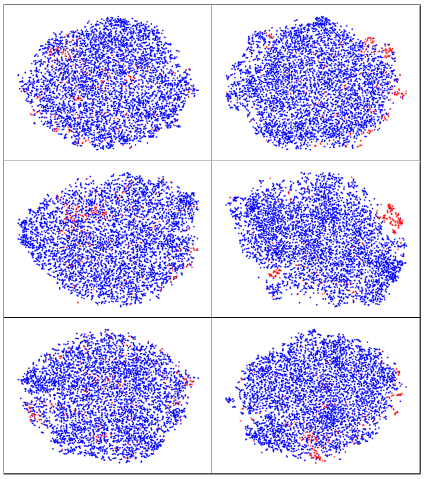
(c) Blend [4]



(d) K Triggers

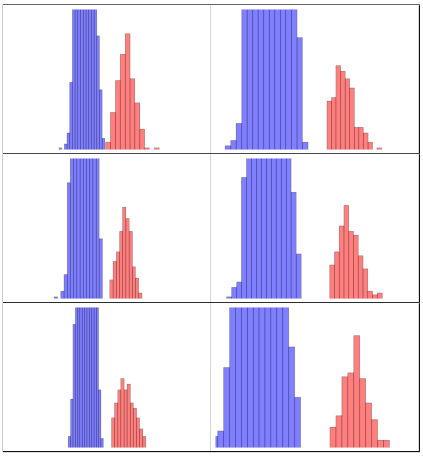


(e) Adaptive-Blend (Ours)

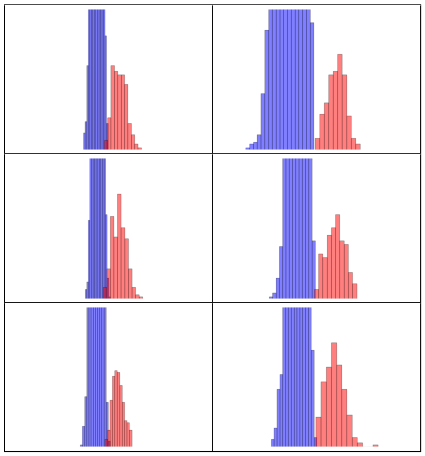


(f) Adaptive-K (Ours)

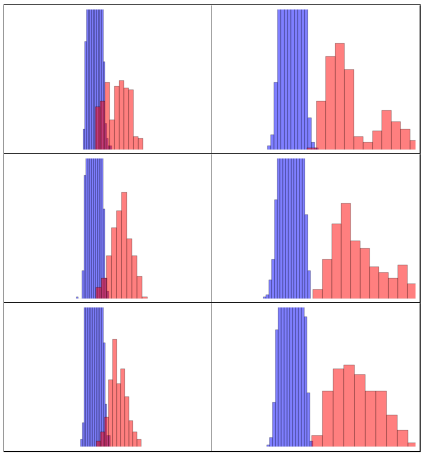
Figure 7: [t-SNE] Visualization of the latent representation space (CIFAR10). Red points correspond to poison samples and blue points correspond to clean samples.



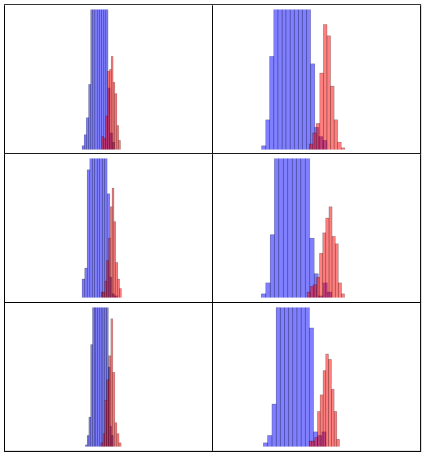
(a) TaCT [31]



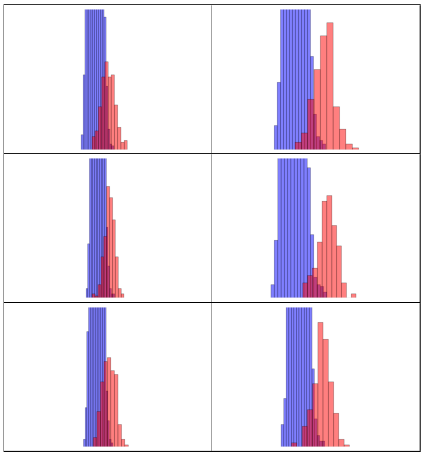
(b) Blend [4]



