Towards A <u>Proactive</u> ML Approach for Detecting Backdoor Poison Samples

Xiangyu Qi
Princeton University

Tinghao Xie
Princeton University

Jiachen T. Wang
Princeton University

Tong Wu
Princeton University

Saeed Mahloujifar
Princeton University

Prateek Mittal
Princeton University

Abstract

Adversaries can embed backdoors in deep learning models by introducing backdoor poison samples into training datasets. In this work, we investigate how to detect such poison samples to mitigate the threat of backdoor attacks. **First**, we uncover *a post-hoc workflow* underlying most prior work, where defenders passively allow the attack to proceed and then leverage the characteristics of the post-attacked model to uncover poison samples. We reveal that this workflow does not fully exploit defenders' capabilities, and defense pipelines built on it are prone to failure or performance degradation in many scenarios. **Second**, we suggest a paradigm shift by promoting *a proactive mindset* in which defenders engage proactively with the entire model training and poison detection pipeline, *directly enforcing and magnifying distinctive characteristics of the post-attacked model to facilitate poison detection*. Based on this, we formulate a unified framework and provide practical insights on designing detection pipelines that are more robust and generalizable. **Third**, we introduce the technique of *Confusion Training* (CT) as a concrete instantiation of our framework. CT applies an additional poisoning attack to the already poisoned dataset, actively decoupling benign correlation while exposing backdoor patterns to detection. Empirical evaluations on 4 datasets and 14 types of attacks validate the superiority of CT over 14 baseline defenses.[1]

1 Introduction

Deep learning relies on large datasets [22, 39, 49]. Yet, the creation of these datasets often involves automation and outsourcing, making it difficult to ensure strict supervision and leaving them vulnerable to *backdoor poisoning attacks* [5, 12, 20]. In these attacks, a typical adversary will manipulate a few training samples by planting a *backdoor trigger* (e.g., a pixel patch) and (mis)labeling them as a *target class*. These manipulated samples are referred to as *backdoor poison samples*,

and the compromised dataset is called a *poisoned dataset*. This manipulation creates a backdoor correlation between the trigger and target class, causing models trained on the poisoned dataset to learn this backdoor while still maintaining normal behaviors under standard evaluation metrics. Backdoor poisoning attacks pose a significant risk as they allow adversaries to stealthily control models. To combat this, *we investigate methods for detecting backdoor poison samples in potentially poisoned datasets* as an additional safeguard before they are used by downstream applications.

Limitations of Prior Works.[2] Prior works on the detection of backdoor poison samples have primarily employed *a post-hoc workflow* [4, 7, 11, 13, 47, 50]. In this workflow, defenders passively allow the attack to first proceed without intervention, by training a model on the given poisoned dataset using a routine procedure (without any defense), resulting in the model being backdoored. By analyzing the model's behaviors, defenders then try to trace the attack back to the poison samples in the dataset. The rationale behind these post-hoc approaches is a passive assumption that post-attacked models (i.e., backdoored models) will spontaneously exhibit distinctive behaviors (e.g., latent separation characteristics [36]) on poison and clean samples, which can be utilized to distinguish between the two populations [4, 13, 47, 50]. However, in practice, these post-hoc approaches are prone to failure or performance degradation in many scenarios, because the characteristics they rely on could often be inherently weak (e.g., when the poison rate is low [13, 50]) or be deliberately suppressed by adaptive attacks [36, 47]. *In this work, we point out that these failure modes should be attributed to the underlying post-workflow which passively counts on post-hoc characteristics that are not within the control of defenders.*

Our Proposal: proactively enforce and magnify distinctive characteristics of the post-attacked model to facilitate poison detection. To alleviate the limitations of the post-hoc workflow, this work advocates for a *proactive mindset*

---

[2]The discussion is strictly within the context of *detecting* backdoor poison samples; our analysis of limitations is not directly applicable to other tracks of defenses that focus on alternative defensive goals (refer Sec § 2.3).

as an alternative. Instead of passively allowing the attack to proceed as per the adversary's expectation and hoping for distinctive behaviors to emerge in the post-attacked model, we propose that defenders should engage proactively with the entire model training and detection pipeline. This involves directly enforcing and magnifying distinctive characteristics of the post-attacked model to facilitate poison detection. The key insight is that defenders have a *"home field" advantage* against backdoor poisoning attacks (that has not been fully exploited by post-hoc approaches), as they still maintain full control over the system space after the dataset has been compromised. This control allows defenders to strategically intervene in the attack process (e.g., post-process data samples, manipulate training procedures) in a way that can proactively enforce post-attacked models to exhibit discriminative characteristics (as per the defenders' expectations), which can be more reliably used to separate clean and poison samples. We formulate this proposal within an abstract framework (Sec § 3.3) and provide practical insights on designing proactive poison detectors that are more robust and generalizable.
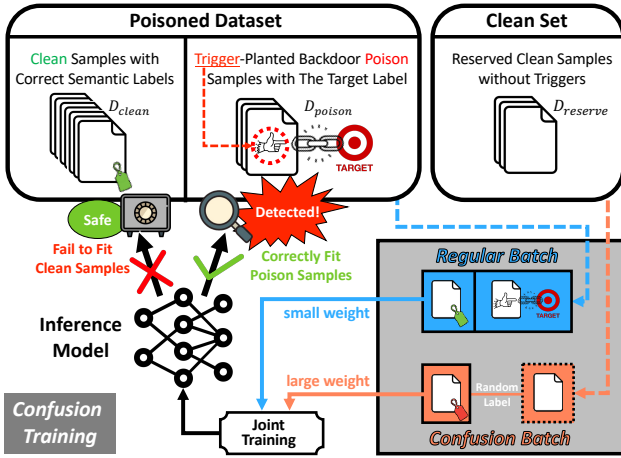

Figure 1: **Overview of our confusion training (CT) method for poison samples detection.** During training the inference model, we utilize randomly labeled clean samples (***confusion batch***) that do not contain triggers to proactively decouple benign correlations. The resulting model thus fails to fit clean samples while still fitting backdoor poison samples correctly.

Confusion Training (CT): a concrete instantiation of our proposal. To showcase how our proposal may empower the advancement of future research on backdoor poison samples detection, we introduce the technique of *Confusion Training* (CT) as a concrete instantiation of the proposal. As illustrated in Fig 1, CT produces an <u>inference model</u> via launching a training procedure jointly on *regular batches* from the poisoned dataset and *confusion batches* consisting of randomly labeled clean samples from a reserved clean set (much smaller than the poisoned dataset). The random labeling decouples

the benign correlations between normal semantic features and semantic labels in the confusion batch. During training, a confusion batch is also assigned a *large weight* while a regular batch is only assigned a small weight. In this way, the confusion batch serves as another set of poison samples that strongly obscure the benign correlations, making the clean data points hard to be fitted. On the other hand, since the confusion batch does not contain backdoor triggers, *correlations between the trigger and target class still remain intact*. Therefore, the resulting inference model fails to fit most clean samples but still correctly fits most backdoor poison samples. This allows defenders to distinguish between clean and poison samples based on the inference model's state of <u>fitting</u>.

Emperical and Theoretical Analysis. We extensively evaluate a diverse set of 14 backdoor poisoning attacks [3, 5, 12, 21, 26, 30, 31, 36, 42, 47, 52, 57] and 4 benchmark datasets, covering domains of both image (CIFAR10 [17], GTSRB [45], ImageNet [8]) and malware (Ember [1]) classifications. By comparing with 14 baseline defenses [4, 7, 11, 13, 16, 18, 19, 23, 47, 48, 50, 55, 56, 60] along with ablation studies, we show the effectiveness and superiority of the Confusion Training (CT) defense, illustrating the potential of the proactive mindset that we promote in this work. We also provide a theoretical analysis in a simplified setting to formally illustrate the working principles of CT in our technical report [37].

Finally, we summarize our contributions as follows:

- We uncover a post-hoc workflow that is prevalent in previous research on detecting backdoor poison samples, and reveal the limitations it imposes on prior arts.

- We propose a proactive mindset as an alternative and formulate it within an abstract framework, providing grounded insights on designing proactive poison detection pipelines that are more robust and generalizable.

- We introduce Confusion Training (CT) as a concrete illustration of our proposal, whose effectiveness is supported by both empirical and theoretical results. We position CT as evidence to showcase how our proposal of the proactive mindset may inspire future advancement in the detection of backdoor poison samples.

2 Preliminaries

In this section, we define our setup (Sec § 2.1), formulate the threat model of backdoor poisoning attacks (Sec § 2.2), and clarify the goals and capabilities of our defense (Sec § 2.3).

2.1 Model, Dataset and Training Procedure

We consider deep neural network (DNN) classification model $f(\cdot; \theta) : \mathcal{X} \mapsto \Delta^C$, where $\mathcal{X} := \mathbb{R}^d$ is the input space, C is the number of classes, Δ^C is the space of probability simplex over C classes, and θ is the model parameters. Note that $f(x; \theta)$

outputs a probability distribution over all possible classes. We also define the hard-label classifier $F(\cdot;\theta) := \arg\max_c f_c(\cdot;\theta)$ that gives the label prediction. In general, a deep neural network $f(\cdot;\theta)$ can be decomposed as $l \circ h(\cdot;\theta)$, where l is the last linear prediction layer, and h is the feature extractor before the last linear layer. Given an input $x \in \mathcal{X}$, we call $h(x;\theta) \in \mathcal{H}$ the *latent representation* of x w.r.t. the model $f(\cdot;\theta)$ where $\mathcal{H}$ denotes the space of latent representation. $l(\cdot;\theta)$ transforms the latent representation into a probability distribution over the classes $l \circ h(x;\theta) \in \Delta^C$. We use $\mathcal{P}$ to denote the clean data distribution over $\mathcal{X} \times \{1,\dots,C\}$, and $D = \{(x_i,y_i)\}_{i=1}^n$ is a clean dataset where each $(x_i,y_i) \sim \mathcal{P}$ sampled independently. We use $\mathcal{A}$ to denote the learning algorithm. Accordingly, $\theta := \mathcal{A}(D)$ is the benign model trained on the clean dataset D. We call $\mathbb{P}_{(x,y)\sim\mathcal{P}}\{F(x,\theta) = y\}$ the *clean accuracy (ACC)* of the classifier $F(\cdot;\theta)$.

2.2 Threat Model

In this paper, we follow the standard threat model of *backdoor poisoning attacks* [5].

Adversary's Capabilities. Similar to most existing backdoor poisoning attacks, the adversary: **1)** can manipulate a limited portion (no more than a half) of the training dataset, **2)** has access to the victim's model architecture, training algorithm, and potential backdoor mitigation techniques. Besides, the adversary has no control over either the training process or the environment where models are deployed.