(c) K Triggers



(d) Adaptive-Blend (Ours)



(e) Adaptive-K (Ours)

Figure 8: [Oracle] Visualization of the latent representation space (CIFAR10). Each plot is a histogram, where the X-coordinate is the **signed** distance to the oracle hyperplane (approximated by SVM) and Y-coordinate denotes the number of samples for the corresponding distance level. The X-coordinate range of all histograms is [-3,3] and the Y-coordinate range is [0, 75]. Bars that exceed the maximum Y range are directly cutoff at the maximum height, since the information is not important. Red bars correspond to poison samples and blue bars correspond to clean samples.

	(%)	No Poison	Blend [4]	K Triggers	TaCT [31]	**Adaptive-Blend (Ours)**	**Adaptive-K (Ours)**
Without Defense	ASR	/	92.3	100	97.7	54.9	93.3
	Clean Accuracy	92	92	91.7	91.7	91.7	91.4
Spectral Signature [32]	Elimination Rate	/	74.4	78	68.8	70	9.6
	Sacrifice Rate	15	7.2	7.1	7.2	7.2	7.5
	ASR	/	81.8	13.6	95	25.5	78.9
	Clean Accuracy	90.6	91.6	91.3	91.8	91.6	91.2
Activation Clustering [3]	Elimination Rate	/	98	76.8	95.6	0	48.4
	Sacrifice Rate	12.3	40.7	35.8	35.9	0	37.2
	ASR	/	2.6	4.9	34.4	54.9	90.9
	Clean Accuracy	91.9	90.4	90.9	90.6	91.7	90.3
SCAn [31]	Elimination Rate	/	92.4	88.4	0	0	0
	Sacrifice Rate	0	3.6	3.8	0	0	0
	ASR	/	13.2	3.1	97.7	56.3	93.3
	Clean Accuracy	92	91.5	91.7	91.7	91.6	91.4
SPECTRE [11]	Elimination Rate	/	97.6	99.2	100	94	67.6
	Sacrifice Rate	1.5	0.3	0.3	0.3	0.3	0.4
	ASR	/	3.6	3.4	1.2	2.7	17.3
	Clean Accuracy	91.8	92.2	92.2	91.8	92	91.4

Table 4: **Latent separability based defenses against our adaptive attacks on CIFAR-10.** (seed #1)

	(%)	No Poison	Blend [4]	K Triggers	TaCT [31]	**Adaptive-Blend (Ours)**	**Adaptive-K (Ours)**
Without Defense	ASR	/	91.2	100.0	98.5	54.7	99.6
	Clean Accuracy	92.1	91.8	91.9	91.8	91.6	91.7
Spectral Signature [32]	Elimination Rate	/	48.0	84.4	82.0	56.8	12.8
	Sacrifice Rate	15.0	7.3	7.1	7.1	7.3	7.5
	ASR	/	91.0	7.8	93.9	32.0	98.7
	Clean Accuracy	90.8	91.7	91.5	91.2	91.7	91.4
Activation Clustering [3]	Elimination Rate	/	82.8	90.4	100.0	0.0	42.0
	Sacrifice Rate	12.3	36.5	36.2	36.0	0.0	37.2
	ASR	/	67.2	3.3	0.6	54.7	97.7
	Clean Accuracy	91.6	91.0	90.4	90.6	91.6	90.1
SCAn [31]	Elimination Rate	/	90.4	91.6	100.0	0.0	2.8
	Sacrifice Rate	0.0	0.1	3.6	3.3	0.0	9.4
	ASR	/	16.3	3.8	0.7	54.7	99.1
	Clean Accuracy	92.1	91.8	91.8	91.2	91.6	88.4
SPECTRE [11]	Elimination Rate	/	96.0	99.2	100.0	94.8	0.0
	Sacrifice Rate	1.5	0.3	0.3	0.3	0.3	8.0
	ASR	/	10.3	1.6	1.5	2.3	98.0
	Clean Accuracy	91.4	91.8	91.9	91.5	91.3	91.1

Table 5: **Latent separability based defenses against our adaptive attacks on CIFAR-10.** (seed #2)

set of inference-time asymmetric triggers, i.e. Fig 4c and Fig 4e, for high ASR. As Table 10 tells, our adaptive strategy could circumvent the latent separability based defenses on GTSRB, as on CIFAR-10.