Notations. We use $\mathcal{T} : \mathcal{X} \mapsto \mathcal{X}$ to denote the adversary's trigger planting strategy that plants the backdoor trigger to a data sample, $\mathcal{L} : \mathcal{X} \times [C] \mapsto [C]$ to denote the adversary's labeling function on poison samples, and t to denote the target class. The adversary will manipulate k samples in the clean dataset D, and we use $\mathcal{J} := \{j_1,...,j_k\}$ to denote indices of the k samples. Then, the resulting poisoned dataset is denoted as $\widetilde{D} = \{(\tilde{x}_i, \tilde{y}_i)|i = 1,\dots,n\}$, where

$$\tilde{x}_i = \begin{cases} \mathcal{T}(x_i), & i \in \mathcal{J} \\ x_i, & \text{o/w} \end{cases}, \quad \tilde{y}_i = \begin{cases} \mathcal{L}(x_i,y_i), & i \in \mathcal{J} \\ y_i, & \text{o/w} \end{cases}. \quad (1)$$

For convenience, we also decompose $\widetilde{D} := D_{\text{clean}} \cup D_{\text{poison}}$, where $D_{\text{poison}} := \{(\tilde{x}_i,\tilde{y}_i)|i \in \mathcal{J}\}$ and $D_{\text{clean}} := \widetilde{D} \setminus D_{\text{poison}}$. That is, D_{clean} corresponds to the portion of clean samples, D_{poison} is the portion of backdoor poison samples. The model $\widetilde{\theta} = \mathcal{A}(\widetilde{D})$ trained on the poisoned dataset $\widetilde{D}$ is the backdoored model. The quantity $\mathbb{P}_{(x,y)\sim\mathcal{P}|_{y\neq t}}\{F(\mathcal{T}(x),\widetilde{\theta}) = t\}$ is the attack success rate (ASR) on the backdoored model.

Attack Goals. The adversary aims to construct a poisoned dataset $\widetilde{D}$ such that the trained model $\widetilde{\theta} := \mathcal{A}(\widetilde{D})$ will be backdoored. In general, a desired backdoor poisoning attack satis-

fies the following conditions (using notations from Sec § 2.1):

$$\mathbb{P}_{(x,y)\sim\mathcal{P}}\left\{F(x;\widetilde{\theta}) = y\right\} \approx \mathbb{P}_{(x,y)\sim\mathcal{P}}\left\{F(x;\theta) = y\right\}, \quad (2)$$

$$\mathbb{P}_{(x,y)\sim\mathcal{P}|_{y\neq t}}\left\{F(\mathcal{T}(x);\widetilde{\theta}) = t\right\} \geq \tau, \quad (3)$$

Eqn 2 requires the backdoored model $F(\cdot;\widetilde{\theta})$ still maintains a level of clean accuracy (ACC) that is comparable to that of a benign model $F(\cdot;\theta)$, in order to ensure the stealthiness of the attack. Eqn 3, on the other hand, asserts that the backdoored model should have a high ASR ($>$ some threshold τ).

2.3 Detecting Backdoor Poison Samples

We investigate methods for *detecting backdoor poison samples as a means of defense* against backdoor poisoning attacks. We focus on offline detection that aims to identify potential backdoor poison samples within a given dataset.

Defender's Capabilities. 1) Autonomy over the poisoned dataset: After acquiring the poisoned dataset, the defender possesses complete autonomy and freedom to manipulate it. This includes the ability to access, scrutinize, and process the dataset, as well as the liberty to independently train models on it; **2)** Access to a small reserved clean set: similar to many prior backdoor defenses [19, 23, 24, 26, 47, 51, 62], our defender has access to a small clean set, e.g., as few as 250 to 2000 samples for CIFAR10 (Figure 5). This small set, while insufficient for training a high-accuracy model, is intended to bootstrap defenses. In practice, this small clean dataset can be in-house data directly generated by the defenders themselves (e.g., a defender could gather photos using their own camera and label them) or data collected from trustworthy sources. Alternatively, Zeng et al. [61] show the feasibility of sifting out a small clean set directly from a poisoned dataset to bootstrap subsequent defenses.

Defender's Goals. The defense intends to isolate a subset D_{suspect} from the dataset $\widetilde{D}$. We denote $\frac{|D_{\text{suspect}} \cap D_{\text{poison}}|}{|D_{\text{poison}}|}$ as the True Positive Rate (TPR) of the defense and $\frac{|D_{\text{suspect}} \setminus D_{\text{poison}}|}{|D_{\text{clean}}|}$ as the False Positive Rate (FPR). The goals of the defense are three-fold: **1)** High TPR: eliminate as many as possible backdoor poison samples; **2)** Low FPR: the quantity of mistakenly eliminated clean samples should be controlled at a low level; **3)** Generalizability: the defense should be effective against different attacks and should keep robustness across different datasets and attack hyperparameters (e.g., poison rate). Note that the defense we consider falls under poison-detection-based defenses [4, 13, 47, 50], which differs from some other defenses with alternative goals (e.g., robust training, post-training backdoor removal) [16, 18, 19, 23, 48, 55, 56].

Poison Detection for Backdoor Defenses. Detecting poison samples provides a *flexible* foundation for countering backdoor attacks. **First**, if one can accurately identify and eliminate backdoor poison samples from a poisoned dataset, the threat of backdoor poisoning attacks can be mitigated at

the outset. The cleansed dataset can be flexibly utilized to train any clean model, irrespective of its algorithm or architecture. This flexibility presents advantages over many defenses (e.g., robust training [16, 18, 48, 56]) that are tied to rigid training algorithms and optimized for specific model architectures. For instance, *one has the flexibility to use a lightweight model (e.g., ResNet18) to cleanse a dataset and then use any model architecture (e.g., ViT [9]) and training algorithms to train an advanced clean model* on the cleansed dataset. **Second**, poison detection can act as a foundational building block in other backdoor defenses. For example, some defenses [16, 18, 56] also rely on isolating (only) a portion of poison samples and strategically employing them to mitigate backdoor learning during training. Thus, *even if the poison detection is imperfect (For instance, TPR is not sufficiently high), it can still be integrated with other techniques to construct effective backdoor defenses.*

3 Methodological Analysis

To tackle backdoor poison samples detection[3], defenders rely on some distinctive characteristics of poison samples, as outlined in Sec § 3.1. We analyze this issue methodologically and critique the post-hoc workflow of state-of-the-art approaches in Sec § 3.2, highlighting its limitations. In Sec § 3.3, we suggest a proactive mindset as an alternative, based on which we formulate a unified framework and offer practical insights on designing proactive poison detection pipelines.

3.1 Defining Poison Samples Detector

Definition 1 (Backdoor Characteristic Function). *A backdoor characteristic function is defined as* $\phi(\cdot;\theta) : X \times [C] \mapsto Z$, *which maps data point* $(x, y) \in X \times [C]$ *to characteristic vectors* $\phi(x, y; \theta) \in Z$, *where* Z *is a characteristic space that facilitates discrimination between clean and backdoor poison samples, and* θ *is the parametrization (if applicable).*

Definition 2 (Backdoor Poison Samples Detector). *A backdoor poison samples detector is the composition of a backdoor characteristic function* $\phi(\cdot;\theta)$ *and a decision function* $\xi : Z \times [C] \mapsto \{0, 1\}$. *Given a data entry* $(x, y) \in X \times [C]$, *the decision function* ξ *takes* $(\phi(x, y; \theta), y)$ *as its input and predicts whether* (x, y) *is a poison sample —* $\xi[\phi(x, y; \theta), y] = 1$ *indicates a positive prediction and* $\xi[\phi(x, y; \theta), y] = 0$ *otherwise. We use* $TPR(\phi, \theta, \xi)$ *and* $FPR(\phi, \theta, \xi)$ *to denote the True Positive Rate and False Positive Rate of the poison detector.*

---

[3]Our analysis strictly pertains to the context of detecting backdoor poison samples in poisoned datasets, which is formally indicated in Definitions 3 and 4. Thus, the analysis is not meant to be directly applied to other tracks of defenses [16, 18, 19, 23, 48, 55, 56] (that do not intend to accurately separate poison and clean samples) within the broad backdoor defenses literature [20].

3.2 The Post-hoc Workflow and Its Limitations

The Post-hoc Workflow. We reveal that most prior studies [4, 7, 11, 13, 47, 50] *on the detection of backdoor poison samples* have employed a post-hoc workflow, where the defender proceeds as follow: **1)** Allow the attack to first proceed as per the adversary's expectation by routinely training a backdoored model $\widetilde{\theta} := \mathcal{A}(\widetilde{D})$ on the poisoned dataset. **2)** Apply some presumed characteristics $\phi(\cdot; \widetilde{\theta})$ of the post-attacked model to project $\widetilde{D}$ into the characteristic space for analysis. **3)** Optimize a decision function ξ on $Z := \{\phi(\tilde{x}, \tilde{y}; \widetilde{\theta}) | (\tilde{x}, \tilde{y}) \in \widetilde{D}\}$ with a goal of maximizing TPR while constraining FPR. **4)** Apply the decision function $\xi[\phi(\tilde{x}, \tilde{y}; \widetilde{\theta}), \tilde{y}]$ for every sample $(\tilde{x}, \tilde{y}) \in \widetilde{D}$, and a sample is identified as a potential poison sample if and only if $\xi[\phi(\tilde{x}, \tilde{y}; \widetilde{\theta}), \tilde{y}] = 1$. The rationale behind this workflow is *an assumption that post-attacked models will spontaneously exhibit distinctive characteristics* on poison and clean samples, which can be utilized to distinguish between the two populations. In Definition 3, we formulate this workflow within an optimization framework.

Definition 3 (The Post-hoc Workflow). *A post-hoc poison detection defender solves the following problem for some (presumed) characteristics* ϕ *of a post-attacked model:*

$$\widetilde{Q} := \max_{\xi} \ TPR(\phi, \widetilde{\theta}, \xi), \quad s.t. \ FPR(\phi, \widetilde{\theta}, \xi) \leq \varepsilon,$$

$$\widetilde{\theta} := \mathcal{A}(\widetilde{D}),$$

where the defender aims to optimize the TPR within an FPR budget ε, *by searching for a decision function* ξ *w.r.t certain characteristic* ϕ *of the backdoored model* $\widetilde{\theta}$.

Failure Modes of Post-hoc Approaches: A Case Study. The most successful examples of the post-hoc workflow are *latent separation based poison detectors* [4, 13, 47, 50]. These methods operate on the premise that backdoored models will learn separate latent representations for clean and backdoor poison samples. Consequently, the latent representation, denoted as $h(\cdot; \widetilde{\theta})$, is perceived as the backdoor characteristic function for poison detection. The next step involves conducting a clustering analysis in the latent representation space to derive a decision function ξ that separates clean and backdoor poison samples. These approaches constitute a state-of-the-art frontier of backdoor defenses, with methods such as Spectral Signature [50] and Activation Clustering [4] being considered as canonical baselines within the literature, while SCAn [47] and SPECTRE [13] have reported nearly perfect results against a diverse set of baseline attacks. These works heuristically optimize the decision function ξ (as described in Definition 3), primarily by refining the clustering analysis. However, they are prone to failure or performance degradation in many scenarios where the assumed characteristics are inherently weak or even deliberately suppressed. For instance, latent separation characteristics can be less discernible [13, 50] when the poison rate is low (Fig 2a). Recently, Tang et al. [47]

and Qi et al. [36] also propose adaptive backdoor poisoning attacks that can even intentionally suppress the latent separation (Fig 2b). Furthermore, these characteristics can also vary across different datasets (Fig 2c).

Besides the latent separation based approaches, Gao et al. [11] assume that backdoor models' predictions on poison samples have less entropy under intentional perturbation, and Chou et al. [7] count on abnormal regions in the backdoored model's saliency map. However, they also suffer from similar failure modes. For example, it is well understood that characteristics assumed by the two works do not hold true for many backdoor attacks with non-local triggers (e.g., [5, 21]).

Methodological Limitations. *We posit that this post-hoc workflow is fundamentally limited in that it does not fully exploit defenders' capabilities.* Particularly, this workflow only passively builds detection pipelines based on some post-hoc characteristics that are not within the control of defenders. However, these (presumed) characteristics could often be inherently weak or even deliberately suppressed by attackers.