E.3 Effectiveness on other Architectures

For other popular architectures, *e.g.* VGG-16 and MobileNet-V2, our adaptive strategy also effectively circumvents latent-space defenses. Notice that these architectures have much larger latent spaces – MobileNet-V2 has a latent space of dimension 1280, and VGG-16 has a latent space of dimension 512 (for comparison, 64 for ResNet-20). Therefore, we have to adapt some defense configurations:

- The decision threshold of AC could not be reused. So for VGG-16 and MobileNet-V2, we adapt AC by eliminating any clusters with size <35% of the class size.
- SCAn could not finish computing for MobileNet-V2 after 2h, so we manually reduce the latent representation's dimension to 128 by PCA before SCAn starts processing.

In Table 11, we demonstrate our adaptive attacks with these network architectures on CIFAR-10. As shown, none of these latent separability based defenses could completely eliminate the backdoor.

	(%)	No Poison	Blend [4]	K Triggers	TaCT [31]	**Adaptive-Blend (Ours)**	**Adaptive-K (Ours)**
Without Defense	ASR	/	93.3	100.0	97.3	52.1	99.0
	Clean Accuracy	91.9	91.3	91.7	91.8	91.8	91.7
Spectral Signature [32]	Elimination Rate	/	62.4	86.8	80.8	57.2	11.6
	Sacrifice Rate	15.0	7.2	7.1	7.1	7.3	7.5
	ASR	/	89.8	7.2	93.6	32.0	98.7
	Clean Accuracy	91.3	91.5	91.3	91.5	91.2	90.9
Activation Clustering [3]	Elimination Rate	/	96.0	92.8	100.0	0.0	38.8
	Sacrifice Rate	12.3	41.6	41.1	37.6	0.0	36.5
	ASR	/	6.9	2.5	0.5	52.3	97.7
	Clean Accuracy	91.9	90.0	90.5	89.6	91.9	90.7
SCAn [31]	Elimination Rate	/	97.2	89.6	100.0	0.0	0.0
	Sacrifice Rate	0.0	2.1	0.0	0.0	0.0	0.0
	ASR	/	2.4	5.5	1.2	52.1	99.0
	Clean Accuracy	91.9	92.0	91.9	91.9	91.8	91.7
SPECTRE [11]	Elimination Rate	/	95.2	99.2	100.0	96.4	0.0
	Sacrifice Rate	1.5	0.3	0.3	0.3	0.3	0.8
	ASR	/	10.9	3.9	0.4	3.1	99.5
	Clean Accuracy	91.6	91.8	91.5	91.7	92.1	91.5

Table 6: **Latent separability based defenses against our adaptive attacks on CIFAR-10.** (seed #3)

Defenses→	FP [18]		STRIP (C) [8]				STRIP (F) [8]		NC		[34]	ABL [15]		
Attacks ↓	ASR	CA	Eli	Sac	ASR	CA	Eli	Sac	AI	ASR	CA	IP	ASR	CA
Adaptive-Blend	61.9	81.1	1.6	10.4	52.1	91.4	1.1	10.0	3.7	27.4	89.3	0.0	41.7	80.3
Adaptive-K	68.2	80.3	5.6	10.6	91.5	91.3	100.0	10.0	3.6	1.9	90.7	1.0	96.3	91.0

Table 7: **Other defenses against our adaptive attacks on CIFAR-10.** (seed #1)

E.4 Effectiveness on other Triggers

We demonstrate our adaptive strategy's effectiveness when combined with other triggers. We take BadNet patch trigger (Fig 4d, opacity=1.0), watermark trigger (Fig 4m, opacity=0.2) and k-way (k pixels) attack (Fig 4v-4y) as examples. Specifically,

- the BadNet patch conforms to traditional symmetric trigger setting, where the trigger's opacity is 1.0 at both train time and inference time;
- the watermark trigger is used to poison the training set with opacity=0.2 (Fig 4t) and used to activate the backdoor at inference time with opacity=0.3 (Fig 4u);
- the k-way attack poison training samples with one of the four pixels (Fig 4v-4y), and all four pixels are used together for inference time triggering (Fig 4z).

Results in Table 13 validate that our adaptive strategy is still stealthy across different triggers – successfully evading or reducing the effectiveness latent space defenses. Nevertheless, we want to point out that the attacker's trigger selection might affect the stealthiness of backdoor in the latent.