3.3 Towards A Proactive Mindset

The "Home Field" Advantage. In this study, we highlight the "home field" advantage held by defenders in the face of backdoor poisoning attacks. This advantage stems from the fact that once attackers have poisoned a training dataset and then subsequently handed it over to defenders, the latter will have *full control and autonomy over both the poisoned dataset and their own operational environment*. Conversely, the attackers will not be able to interfere with any subsequent defense measures taken by the defenders.

We point out that the post-hoc workflow (Definition 3) does not fully exploit this "home field" advantage. It passively allows the attack to first proceed without exerting any intervention, even though it has the capability to do so. As a result, the post-hoc characteristics of the attacked model are completely out of the control of defenders. As we have reviewed in Sec § 3.2, these post-hoc characteristics could inherently fail to emerge *(so we need to proactively enforce them)* and might also be deliberately suppressed by attackers when they design the attack *(so we should not allow the attack to proceed as per the adversary's expectation)*.

A Proactive Mindset: proactively enforce and magnify distinctive characteristics of the post-attacked model to facilitate poison detection. To alleviate the limitations of the post-hoc workflow, we suggest a paradigm shift by promoting a proactive mindset. We encourage defenders to fully exploit their "home field" advantage by engaging proactively with the entire model training and detection pipeline. Specifically, we highlight that defenders have the potential to strategically intervene in the attack process (e.g., post-process data samples, manipulate training procedures) in a way that can proactively enforce post-attacked models to exhibit discriminative characteristics (as per the defenders' expectations), which can

be more reliably used to separate clean and poison samples. Formally, this paradigm shift can be concisely expressed by a key adaptation on the prior framework in Defintion 3, and we present this new formulation in Definition 4 as follows.

Definition 4 (The Proactive Mindset). *A proactive poison detection defender selects a backdoor characteristic function ϕ, and proactively enforces and magnifies this characteristic in a post-attacked model by solving the following problem:*

$$Q^* := \max_{\xi, \mathcal{A}^*} TPR(\phi, \theta^*, \xi), \qquad s.t. \ FPR(\phi, \theta^*, \xi) \leq \varepsilon,$$
$$\theta^* := \mathcal{A}^*(\widetilde{D}),$$

where $\mathcal{A}^*$ is proactively designed by the defender to magnify the intended characteristic to facilitate poison detection.

Corollary 5 (Nondegradation). *For fixed $\phi, \varepsilon, \widetilde{D}$,*
$$Q^* \geq \widetilde{Q},$$
where $\widetilde{Q}$ is the ideally optimal TPR in Definition 3, while Q^* is the counterpart in Definition 4.

Practical Insights. For a simple methodological illustration, in Definition 4, we intentionally adopt a very inclusive variable $\mathcal{A}^*$ to express the proactiveness of our proposal. *In practice, $\mathcal{A}^*$ can encapsulate a number of design choices.* A major design choice that we focus on in this work it the training algorithm that generates the post-attacked model. The key insight is that the goal of $\mathcal{A}^*(\widetilde{D})$ is never to find a model that has high accuracy, but a model that exhibits distinguishable characteristics on poison and clean samples. Thus, an empirical risk minimization procedure $\mathcal{A}$ that aims to best fit the poisoned dataset is not necessarily an optimal take. Instead, we suggest defenders proactively search for specialized training algorithms that can generate better θ^* to facilitate poison detection. The design of confusion training (see Sec § 4 for technical details) in this work is an illustration.

Additionally, Definition 4 also sheds light on other dimensions, including data post-processing. The utilization of conventional data augmentation techniques, while common, can be deemed as a simplistic take on this dimension, and it has been shown to amplify latent separation between poison and clean samples [36]. By explicitly positioning the data post-processing as a design choice, we posit that there might exist more delicate post-processing procedures, which can be utilized to better facilitate poison detection. We encourage future research to investigate what could be an optimal take on this dimension. Besides, the choice of model architecture can also make a difference. For example, against Adap-Blend Attack [36] on CIFAR10, the standard version of ResNet18 exhibits much stronger latent separation than that of a tailored version of ResNet20 (see Fig 2d). Later, in Sec 5.2.3, we also show that model architecture choice can play an important role in the implementation of confusion training.

---

[4]Compared to the standard ResNet18 (11.2M parameters), the tailored version of ResNet20 has much smaller capacity (270K parameters).

(a) High v.s. low poison rate.

(b) Blend v.s. Adap-Blend.
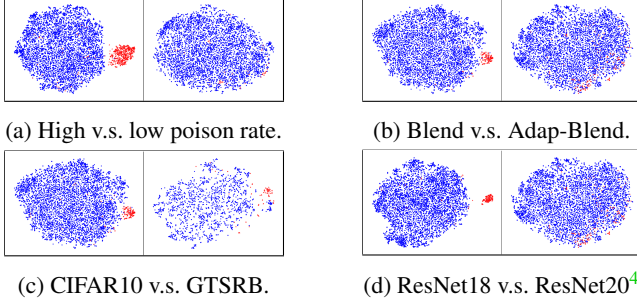
(c) CIFAR10 v.s. GTSRB.

(d) ResNet18 v.s. ResNet20[4]

Figure 2: **Case illustrations.** We visualize the latent representations of the target class training samples. Red and blue points represent poison and clean samples, respectively. Fig 2a compares Blend [5] attacks with 1% and 0.1% poison rates. Fig 2b shows Blend attack and its adaptive version [36] (0.3% poison rate). Fig 2c demonstrates the Blend attack on CIFAR10 and GTSRB (0.3% poison rate). Fig 2d visualizes Adap-Blend attack [36] on the standard ResNet18 and the tailored version of ResNet20 [14] (0.3% poison rate).

4 Confusion Training

To showcase how the proactive mindset that we promote in Sec § 3.3 may empower the advancement of future research on backdoor poison sample detection, we introduce the technique of Confusion Training (CT) as a concrete instantiation of the proposal. This section presents the design of CT. We start from a high-level overview in Sec § 4.1, followed by technical details in Sec § 4.2. To formally illustrate the working principles of CT, we provide a theoretical analysis of CT in a simplified problem setting in our technical report [37].

4.1 Overview

We sketch the confusion training pipeline in Algorithm 1, and present a high-level overview of the approach as follows.

The Reserved Clean Set. The defender initially has a *reserved clean set* (D_{reserve}) at hand, which consists of a small number of clean samples (without backdoor triggers) that serve to (roughly) approximate the clean distribution of D_{clean}. In practice, D_{reserve} is much smaller (Fig 5) than the training dataset $\widetilde{D}$ to an extent that training solely on D_{reserve} is not sufficient to generate an accurate model, as stated in Sec § 2.3. For clarity, in this subsection, we also assume $D_{reserve}$ is correctly labeled, which is consistent with prior arts [19, 47]. In Sec § 4.2, we show this assumption could be removed, and an unlabeled reserved set D_{reserve} is sufficient in practice.

Initialization. We initialize the model with $\mathcal{A}(\widetilde{D})$ in Line 1, which is the backdoored model routinely trained on the poisoned dataset. This initialization provides a prior of the backdoor and empirically makes confusion training more stable.

Confusion Training. Confusion training proceeds by a joint training on the poisoned dataset $\widetilde{D} = D_{\text{clean}} \cup D_{\text{poison}}$ and

---

Algorithm 1 Poison Detection with Confusion Training

Input: Dataset $\widetilde{D} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^n$ to be cleansed, reserved clean set D_{reserve}, loss function ℓ, confusion factor λ, number of confusion iterations m
Output: The cleansed dataset D^*

1: $\theta_{\text{ct}} \leftarrow \mathcal{A}(\widetilde{D})$ ▷ Pretrain
2: **for** $i = 1, ..., m$ **do** ▷ Confusion Training
3:     $(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{Y}}_i) \leftarrow$ a random batch from $\widetilde{D}$
4:     $(\mathbf{X}'_i, \mathbf{Y}'_i) \leftarrow$ a random batch from D_{reserve}
5:     $\mathbf{Y}^* \leftarrow$ random mislabels other than $\mathbf{Y}'_i$
6:     $\ell_{ct} \leftarrow \frac{\ell(f(\widetilde{\mathbf{X}}_i; \theta_{ct}), \widetilde{\mathbf{Y}}_i) + (\lambda-1) \cdot \ell(f(\mathbf{X}'_i; \theta_{ct}), \mathbf{Y}^*)}{\lambda}$
7:     $\theta_{\text{ct}} \leftarrow$ one step gradient descent on ℓ_{ct}
8: **end for**
9: $D^* \leftarrow \widetilde{D}$
10: **for** $i = 1, ..., n$ **do** ▷ Poison Detection
11:     **if** $\tilde{y}_i == F(\tilde{x}_i; \theta_{ct})$ **then**
12:         $D^* \leftarrow$ remove $(\tilde{x}_i, \tilde{y}_i)$ from D^*
13:     **end if**
14: **end for**
15: **return** D^*

---

a set of randomly mislabeled clean samples (referred to as *confusion batch*), as presented in Line 6. Specifically, in each iteration of the gradient descent update, a random batch of clean data $(\mathbf{X}'_i, \mathbf{Y}'_i)$ is sampled from the reserved clean set D_{reserve} (Line 4). The confusion batch $(\mathbf{X}'_i, \mathbf{Y}^*)$ is constructed by reassigning labels (Line 5) to this clean batch with a randomly selected $\mathbf{Y}^*$ different from the ground-truth $\mathbf{Y}'_i$. The model is then trained jointly on the confusion batch $(\mathbf{X}'_i, \mathbf{Y}^*)$ and a batch $(\mathbf{X}'_i, \mathbf{Y}'_i)$ from the poisoned dataset $\widetilde{D}$ (Line 3).

Conceptually, the random mislabeling step in Line 5 *decouples the benign correlations* between normal semantic features and labels in the confusion batch. In practice, we also assign the confusion batch a large weight $\lambda - 1 > 1$. As a result, *the confusion batch serves as another set of poison samples that strongly obscure the benign correlations, making the clean data points hard to be fitted.* On the other hand, since the confusion batch does not contain the backdoor trigger, the backdoor correlation still remains intact. Therefore, at the end of confusion training, *the resulting model θ_{ct} fails to fit most clean samples in D_{clean} but still correctly fits the set of backdoor poison samples D_{poison}.* This allows defenders to distinguish between clean and backdoor poison samples based on the model's *state of fitting* on each sample of $\widetilde{D}$.

Poison Detection. In our design, we employ the *"consistency of prediction"* of the post-attacked model θ_{ct} as the indicator to separate poison and clean populations. As formulated in Line 11, a training sample $(\tilde{x}, \tilde{y}) \in \widetilde{D}$ is suspected as a backdoor poison sample if and only if the model's prediction $F(\tilde{x}; \theta_{\text{ct}})$ on $\tilde{x}$ is consistent with its label $\tilde{y}$, i.e., $(\tilde{x}, \tilde{y})$ is fitted. We note that this is similar to the process of a label-only membership inference [6], and thus we also call the post-attacked

model θ_{ct} as an *inference model* in our context.

The Proactive Mindset. We position confusion training as an illustration of how a proactive defender can directly enforce and magnify distinctive characteristics in post-attacked models to facilitate poison detection. The procedure from Line 1 to Line 8 can be deemed as an instantiation of $\mathcal{A}^*$ that we formulate in Definition 4. In Line 10-14, we take the label prediction of the inference model $\phi(\cdot;\theta_{ct}) := F(\cdot;\theta_{ct})$ as the backdoor characteristic function and employ $\xi(F(\tilde{x};\theta_{ct}), \tilde{y}) := \mathbb{1}(F(\tilde{x};\theta_{ct}) = \tilde{y})$ as the decision function.

4.2 Technical Details of The Design

Now, we delve into the engineering techniques that are employed to convert Algorithm 1 into a practical implementation.

Creating The Confusion Batch. In practice, we have three technical considerations in the creation of the confusion batch.

1) Dynamic Labels. One challenge for implementing confusion training is the overfitting effect of deep learning models. If we use fixed labels for the confusion batch, the inference model θ_{ct} will overfit all samples in both $\widetilde{D}$ and the confusion batch by naive memorization. Then, the decision rule in Line 11 will trivially identify the entire $\widetilde{D}$ as positive. To combat overfitting, the random labels Y^* of the confusion batch are regenerated in every iteration (Line 5). This dynamic labeling endows confusion training with a dynamic nature — since labels are constantly changing, the model can not be optimized by simply memorizing.

2) Random Mislabeling. For samples in the confusion batch, we rule out their *ground truth labels* during random labeling. This can make the model's fitting on clean samples even worse than a random classifier, and thus helps to reduce the false positive rate (FPR) of poison detection below $1/C$.

3) Using unlabeled D_{reserve}. Since the ground truth labels of D_{reserve} are required for implementing the random mislabeling, we initially assume D_{reserve} is labeled in Sec § 4.1. In practice, we remove this assumption by utilizing the fact that the backdoored model $\widetilde{\theta} := \mathcal{A}(\widetilde{D})$ routinely trained on the poisoned dataset already has a sound clean accuracy in principle. Thus, we directly use the pseudo labels generated by $\widetilde{\theta}$ in our implementations. The rationale behind this design is that we do not need D_{reserve} to be perfectly labeled in our use case. We deem this as another advantage of our method over some prior arts [19, 47] that require accurately labeled clean samples for backdoor defense.

Iterative Poison Distillation. In our practical implementation, we iteratively run confusion training (Line 1-8) for multiple rounds. In principle, after one round of confusion training, most clean samples in D_{clean} will have high loss values due to the failure of fitting while backdoor poison samples in D_{poison} that are correctly fitted by θ_{ct} will still concentrate in a region of lower loss values. Thus, after each round of confusion training, we filter out a ratio of samples with the highest loss values and only use the left part for the next round of confusion training. This is analogous to a physical distillation process. By iteratively removing impurities (clean samples) separated by the previous round of distillation, the density of the desired substances (backdoor poison samples) will be increasingly higher. Such a process will thus keep amplifying the backdoor correlations (higher poison rate) meanwhile weakening the benign correlations (less clean samples) in the distilled poisoned training set. Consequently, the quality of the resulting inference model will also keep improving.

Identify The Target Class. Many state-of-the-art poison detection approaches [4, 13, 47] apply clustering analysis to identify potential target classes and only perform poison detection on training samples that are labeled to these classes. This technique helps to reduce the FPR of the detection because it avoids false positives on obviously innocent classes. In our implementation, we also follow this practice by using a Gaussian Mixture Model similar to Tang et al. [47].

We refer readers to Algorithm 2-3 in Appendix § A.1 for a formal description of these techniques as well as our implementation details.

5 Empirical Evaluation of Confusion Training

In this section, we present the empirical evaluation of the confusion training (CT) defense. We discuss the evaluation setup in Sec § 5.1, followed by Sec § 5.2, presenting our major results and ablation studies on CIFAR10 and GTSRB. Sec § 5.3 demonstrates the scalability and generalizability of CT on two large datasets for image classification and malware detection, namely ImageNet and Ember. We discuss additional adaptive attacks against CT in Sec § 5.4.

5.1 Evaluation Setup

Datasets. Our primary results focus on two commonly studied image classification datasets, CIFAR10 [17] and GTSRB [45]. These datasets are the primary focus of prior works on backdoor attacks and defenses, and their use allows us to perform a thorough comparison with state-of-the-art approaches. We further consider the 1000 classes ImageNet dataset and the Ember dataset for malware classification in Sec § 5.3.

Models. For a consistent evaluation, we use ResNet18 [14] as the default model architecture for our primary experiments. Later in Sec 5.2.3, we present additional ablation results on other architectures (VGG16 [44], MobileNetV2 [40], DenseNet121 [15]).

Metrics. We report our evaluation results with two sets of metrics: **1)** for poison detection defenses, we report the *true positive rate (TPR)* and *false positive rate (FPR)* of the detection (Table 1); **2)** for end-to-end backdoor defenses, we report the *clean accuracy (ACC)* and *attack success rate (ASR)* of the final model protected by the defense. Moreover, to compare all the different types of defenses, we also evaluate the ACC and ASR for poison detection defenses. As noted earlier

Table 1: **Comparison of CT with existing backdoor poison samples detectors.** We report the true positive rate (TPR) and false positive rate (FPR) of the poison detection in the format of "mean (standard deviation)" over three independent experiments.

(%)	mean (std)	SentiNet		STRIP		SS		AC		Frequency		SCAn		SPECTRE		CT (ours)	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
C I F A R 10	No Poison	-	5.0 (0)	-	10.3 (0.2)	-	15.0 (0)	-	3.5 (1.0)	-	0.8 (0)	-	2.4 (0.6)	-	7.5 (0)	-	2.2 (0.6)
	BadNet	100 (0)	5.0 (0)	100 (0)	10.1 (0.7)	100 (0)	4.2 (0)	100 (0)	2.5 (0.1)	100 (0)	0.8 (0)	100 (0)	1.5 (1.1)	100 (0)	2.0 (0)	100 (0)	1.0 (0.8)
	Blend	2.2 (0.8)	5.0 (0)	14.0 (6.3)	10.0 (0.6)	91.8 (7.4)	4.2 (0)	65.6 (46.4)	3.6 (1.6)	90.7 (0.8)	0.8 (0)	93.1 (0.6)	0 (0)	100 (0)	2.0 (0)	100 (0)	1.6 (0.8)
	Trojan	100 (0)	5.0 (0)	100 (0)	10.6 (0.5)	18.0 (10.7)	4.5 (0.1)	0 (0)	3.5 (3.9)	100 (0)	0.8 (0)	100 (0)	1.9 (2.0)	100 (0)	2.0 (0)	100 (0)	1.8 (0.5)
	CL	2.0 (2.9)	5.0 (0)	100 (0)	10.3 (0.3)	58.7 (33.8)	4.3 (0.1)	0 (0)	5.6 (2.7)	100 (0)	0.8 (0)	100 (0)	1.9 (1.6)	100 (0)	2.0 (0)	100 (0)	0.7 (0.4)
	SIG	34.0 (57.1)	5.0 (0)	97.5 (1.5)	9.5 (0.2)	99.0 (0.8)	28.6 (0)	99.5 (0.6)	2.6 (2.0)	41.0 (0)	0.7 (0)	98.0 (0.9)	0 (0)	99.0 (0.7)	13.3 (0)	100 (0)	0.3 (0.2)
	Dynamic	100 (0)	5.0 (0)	99.8 (0.3)	10.0 (0.3)	68.0 (21.5)	4.3 (0.1)	0 (0)	4.8 (1.3)	99.3 (0)	0.8 (0)	96.0 (1.1)	0 (0)	100 (0)	2.0 (0)	99.8 (0.3)	1.4 (0.6)
	ISSBA	9.7 (6.7)	5.0 (0)	67.4 (45.8)	10.0 (0.1)	94.7 (7.4)	28.7 (0.1)	93.1 (9.7)	3.1 (2.6)	80.3 (0)	0.8 (0)	92.4 (10.5)	0 (0)	92.8 (10.1)	9.3 (5.6)	100 (0)	0.7 (0.5)
	WaNet	2.3 (0.5)	5.0 (0)	4.3 (0.2)	9.7 (0.8)	87.2 (0.7)	48.0 (0.1)	94.8 (0.3)	3.8 (1.5)	6.1 (0)	10.8 (0)	87.7 (2.8)	0 (0)	88.7 (3.4)	22.7 (0.2)	96.5 (0.6)	0.3 (0.2)
	TaCT	100 (0)	5.0 (0)	50.7 (6.8)	9.8 (0.2)	25.3 (28.3)	4.4 (0.1)	0 (0)	4.5 (2.8)	100 (0)	1.1 (0)	0 (0)	1.6 (1.2)	100 (0)	2.0 (0)	100 (0)	1.5 (1.1)
	Adap-Patch	94.4 (2.8)	5.0 (0)	0.5 (0.6)	10.4 (0.2)	1.1 (0.6)	4.5 (0)	0 (0)	2.8 (0.2)	0 (0)	1.2 (0)	0 (0)	2.0 (0.4)	77.8 (0.8)	2.0 (0)	96.0 (2.4)	2.8 (0.4)
	Adap-Blend	1.4 (1.2)	5.0 (0)	0.9 (1.3)	9.9 (0.7)	51.6 (30.4)	4.4 (0.1)	0 (0)	3.6 (1.4)	78.0 (0)	1.0 (0)	0 (0)	1.6 (1.1)	90.2 (2.3)	2.0 (0)	96.2 (4.0)	3.5 (0.7)
G T S R B	No Poison	-	5.0 (0)	-	11.0 (0.4)	-	39.1 (0)	-	7.1 (1.1)	-	34.4 (0)	-	2.1 (1.3)	-	3.1 (0.1)	-	3.3 (1.1)
	BadNet	100 (0)	5.0 (0)	99.7 (0.5)	10.8 (0.6)	98.0 (2.3)	38.3 (0)	97.0 (1.9)	7.1 (1.4)	100 (0)	34.3 (0)	99.0 (1.0)	0.9 (1.2)	98.4 (2.3)	7.1 (0.5)	100 (0)	0 (0)
	Blend	42.2 (20.2)	5.0 (0)	58.4 (39.5)	11.2 (0.8)	93.7 (3.5)	38.4 (0)	95.6 (1.4)	8.9 (0.7)	93.6 (0)	34.3 (0)	95.5 (2.0)	2.3 (0.5)	98.7 (1.0)	8.1 (1.2)	100 (0)	0.4 (0.2)
	Trojan	100 (0)	5.0 (0)	99.9 (0.2)	10.1 (0.7)	97.9 (2.1)	38.3 (0.1)	98.7 (1.5)	5.8 (0.6)	100 (0)	34.3 (0)	100 (0)	0.2 (0.4)	100 (0)	6.5 (0.3)	100 (0)	0.5 (0.2)
	SIG	0.1 (0.1)	5.0 (0)	51.6 (13.0)	11.3 (0.3)	61.2 (1.8)	49.8 (0)	80.2 (7.9)	8.1 (2.0)	63.5 (0)	34.1 (0)	64.3 (11.4)	1.0 (1.4)	39.1 (27.6)	8.5 (1.2)	100 (0)	0.5 (0.1)
	Dynamic	79.7 (9.1)	5.0 (0)	96.9 (3.9)	11.3 (0.3)	91.6 (4.3)	17.8 (0)	86.6 (4.2)	8.1 (2.9)	84.8 (0)	34.3 (0)	78.5 (6.3)	1.0 (1.4)	98.3 (2.4)	3.0 (0)	99.2 (1.2)	1.8 (1.5)
	WaNet	2.3 (0.9)	5.0 (0)	9.5 (2.3)	10.1 (1.1)	54.0 (1.7)	49.7 (0.1)	0 (0)	3.3 (1.2)	33.9 (0)	40.7 (0)	93.1 (6.9)	1.6 (1.4)	0 (0)	10.2 (1.2)	99.6 (0.3)	0.6 (0.4)
	TaCT	22.1 (38.2)	5.0 (0)	35.9 (5.8)	11.6 (0.6)	98.7 (0.3)	25.3 (0)	99.5 (0.4)	9.6 (1.9)	100 (0)	34.7 (0)	100 (0)	2.6 (1.4)	0 (0)	5.5 (0)	98.5 (1.6)	2.4 (0.9)
	Adap-Patch	100 (0)	5.0 (0)	3.0 (1.2)	11.6 (0.2)	64.7 (3.2)	25.4 (0)	0 (0)	4.1 (1.4)	69.2 (0)	34.6 (0)	0 (0)	0.8 (1.1)	0 (0)	3.7 (0.3)	94.2 (2.2)	2.6 (1.2)
	Adap-Blend	1.7 (0.7)	5.0 (0)	2.6 (1.8)	11.1 (0.8)	43.4 (6.3)	17.9 (0)	0 (0)	6.9 (1.3)	79.7 (0)	34.4 (0)	0 (0)	0.8 (1.1)	0 (0)	3.5 (0.1)	98.3 (1.6)	1.6 (0.5)