Defenses→	FP [18]		STRIP (C) [8]				STRIP (F) [8]		NC		[34]	ABL [15]		
Attacks ↓	ASR	CA	Eli	Sac	ASR	CA	Eli	Sac	AI	ASR	CA	IP	ASR	CA
Adaptive-Blend	20.9	79.7	0.8	9.8	56.3	91.4	1.1	10.0	0.6	48.8	89.9	0.0	48.5	85.1
Adaptive-K	85.3	77.2	14.0	10.1	99.7	90.8	100.0	10.0	2.8	3.9	90.5	0.8	96.2	91.0

Table 8: **Other defenses against our adaptive attacks on CIFAR-10.** (seed #2)

Defenses→	FP [18]		STRIP (C) [8]				STRIP (F) [8]		NC		[34]	ABL [15]		
Attacks ↓	ASR	CA	Eli	Sac	ASR	CA	Eli	Sac	AI	ASR	CA	IP	ASR	CA
Adaptive-Blend	61.5	81.6	0.8	10.6	56.6	91.3	1.3	10.0	0.2	1.3	89.6	0.0	34.8	82.7
Adaptive-K	63.3	80.5	10.4	10.7	99.3	91.3	100.0	10.0	4.8	1.6	90.9	0.8	96.3	86.3

Table 9: **Other defenses against our adaptive attacks on CIFAR-10.** (seed #3)

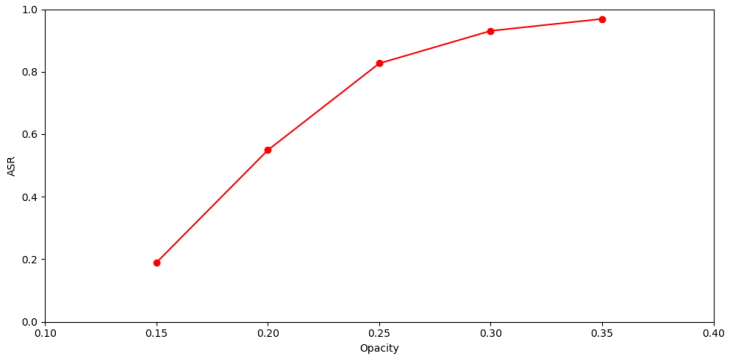


Figure 9: Asymmetrically triggering Adaptive-Blend. The attacker could control the ASR by altering the trigger's opacity at inference time.

Defenses→	No Defense		Spectral Signature [32]				Activation Clustering [3]				SCAn [31]				SPECTRE [11]			
Attacks ↓	ASR	CA	Eli	Sac	ASR	CA	Eli	Sac	ASR	CA	Eli	Sac	ASR	CA	Eli	Sac	ASR	CA
Adaptive-Blend	48.9	97.7	61.1	25.5	30.8	96.7	0.0	0.1	49.2	97.6	28.3	2.3	33.6	97.4	0.0	0.1	39.2	97.1
Adaptive-K	63.1	97.8	74.4	25.4	19.4	96.7	0.0	0.3	67.3	96.7	35.6	4.6	47.4	97.5	0.0	0.1	72.2	96.5

Table 10: **Results of our adaptive attacks on GTSRB.** "Eli" for elimination rate, "Sac" for sacrifice rate, "ASR" for attack success rate and "CA" for clean accuracy.

Defenses↓	**Archs→**	VGG-16		MobileNet-V2	
	Attacks→	Adaptive-Blend	Adaptive-K	Adaptive-Blend	Adaptive-K
Without Defense	ASR	53.6	94.6	60.8	91.3
	Clean Accuracy	93.5	93.3	92.0	92.0
Spectral Signature [32]	Elimination Rate	66.7	63.6	13.5	6.3
	Sacrifice Rate	7.2	7.2	7.5	7.5
	ASR	29.0	82.0	57.7	89.4
	Clean Accuracy	92.9	93.1	91.8	91.8
Activation Clustering [3]	Elimination Rate	0.0	0.0	0.0	0.0
	Sacrifice Rate	0.0	0.0	0.0	0.0
	ASR	53.6	94.6	60.8	91.3
	Clean Accuracy	93.5	93.3	92.0	92.0
SCAn [31]	Elimination Rate	23.7	0.0	0.0	25.6
	Sacrifice Rate	1.7	0.9	0.0	1.6
	ASR	44.2	94.4	60.8	65.9
	Clean Accuracy	93.0	93.1	92.0	91.9
SPECTRE [11]	Elimination Rate	0.0	49.6	10.3	63.6
	Sacrifice Rate	0.8	0.5	0.7	0.4
	ASR	56.5	50.9	55.1	41.9
	Clean Accuracy	93.4	93.0	91.8	92.0