in Sec § 2.3, the end-to-end performance indicators (ACC and ASR) of poison detection defenses depend on how subsequent models are trained (e.g., algorithms/architectures) on the cleansed dataset. For a fair and straightforward comparison, these two metrics are measured by directly retraining the same model architecture (ResNet18 by default) from scratch on the dataset cleansed by the corresponding detection approach. For rigor of the evaluation, all major experiments are independently repeated three times, and we report the results in the format of "mean (standard deviation)".

Attacks. To validate the generalizability of confusion training, we evaluate it against 11 different types of backdoor poisoning attacks in the literature, including **1)** classical *dirty label* attacks (BadNet [12], Blend [5], Trojan [26]), **2)** *clean label* attacks (CL [52], SIG [3]), **3)** attacks with *sample-specific triggers* (Dynamic [31], ISSBA [21], WaNet [30]), as well as **4)** *latent space adaptive attacks* (TaCT [47], Adap-Blend [36], Adap-Patch [36]). During the implementation of these attacks, we strictly followed the protocols and suggested configurations of their original papers and open-source implementations. On CIFAR10, we implement and evaluate all these attacks. On GTSRB, we omit CL and ISSBA since their original papers only release poison sets or models for CIFAR10. In our main evaluation, *we try our best to pick a minimum **poison rate** (defined as $|D_{\text{poison}}|/|\widetilde{D}|$) for each attack, maximizing its stealthiness while still maintaining a high ASR*. The rationale behind this choice is that a low poison rate is more realistic and always preferable by attackers in practical scenarios, and attacks with lower poison rates are also harder to defend — we choose the setting that is most realistic and challenging. In Sec 5.2.3, we present ablation results on different poison rates for comprehensiveness. In Appendix § D, we describe the detailed configurations of all the considered attacks.

Baseline Defenses. We compare confusion training with 14

backdoor defenses in the literature: **1)** To illustrate the superiority of our proactive mindset, we include the *6 post-hoc poison detection approaches* (SentiNet [7], STRIP [11], SS [50], AC [4], SCAn [47] and SPECTRE [13]) that we analyze in Sec § 3.2. **2)** For comprehensiveness, we also include the frequency based detection method (Frequency [60]) that directly scans input samples without utilizing post-attacked models' characteristics, presenting it as an outlier that does not fit into the paradigm we discuss. **3)** We also include 7 other representative backdoor defenses that are not built on poison detection. Specifically, FP [23], NC [55], ABL [18] and NAD [19] are covered in Table 2, while DBD [16], MOTH [48] and NONE [56] are deferred to Appendix § B. Hyperparameters of these baseline defenses are optimized following their original papers and open-source implementations.

5.2 Effectivness Across Settings: A Benchmark Evaluation on CIFAR10 and GTSRB

We report our major results on CIFAR10 and GTSRB, two standard benchmark datasets in backdoor research. Specifically, we present an overview of the results in 5.2.1, followed by a thorough analysis in 5.2.2 and ablation studies in 5.2.3.

5.2.1 Consistent Effectiveness Across Settings

We present our main results in Table 1 (for poison detection based defenses) and Table 2 (for all defenses). We consider a poison detector is **successful** against an attack if the TPR is more than 80%; otherwise, we say the detector is **unsuccessful**. We consider a defense is **successful** against an attack only if the ASR is reduced below 20%, otherwise **unsuccessful**.

TPR and FPR. According to Table 1, as a backdoor poison samples detector, CT is **successful** in detecting poison sam-

Table 2: **Evaluation of CT's effectiveness as a backdoor defense.** We report the defended models' clean accuracy (ACC) and attack success rate (ASR) in the format of "mean (standard deviation)" over three independent experiments.

(%)	mean (std)		No Defense	SentiNet	STRIP	SS	AC	Frequency	SCAn	SPECTRE	FP	NC	ABL	NAD	CT (ours)
C I F A R 10	No Poison	ACC	94.1 (0.1)	93.4 (0.6)	92.8 (0.1)	91.3 (0.1)	91.6 (0.4)	93.7 (0.2)	92.8 (0.1)	92.0 (0.1)	82.9 (1.1)	93.5 (0.7)	91.6 (2.5)	87.4 (0.6)	92.2 (0.3)
		ASR	-	-	-	-	-	-	-	-	-	-	-	-	-
	BadNet	ACC	93.8 (0.1)	93.2 (0.2)	92.7 (0.2)	92.8 (0.1)	91.7 (0.3)	93.3 (0.4)	92.7 (0.2)	93.2 (0.2)	83.5 (0.4)	93.7 (0.2)	90.9 (0.5)	86.8 (1.3)	93.2 (0.1)
		ASR	100 (0)	0.7 (0.1)	0.8 (0.1)	0.6 (0)	0.8 (0.2)	0.7 (0.1)	0.8 (0.1)	0.8 (0.1)	100 (0)	1.0 (0.7)	7.7 (5.8)	2.7 (1.6)	0.8 (0.2)
	Blend	ACC	93.6 (0.1)	93.1 (0.5)	92.8 (0.3)	92.4 (0.3)	90.7 (2.0)	93.2 (0.3)	93.1 (0.2)	93.2 (0.2)	83.6 (0.1)	93.1 (0.6)	90.5 (0.9)	82.3 (6.9)	93.1 (0.2)
		ASR	93.5 (0.2)	90.7 (1.3)	88.5 (0.3)	7.3 (3.1)	31.2 (41.3)	33.7 (5.4)	7.1 (1.5)	2.4 (0.1)	85.2 (8.6)	63.2 (44.1)	6.9 (5.2)	11.7 (2.7)	1.6 (0.3)
	Trojan	ACC	93.8 (0.1)	93.2 (0.1)	93.0 (0.6)	92.7 (0.1)	91.7 (1.9)	93.3 (0.4)	93.1 (0.7)	93.1 (0.4)	81.1 (2.4)	93.3 (0.2)	92.4 (1.7)	86.4 (0.5)	92.9 (0.1)
		ASR	99.9 (0.1)	1.6 (0.1)	2.4 (1.1)	99.8 (0.1)	1.7 (0.2)	2.3 (0.4)	1.7 (0.5)	1.7 (0.2)	71.6 (41.2)	1.0 (0.5)	3.3 (3.2)	13.0 (2.7)	1.7 (0.4)
	CL	ACC	93.6 (0)	92.8 (0.4)	93.0 (0)	92.7 (0.2)	90.4 (2.7)	93.4 (0.1)	92.8 (0.1)	93.0 (0.1)	82.1 (1.8)	93.5 (0.2)	81.7 (3.0)	86.4 (1.5)	93.5 (0.2)
		ASR	99.8 (0.1)	99.6 (0.6)	1.4 (0.2)	54.2 (39.8)	97.9 (1.7)	1.5 (0.5)	1.2 (0.4)	1.6 (0.2)	94.3 (3.5)	1.0 (0.4)	13.8 (8.7)	12.0 (10.5)	0.9 (0)
	SIG	ACC	93.8 (0)	92.6 (0.6)	92.6 (0.1)	90.1 (0.8)	90.3 (0)	93.3 (0.2)	92.9 (0.2)	89.7 (0.2)	82.8 (1.1)	93.7 (0.4)	87.5 (1.8)	86.3 (0.9)	93.2 (0)
		ASR	80.2 (0.6)	81.2 (0.7)	10.1 (16.7)	0.8 (0.2)	0.1 (0)	67.3 (2.9)	1.1 (0.8)	0.8 (0.2)	56.9 (28.5)	82.7 (0.8)	29.0 (31.9)	2.3 (1.1)	0.1 (0)
	Dynamic	ACC	93.8 (0)	92.8 (0.4)	93.0 (0)	92.6 (0.1)	92.9 (0.9)	93.3 (0.2)	93.4 (0.2)	93.2 (0.1)	82.2 (1.2)	93.3 (0.2)	89.5 (5.0)	84.3 (3.2)	93.2 (0)
		ASR	99.3 (0.4)	4.6 (1.0)	4.9 (0.6)	60.3 (38.2)	98.4 (0.4)	6.8 (1.1)	9.6 (2.5)	4.8 (1.0)	95.0 (7.9)	5.1 (2.3)	95.8 (5.8)	35.1 (23.5)	3.7 (1.6)
	ISSBA	ACC	93.7 (0)	92.3 (0.7)	93.0 (0.2)	90.5 (0.2)	92.5 (0.5)	92.8 (0.4)	93.2 (0.2)	89.8 (0)	83.4 (0.3)	93.5 (0.3)	89.2 (4.9)	86.7 (1.1)	93.2 (0)
		ASR	100 (0)	34.7 (55.7)	33.8 (46.7)	1.2 (0.1)	1.6 (0.2)	0.8 (0.1)	0.8 (0.1)	1.5 (0.2)	0.1 (0.2)	0.5 (0.1)	99.6 (0.6)	2.8 (1.2)	0.6 (0)
	WaNet	ACC	93.0 (0.5)	92.7 (0.2)	92.4 (0.6)	87.4 (0.2)	90.9 (0.6)	92.3 (0.1)	92.7 (0.3)	87.4 (0.1)	82.6 (0.4)	92.5 (0.6)	88.1 (5.9)	86.2 (1.2)	93.0 (0.2)
		ASR	93.9 (0.8)	86.4 (2.1)	81.8 (4.5)	3.6 (0.4)	1.1 (0.2)	86.3 (2.6)	2.0 (0.3)	2.7 (0.9)	1.9 (1.6)	90.8 (5.0)	22.1 (52.6)	1.4 (0.5)	1.0 (0.1)
	TaCT	ACC	93.6 (0)	93.3 (0.5)	93.0 (0.1)	92.5 (0)	91.1 (2.1)	93.2 (0.1)	92.7 (0.2)	93.2 (0)	83.6 (0.3)	92.9 (0.2)	89.3 (4.2)	84.8 (0.6)	92.5 (0.5)
		ASR	99.6 (0.2)	1.8 (0.5)	97.3 (0.4)	98.7 (0.1)	98.4 (0.9)	0.9 (0.5)	98.0 (0.9)	1.0 (0)	96.4 (2.0)	4.0 (3.4)	99.0 (0.9)	7.5 (3.6)	1.2 (0.1)
	Adap-Patch	ACC	93.5 (0.2)	92.7 (0.4)	93.0 (0.1)	92.5 (0)	91.6 (1.7)	93.5 (0.5)	92.5 (0.1)	92.9 (0)	85.3 (3.1)	93.4 (0.3)	86.9 (5.9)	86.5 (1.1)	92.4 (0.1)
		ASR	100 (0)	1.4 (1.0)	98.3 (0.1)	99.8 (0.1)	99.8 (0.1)	98.5 (2.0)	99.6 (0.1)	1.1 (0.5)	99.9 (0.1)	33.8 (56.9)	99.9 (0.1)	11.1 (7.0)	1.4 (0.1)
	Adap-Blend	ACC	94.0 (0)	93.1 (0.5)	93.2 (0.1)	92.8 (0)	91.8 (0.6)	93.2 (0.1)	92.8 (0)	92.7 (0.1)	83.9 (0.6)	93.1 (1.2)	85.7 (4.6)	86.3 (1.2)	92.8 (0.2)
		ASR	84.8 (1.1)	77.5 (4.0)	80.3 (0.6)	41.5 (28.3)	78.7 (2.4)	20.5 (5.3)	81.6 (5.0)	3.9 (0.9)	61.9 (38.6)	82.9 (5.2)	95.1 (3.9)	7.5 (2.4)	2.2 (0.1)
G T S R B	No Poison	ACC	97.0 (0.4)	96.5 (0.5)	96.2 (0.1)	94.8 (0.1)	95.3 (0.1)	95.7 (0.6)	96.5 (0.2)	95.9 (0.2)	86.1 (1.8)	96.2 (1.7)	84.0 (6.6)	96.2 (0.2)	96.3 (0.1)
		ASR	-	-	-	-	-	-	-	-	-	-	-	-	-
	BadNet	ACC	96.7 (0.1)	96.6 (0.5)	96.3 (0.1)	94.7 (0.4)	95.7 (0.5)	95.5 (0.4)	96.2 (0.2)	95.4 (0.3)	87.2 (0.6)	96.9 (0.5)	86.0 (8.5)	96.5 (0.3)	96.4 (0.1)
		ASR	100 (0)	0.1 (0.1)	0.2 (0)	0.2 (0)	0.2 (0)	0.2 (0.1)	0.2 (0)	0.2 (0)	37.0 (45.0)	0.9 (0.3)	12.6 (17.0)	0.8 (0.5)	0.2 (0)
	Blend	ACC	96.5 (0)	96.3 (0.3)	95.9 (0.3)	94.2 (0.4)	94.9 (0.5)	95.3 (0.2)	96.2 (0.2)	95.3 (0.1)	86.7 (0.7)	96.4 (0.4)	90.6 (1.8)	96.2 (0.5)	96.2 (0)
		ASR	98.2 (0.1)	97.6 (0.3)	89.9 (2.4)	72.9 (13.1)	71.8 (3.9)	75.4 (2.7)	37.0 (18.0)	14.6 (11.7)	97.1 (0.9)	27.9 (3.8)	14.6 (3.9)	41.5 (13.1)	1.4 (0.8)
	Trojan	ACC	97.1 (0.2)	96.8 (0.1)	96.6 (0.2)	95.1 (0.1)	95.7 (0.2)	95.5 (0.4)	96.7 (0.5)	96.0 (0.3)	87.1 (0.2)	96.8 (0.3)	92.9 (0.8)	96.2 (0.2)	96.8 (0.1)
		ASR	100 (0)	0.2 (0)	0.1 (0.1)	0.6 (0.2)	0.3 (0.2)	0.2 (0.1)	0.2 (0)	0.1 (0)	98.3 (1.1)	1.5 (0.7)	6.5 (4.4)	0.7 (0.1)	0.2 (0.1)
	SIG	ACC	96.8 (0.3)	96.5 (0.3)	96.0 (0)	93.2 (0.4)	95.0 (0.2)	95.5 (0.3)	96.3 (0.2)	95.2 (0)	85.9 (1.7)	97.2 (0.1)	76.2 (19.8)	96.3 (0.3)	96.4 (0)
		ASR	52.9 (2.1)	49.6 (4.5)	49.7 (0.1)	49.6 (4.9)	8.5 (6.8)	39.2 (1.6)	26.5 (5.9)	35.2 (12.3)	58.6 (5.9)	50.5 (4.7)	57.5 (49.2)	12.4 (3.4)	0.4 (0.2)
	Dynamic	ACC	97.0 (0.2)	96.7 (0.3)	96.1 (0)	95.9 (0)	95.3 (0.1)	95.6 (0.3)	96.5 (0.3)	95.9 (0.2)	87.0 (0.2)	96.3 (1.0)	87.0 (5.6)	96.7 (0.5)	96.4 (0.2)
		ASR	100 (0)	30.7 (47.3)	0.6 (0)	2.5 (2.0)	8.6 (1.0)	14.1 (5.7)	10.7 (3.2)	1.1 (0.2)	100 (0)	30.3 (16.9)	68.5 (54.6)	55.0 (48.7)	0.3 (0.1)
	WaNet	ACC	92.9 (5.7)	95.6 (0.2)	95.3 (0.4)	93.5 (0.3)	95.3 (0.1)	94.0 (0.4)	96.4 (0.8)	93.4 (0.8)	85.5 (0.8)	96.7 (0.4)	78.1 (7.6)	96.9 (0.2)	96.6 (0.1)
		ASR	70.0 (3.2)	74.0 (2.6)	70.6 (6.1)	7.1 (0.6)	76.5 (1.9)	52.0 (3.3)	0.7 (0.8)	75.9 (3.0)	6.6 (5.2)	28.8 (37.7)	89.4 (13.8)	0.3 (0.1)	0.2 (0.1)
	TaCT	ACC	96.9 (0.1)	96.3 (0.3)	95.9 (0.1)	95.7 (0.2)	94.8 (0.2)	95.6 (0.2)	96.3 (0.1)	95.8 (0.1)	86.5 (0.8)	96.7 (0.6)	92.3 (5.6)	96.3 (0.3)	96.6 (0.4)
		ASR	99.8 (0.1)	99.7 (0.5)	99.8 (0)	0.2 (0)	0.2 (0)	0.5 (0.3)	0.1 (0.1)	100 (0)	99.9 (0.2)	88.8 (19.4)	68.5 (54.6)	(5.0)	1.9 (0.8)
	Adap-Patch	ACC	96.6 (0.1)	96.6 (0.5)	96.2 (0.2)	95.3 (0.3)	95.8 (0)	95.5 (0.2)	96.5 (0.2)	95.8 (0.4)	86.4 (0.4)	96.7 (0.6)	67.0 (40.1)	96.4 (0.1)	96.3 (0.1)
		ASR	53.4 (2.3)	0.2 (0.1)	48.1 (1.4)	10.2 (5.1)	42.8 (5.9)	12.2 (2.6)	52.5 (4.3)	52.8 (2.2)	24.0 (20.0)	27.8 (8.5)	74.7 (24.8)	2.1 (0.8)	0.2 (0)
	Adap-Blend	ACC	96.7 (0.2)	96.6 (0.5)	96.5 (0.1)	95.5 (0.1)	94.9 (0.1)	95.7 (0.1)	96.3 (0.2)	96.1 (0.4)	86.7 (0.4)	96.0 (0.2)	77.5 (9.7)	96.3 (0.3)	96.0 (0.1)
		ASR	93.9 (0.8)	91.2 (2.9)	90.1 (0.5)	74.3 (5.0)	91.4 (0.6)	55.2 (3.6)	92.7 (2.1)	93.4 (2.4)	84.7 (3.4)	42.7 (14.0)	83.9 (17.9)	26.1 (12.2)	0.3 (0.1)

ples across all attacks and datasets we evaluate. In all settings, CT consistently detects over 90% poison samples (oftentimes detects 100%). In terms of false positives, CT also achieves a low FPR comparable to the best of other detectors.

ASR and ACC. According to Table 2, as a general backdoor defense, CT also **succeeds** in reducing the ASR in all settings. As shown, CT consistently reduces the ASR to less than 5%. On the other hand, the ACC drop induced by CT defense is moderate – the ACC is always higher than 92.4% on CIFAR10 and 96.0% on GTSRB, in parallel with the best of other defenses. Note that, for a fair comparison, we use the same ResNet18 architecture to retrain models on the cleansed dataset to report ASR and ACC. *In practice, users can freely train models with more advanced architecture and larger capacity on the cleansed dataset to achieve higher accuracy.*

5.2.2 Strengths of Confusion Training over Prior Arts

Prior arts are less effective against latent space adaptive attacks. The three latent space adaptive attacks (TaCT [47], Adap-Patch and Adap-Blend [36]) constitute the most chal-

lenging cases in our evaluation. In these adaptive attacks, not all trigger-planted samples are labeled to the target class in the poisoned dataset. Thus, the backdoor correlations between the trigger and target class are intentionally complicated, the trigger signal becomes less dominant in the prediction of post-attacked models, and the latent separation between poison and clean samples is also suppressed [36]. As shown in Table 2, *all of the 11 baseline defenses fail to defend against at least one of the adaptive attacks on at least one dataset*. **First**, the suppression of latent separation characteristic makes the four post-hoc poison detection methods (SS [50], AC [4], SCAn [47], SPECTRE [13]) built on this characteristic less effective, as seen in Table 1. **Second**, complicated backdoor correlations make it harder to fit the backdoor poison samples, reducing the effectiveness of ABL [18], which assumes poison samples will be fitted faster than clean samples. **Third**, as the backdoor trigger becomes less dominant, approaches like SentiNet [7], Strip [11], and NC [55] that rely on the dominance of backdoor triggers are also less effective. **Additionally**, Frequency [60] suffers from degradation against Adap-Blend

and Adap-Patch because these attacks use more implicit triggers. NAD [19] also performs worse against Adap-Blend on GTSRB, which could be due to the weakened distinguishability between clean and poison populations in latent space, on which NAD performs knowledge distillation. FP [23] in general performs poorly due to its ineffectiveness in our evaluation settings, where poison rates of all attacks are low.

In contrast, CT demonstrates stronger robustness against such adaptive attacks. As shown in Table 1, CT still consistently achieves over 90% TPR against all of the three latent space adaptive attacks on both datasets. This suffices to defend against these attacks, as ASRs on all the models trained on the cleansed datasets are reduced to less than 4%.

This feat can be attributed to the proactive mindset underlying the design of CT, which steers clear of the reliance on those post-hoc characteristics used by prior arts. Specifically, CT makes no assumption about the type of triggers or labeling of backdoor poison samples. It neither passively waits for backdoored models to "magically" exhibit some characteristics that will expose backdoor poison samples. Instead, *CT starts from an arguably more fundamental premise that the successful execution of a backdoor poisoning attack must be accompanied by the existence of backdoor correlations and corresponding backdoor poison samples (deviating from the clean distribution) that can be fitted*. Overall, the CT design aims to fit backdoor poison samples (that we don't have knowledge of) while avoiding fitting samples from the clean distribution (that we can approximate) by proactively making them.

This design also makes CT intrinsically more robust on other dimensions. Based on our evaluation results in Table 1,2, we summarize them as follows.

1) CT is effective against clean-label attacks (CL [52], SIG [3]). Clean label attacks are designed to evade human inspection by avoiding mislabeling any poison samples. They do not impose any challenges on CT as this strategy can not stop CT from fitting the backdoor correlations.