Table 11: **Results of our adaptive attacks on other network architectures.** (average)

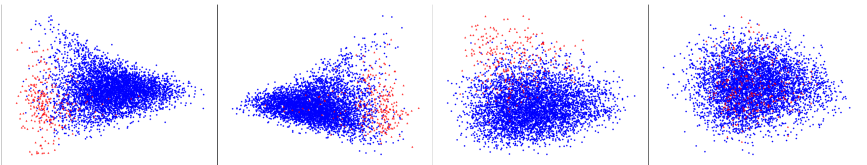
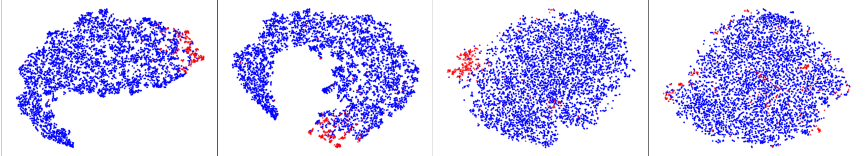
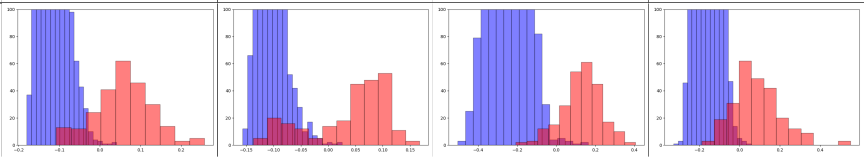
Archs→	VGG-16		MobileNet-V2	
Methods\Attacks	Adaptive-Blend	Adaptive-K	Adaptive-Blend	Adaptive-K
PCA				
t-SNE				
Oracle				

Table 12: Visualization of adaptive attacks on other network architectures in the latent representation space. The corresponding models are trained with data augmentation.

Defenses↓	**Triggers**→	BadNet	Watermark	K Way
Without Defense	ASR	54.6	85.4	66.4
	Clean Accuracy	91.7	91.8	91.3
Spectral Signature [32]	Elimination Rate	15.1	43.9	23.7
	Sacrifice Rate	7.5	7.3	7.4
Activation Clustering [3]	Elimination Rate	16.0	0.0	10.3
	Sacrifice Rate	33.5	31.7	34.1
SCAn [31]	Elimination Rate	36.3	0.0	0.0
	Sacrifice Rate	1.4	0.0	1.2
SPECTRE [11]	Elimination Rate	97.6	93.2	69.7
	Sacrifice Rate	0.3	0.3	0.4

Table 13: **Results of our adaptive attacks with other triggers.** (average)

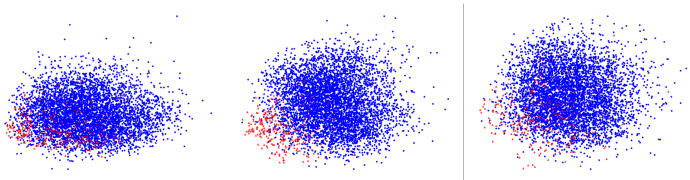
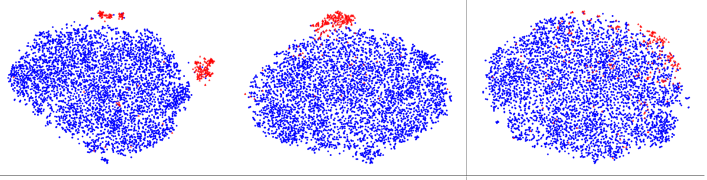
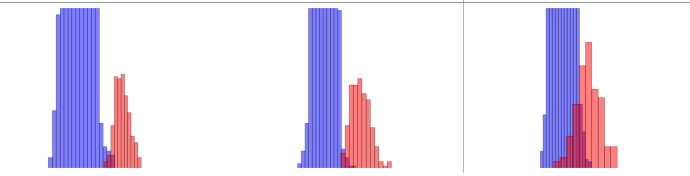
Method \Trigger	BadNet	Watermark	K Way
PCA			
t-SNE			
Oracle			

Table 14: Visualization of adaptive attacks with other triggers in the latent representation space. The corresponding models are trained with data augmentation. See Fig 6, 7 and 8 for our plotting configurations.