2) CT is effective against sample-specific triggers (Dynamic [31], WaNet [30], ISSBA [21]). Sample-specific attacks use different triggers for different backdoor samples. These backdoor attacks with such diversed triggers lead to the reduced effectiveness of many defenses, including SentiNet, STRIP, Frequency, SPECTRE, NC, ABL, and NAD. Meanwhile, CT still defends against these attacks effectively.

3) CT is effective against implicit triggers (Blend [5], SIG [3], WaNet [30], ISSBA [21]). There are some attacks that use more implicit triggers for stealthiness. Unlike other attacks that patch a significant trigger over images, these defenses manage to reduce the strength of the trigger signals, making them less noticeable. According to our results, no other defenses in our evaluation can steadily succeed against all these attacks on both datasets. However, CT is consistently successful in inhibiting the ASR to $< 1.6\%$.

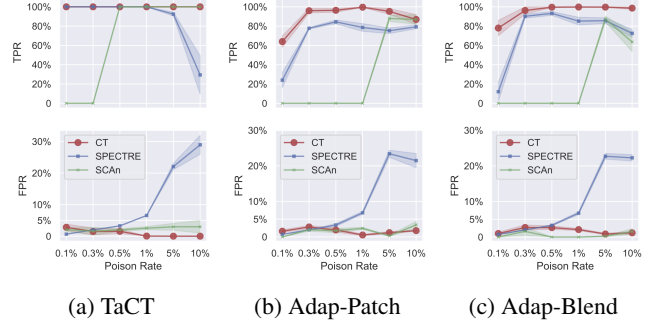*4) CT is effective across datasets.* Another significant is-

Figure 3: **Ablation experiments on different poison rates.** Refer to Appendix 6 for ablation results on more attacks.

sue with the baseline defenses is that they do not generalize equally across datasets. A notable example is SPECTRE, a post-hoc approach based on latent separation, which performs well on CIFAR10 against all attacks but fails in multiple cases on GTSRB. This is a direct result of the post-hoc workflow reviewed in Sec § 3.2, as we previously demonstrated in Fig 2c that the latent separation characteristics of the post-attacked model could vary across datasets if they are not proactively controlled. As a result, post-hoc approaches are prone to failure. In contrast, the proactive nature of CT makes itself more stable across datasets. The effectiveness of CT on additional datasets will be discussed in Sec § 5.3.

5.2.3 Ablation Studies

Next, we present additional ablation studies on three factors: **1)** poison rates of attacks, **2)** model architectures used for CT, and **3)** size of the reserved clean set for CT. We investigate the impacts of these factors on the effectiveness of CT. For the ablation study on each factor, we only vary this single factor while keeping all other factors consistent with the setup of the main evaluation in Sec § 5.1. All ablation studies are performed on CIFAR10.

Poison Rates. Fig 3 presents our ablation study on the poison rate of different attacks (we only demonstrate the three most challenging attacks here; refer to Appendix § C for more results). Specifically, we consider the set of poison rates $\{0.1\%, 0.3\%, 0.5\%, 1\%, 5\%, 10\%\}$ which covers both the high and low poison rate regions. For each setting, we compare CT with the two strongest baseline detectors SPECTRE and SCAn. *CT is consistently effective (TPR≈100%) and outperforms the baselines in both low and high poison rates regions against most attacks (see Fig 6 in Appendix).* Even for three attacks in Fig 3 on which most other defenses fail, CT still achieves TPR≈100% at most time, beyond the reach of the two baselines. Though the TPR of CT slightly drops in the ultra-low poison rate of 0.1% against Adap-Patch and Adap-Blend, it is still noticeably better than the baselines.
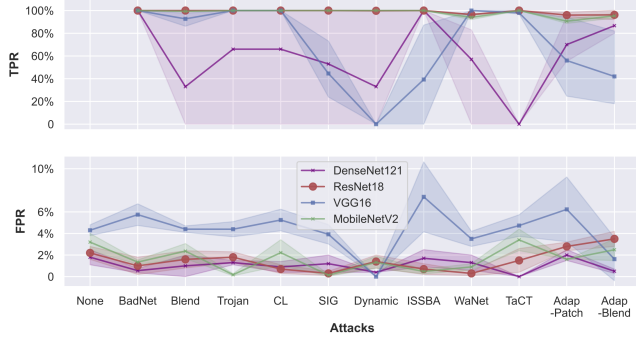
Model Architectures. In addition to ResNet18 [14], which

Figure 4: **Ablation results on different model architectures.**

Figure 5: **Ablation results on size of the reserved clean set.**

is used in the main evaluation, we also consider three other model architectures, VGG16 [44], MobileNetV2 [40], and DenseNet121 [15]. For a fair comparison, the hyperparameters are optimized for each architecture. The ablation results on the model architectures are shown in Fig 4. *The key takeaway is that some architectures are more suitable for implementing CT than others.* As shown, CT with ResNet18 and MobileNetV2 are stable across different attacks, achieving high TPR (close to 100% in most cases) while maintaining an FPR below 5%. In contrast, VGG16 and DenseNet121 are less stable, with large variations in performance and frequent failures. According to our empirical experiences, this is because the dynamics of confusion training require easy-to-train architectures. VGG16 does not use skip links, while DenseNet121 is too deep, and thus they are more difficult to train and less stable for CT. This observation is consistent with our practical insights in Sec § 3.3 and sheds light on a future research direction where specialized model architectures can be designed to facilitate poison detection.

Size of The Reserved Clean Set. CT requires a small reserved clean set to launch. Our default implementation in the main evaluation uses a clean set of 2000 samples. We now investigate how the effectiveness of CT varies when the size of the reserved clean set becomes smaller. Specifically,

we experiment with fewer clean samples of $\{1000, 500, 250\}$ respectively and present the results (TPR and FPR) of the position detection in Fig 5. Overall, the TPR of our CT detection pipeline maintains a relatively robust performance, though the TPR would become worse against the two latent space adaptive attacks (Adap-Patch and Adap-Blend). However, a noticeable trade-off in the FPR is evident. Remarkably, when the quantity of reserved clean samples is reduced to 250, the FPR can, at times, exceed 20%, leading to a substantial loss of training data. Intuitively, this is due to the fact that a smaller clean set approximates the clean distribution worse. Thus the confusion batch is also doing worse in decoupling the benign correlations — many clean samples could still be fitted, leading to higher FPR. The impact of the high FPR on overall defense performance (ACC and ASR) is also evaluated, focusing on the extreme case where only 250 clean samples are used to bootstrap CT for CIFAR10. Even with over 20% of the CIFAR10 training set discarded, CT still achieved an impressive accuracy exceeding 91% in all cases, thanks to the maintained high TPR, which ensured a low Attack Success Rate (ASR). This suggests that for larger datasets, a high FPR might be tolerable during dataset cleansing, as the remaining data can still be used to train a high-quality model. A closely relevant research topic to this observation is Dataset Pruning [32], which shows that many training data can be removed from the dataset while models trained on the remaining dataset can still be accurate.

Table 3: **CT on CIFAR10 with 250 clean samples.**

(%) mean (std)	No Defense		CT (ours)			
	ACC	ASR	TPR	FPR	ACC	ASR
No Poison	94.1 (0.1)	-	-	12.5 (1.8)	92.1 (0.2)	-
BadNet	93.8 (0.1)	100 (0)	100 (0)	14.4 (1.6)	92.1 (0.1)	1.0 (0.2)
Blend	93.6 (0.1)	93.5 (0.2)	100 (0)	19.3 (1.3)	91.9 (0.1)	3.2 (1.3)
Trojan	93.8 (0.1)	99.9 (0.1)	100 (0)	11.0 (1.7)	92.0 (0.2)	1.8 (0.3)
CL	93.6 (0)	99.8 (0.1)	100 (0)	13.7 (2.3)	92.1 (0.3)	1.4 (0.4)
SIG	93.8 (0)	80.2 (0.6)	100 (0)	2.0 (1.1)	92.6 (0.2)	0 (0)
Dynamic	93.8 (0)	99.3 (0.4)	96.2 (3.1)	15.8 (1.0)	92.0 (0.2)	5.2 (2.1)
ISSBA	93.7 (0)	100 (0)	99.8 (0)	5.4 (2.0)	92.7 (0.1)	0.8 (0)
WaNet	93.0 (0.5)	93.9 (0.8)	95.6 (2.1)	10.8 (2.0)	92.5 (0.3)	0.9 (0.2)
TaCT	93.6 (0)	99.6 (0.2)	100 (0)	16.8 (1.2)	92.1 (0.1)	1.0 (0.3)
Adap-Patch	93.5 (0.2)	100 (0)	84.3 (5.3)	17.2 (1.8)	91.9 (0.1)	0.6 (0.1)
Adap-Blend	94.0 (0)	84.8 (1.1)	92.7 (4.7)	21.4 (2.3)	91.7 (0.2)	2.1 (0.2)

5.3 Scalability and Generalizability

Generalization to Domain Other Than Computer Vision. CT only assumes backdoor poison samples could be fitted after the benign correlations have been decoupled, without constraining the modality it works on. This suggests the potential generalizability of CT to different domains other than computer vision. To illustrate this, we evaluate CT on the Ember [1] training dataset, a malware classification dataset that consists of 600K feature vectors extracted from Portable Executable (PE) files. Specifically, we consider the attacks proposed by Severi et al. [42]. We use the same EmberNN architecture used in Severi et al. [42] to build models and also follow the same configurations. We consider both the con-

Table 4: **Defense results of CT on Ember.**

(%) mean (std)	No Defense		CT (ours)			
	ACC	ASR	TPR	FPR	ACC	ASR
No Poison	99.1 (0)	-	-	4.0 (0)	98.7 (0.1)	-
Unconstrained	99.1 (0.1)	84.0 (3.86)	99.9 (0.0)	3.0 (0)	98.8 (0)	0 (0)
Constrained	99.0 (0.1)	62.1 (3.7)	99.8 (0.1)	3.0 (0)	98.8 (0)	0 (0)

Table 5: **Defense results of CT on ImageNet-1k.**

(%)	mean (std)	No Poison	BadNet	Trojan	Blend
No Defense	ACC	67.5 (0.1)	67.1 (0.1)	67.5 (0.2)	67.6 (0.1)
	ASR	-	100 (0)	98.9 (0.4)	93.7 (3.8)
CT (ours)	TPR	-	99.5 (0.1)	100 (0)	100 (0)
	FPR	0.1 (0)	0.1 (0)	0.1 (0)	0.1 (0)
	ACC	67.5 (0.1)	67.6 (0.1)	67.4 (0.1)	67.5 (0.1)
	ASR	-	0 (0)	0.1 (0)	0 (0)

strained and unconstrained attacks of Severi et al. [42] with 1% poison arte. The constrained attack confines backdoor triggers to those features that are editable in PE files, while the unconstrained attack removes this constraint. Table 4 presents our results, validating the effectiveness of CT.

Scale to Realistic Settings. To study the scalability to more realistic settings we will encounter in the real world, we also evaluate CT on the ImageNet-1k training set, consisting of 1.3M images (26x larger than CIFAR10) with high resolutions. We adapt BadNet [12], Trojan [26], and Blend [5] with a 1% poison rate to implement attacks on ImageNet. We use ResNet18 [14] for both confusion training on the poisoned dataset and subsequent retraining on the cleansed dataset. Table 5 presents the results of CT defense against the attacks. As shown, CT still remains effective on ImageNet — with nearly perfect $TPR > 99\%$ and negligible **FPR $\approx$ 0.1**, models trained on cleansed sets reduce *ASR* to 0 without suffering any drop of ACC. An intriguing observation is that the FPR here is even lower than we get for CIFAR10 and GTSRB. We hypothesize that this is because when the size of the dataset is larger and the clean task is more complicated, it also becomes harder for the inference model to fit (memorize) clean samples under the disruption of confusion training, making it even easier to separate clean and poison samples under CT.

Additionally, as previously noted in Sec § 2.3, after we use ResNet18 to cleanse the dataset, we have the flexibility to use any model architecture and algorithm for subsequent training on the cleansed dataset. We confirm this **by replacing ResNet18 (11 M parameters) with ViT-B/16 [9] (150 M parameters) and applying the state-of-the-art training algorithm GSAM [63]** during the retraining phase for cases we evaluate in Table 5. As shown in Table 6, we consistently get clean models with 0 ASR, but the ACC increases to 80% rather than 67% in Table 5. Symmetrically, this also indicates that, in realistic scenarios involving training large models, it is possible to initially deploy CT with a lightweight inference model to sanitize the dataset. The cost would be acceptable, as the computation demanded to train a lightweight model for dataset cleansing can be negligible compared with subsequent training of large models on the cleansed dataset.

5.4 Additional Analysis on Adaptive Attacks

Enforce dependency between benign and backdoor correlations. CT makes very few assumptions about the configurations of backdoor poisoning attacks, but *it does assume that the backdoor correlations can still be fitted after the benign*

Table 6: **Training ViT-B/16 on poisoned ImageNet-1k cleansed by CT:** after CT cleanses the dataset following the same setup in Table 5, we train ViT-B/16 (as opposed to ResNet18) with the state-of-the-art GSAM algorithm [63] on the cleansed dataset and report the new ACC and ASR. (We do one run of each experiment due to the heavy computational overhead; thus, we omit the standard deviation.)

(%)	No Poison	BadNet	Trojan	Blend
ACC	80.7	80.8	80.6	80.7
ASR	-	0	0	0

correlations have been decoupled. Then, a principled class of adaptive attacks against CT can be those attacks where the backdoor correlations depend on benign correlations, making it more difficult to fit backdoor poison samples after the benign correlations have been decoupled. Some attacks that we evaluate in Table 1,2 already fall in this class, including **1)** sample-specific attacks where triggers depend on clean semantics of the samples and **2)** latent space adaptive attacks where trigger planted samples are conditionally labeled to the target class based on the semantic class (e.g. TaCT [47]). Results in Table 1,2, and Fig 6 illustrate that CT still exhibits good robustness. Additionally, we add an evaluation on an all-to-all attack [12] with 1% poison rate on CIFAR10, where trigger-planted samples from class i should be misclassified to class $i+1$. This clearly imposes a dependency between the backdoor and clean tasks. Against this attack, CT still achieves 86.6% TPR and 2.2% FPR, and retraining a ResNet18 on the cleansed dataset results in 5.7% ASR and 93.3% ACC. We hypothesize that this is because backdoor poison samples are always out of the distribution of clean data and thus can still be fitted though the fitting on clean distribution is disrupted.

Launch attack without artifacts. Most existing attacks unavoidably introduce artifacts into poison samples as they inject trigger patterns that significantly deviate from the clean distribution. These artifacts make poison samples easy to be fitted by CT. This explains why CT is consistently effective against different attacks. Thus, another angle to adaptively attack CT is to launch attacks without significant artifacts. One candidate is the rotation attack [57], which uses image rotation as the backdoor trigger without introducing additional artifacts. We implement this attack following the original paper's configuration with 1% poison rate. Still, CT can achieve

83.6% TPR and 2.8% FPR, and retraining a ResNet18 on the cleansed dataset will have 7.2% ASR and 92.9% ACC.

6 Discussion

In Sec § 5, we illustrate nice properties of confusion training (CT) defense. In particular, we highlight that CT is built on **an arguably more fundamental premise** that the successful execution of a backdoor poisoning attack must be accompanied by the existence of backdoor correlations and corresponding backdoor poison samples that are fittable. The design of CT makes use of this fundamental premise, by simply attempting to fit those backdoor poison samples (that we don't have knowledge of, except knowing that they should be fittable), while simultaneously preventing the fitting of samples from the clean distribution (that we can approximate) by proactively making them less fitable. Even though, **we keep a conservative attitude so as not to oversell the security of the confusion training technique itself**. After all, it is an empirical defense without a certified guarantee, and the insights we discuss in Sec § 5.4 may also motivate stronger adaptive attacks that can further challenge CT. Rather, **we position CT as evidence to showcase how our proposal of the proactive mindset may inspire future advancement in backdoor poison samples detection**. In particular, we hope our formulation and practical insights provided in Sec § 3.3 can inspire the design of better proactive backdoor defenses.

7 Related Work

7.1 Backdoor Attacks on Neural Networks

The focus of this study is to address the threat of *backdoor poisoning attacks* on deep neural networks (DNNs), which constitute the most prevalent form of DNN backdoor attacks. These attacks [3, 5, 12, 21, 21, 26, 27, 30, 31, 36, 41, 47, 52, 57] involve the manipulation of a few samples in the training dataset by an attacker, resulting in the "poisoning" of the dataset. Victims will then train their own model on this poisoned dataset. Backdoor attacks also include various methods that do not fit within the paradigm of data poisoning. Examples include modifying the training process [2, 46] or using backdoored pre-trained models [43, 59] for transfer learning, as well as attacks that occur at the deployment stage [25, 34, 35, 38]. These attacks involve different assumptions and threat models, which are out of the scope of this work.

7.2 Backdoor Defenses

Backdoor Poison Samples Detection. This work investigates methods for detecting poison samples in the poisoned dataset as a means of defense. In the existing literature, state-of-the-art approaches [4, 13, 47, 50] of this category consistently build their detectors upon latent separation characteristics, assuming that backdoored models will learn separate latent representations for poison and clean samples. Poison samples can then be identified via cluster analysis in the latent representation space. Tran et al. [50] project the latent representations to the top PCA [33] direction and observe a bimodal distribution — poison and clean samples form their own modal, respectively. Later, Chen et al. [4] independently observe similar separation and propose to use K-means [29] to separate the clean and poison clusters in the latent space. More recently, Tang et al. [47] and Hayase et al. [13] propose to use statistics of the clean distribution to further improve the cluster analysis. Researchers have also proposed using other characteristics such as the entropy of predictions under intentional perturbation [11], abnormal regions in the backdoored model's saliency map [7], and the speed of model fiting [18] to identify poison samples. Zeng et al. [60] propose to detect artifacts of poison samples in the frequency domain.

Other Backdoor Defenses. There are some other defenses, working on different stages. Neural Cleanse [55] proposes to reverse engineer backdoor triggers. Fine-pruning [23] suggests eliminating the model backdoor via pruning. Du et al. [10] and Li et al. [18] apply intervention during model training to suppress the influence of poison samples. Xu et al. [58] directly train meta classifiers to predict whether a model is backdoored. Liu et al. [28] and Li et al. [19] attempt to eliminate the model backdoor via finetuning the model on a small clean dataset. Udeshi et al. [53] introduce input preprocessing to suppress the effectiveness of backdoor triggers during the inference stage. There are also some certified defenses [54] that adapt randomized smoothing for the settings of backdoor attacks and achieve nontrivial certified accuracy against backdoor attacks with small ℓ_p bounded triggers.

8 Conclusion

In this paper, we present a comprehensive investigation into methods for detecting backdoor poison samples in an effort to defend against backdoor poisoning attacks. We uncover a post-hoc workflow underlying many prior works and reveal its limitations. Accordingly, we suggest a paradigm shift by promoting a proactive mindset formulated within a unified framework. We also provide practical insights for grounded implementations. Particularly, we introduce the technique of confusion training (CT) as a concrete instantiation of the proactive mindset. Our empirical evaluations show that CT is effective across different attack settings, outperforming prior ars. We also show that CT is scalable to large datasets and generalizable to the domain of malware detection other than computer vision. We then provide insights for adaptively attacking CT and find CT is quite robust against a number of adaptive attacks. We position CT as evidence to showcase how our proposal of the proactive mindset may inspire future advancement in backdoor poison samples detection.

Acknowledgements

References

[1] Hyrum S Anderson and Phil Roth. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637*, 2018.

[2] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1505–1521, 2021.

[3] 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.

[4] 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.

[5] 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.

[6] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International Conference on Machine Learning*, pages 1964–1974. PMLR, 2021.

[7] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attack against deep learning systems. *IEEE SPW 2020*, 2020.

[8] 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.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[10] Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. *arXiv preprint arXiv:1911.07116*, 2019.

[11] 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.

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

[13] 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.

[14] 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.

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[16] Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. *arXiv preprint arXiv:2202.03423*, 2022.

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

[18] 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.

[19] 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.

[20] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[21] 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.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[23] 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.

[24] Xuankai Liu, Fengting Li, Bihan Wen, and Qi Li. Removing backdoor-based watermarks in neural networks with limited data. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10149–10156. IEEE, 2021.

[25] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138. IEEE, 2017.

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

[27] 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.

[28] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 45–48. IEEE, 2017.

[29] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[30] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. *arXiv preprint arXiv:2102.10369*, 2021.

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

[32] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607, 2021.

[33] 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.

[34] Xiangyu Qi, Jifeng Zhu, Chulin Xie, and Yong Yang. Subnet replacement: Deployment-stage backdoor attack against deep neural networks in gray-box setting. *arXiv preprint arXiv:2107.07240*, 2021.

[35] Xiangyu Qi, Tinghao Xie, Ruizhe Pan, Jifeng Zhu, Yong Yang, and Kai Bu. Towards practical deployment-stage backdoor attack on deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13347–13357, 2022.

[36] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. Revisiting the assumption of latent separability for backdoor defenses. In *International Conference on Learning Representations*, 2023.

[37] Xiangyu Qi, Tinghao Xie, Jiachen T. Wang, Tong Wu, Saeed Mahloujifar, and Prateek Mittal. Towards a proactive ml approach for detecting backdoor poison samples. *arXiv preprint arXiv:2205.13616*, 2023.

[38] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13198–13207, 2020.

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[41] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. Exploring backdoor poisoning attacks against malware classifiers. *arXiv preprint arXiv:2003.01031*, 2020.

[42] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. {Explanation-Guided} backdoor poisoning attacks against malware classifiers. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1487–1504, 2021.

[43] Lujia Shen, Shouling Ji, Xuhong Zhang, Jinfeng Li, Jing Chen, Jie Shi, Chengfang Fang, Jianwei Yin, and Ting Wang. Backdoor pre-trained models can transfer to all. *arXiv preprint arXiv:2111.00197*, 2021.

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

[45] 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.

[46] Te Juin Lester Tan and Reza Shokri. Bypassing backdoor detection algorithms in deep learning. *arXiv preprint arXiv:1905.13409*, 2019.

[47] 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.

[48] Guanhong Tao, Yingqi Liu, Guangyu Shen, Qiuling Xu, Shengwei An, Zhuo Zhang, and Xiangyu Zhang. Model orthogonalization: Class distance hardening in neural networks for better security. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1372–1389. IEEE, 2022.

[49] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[50] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, pages 8000–8010, 2018.

[51] Loc Truong, Chace Jones, Brian Hutchinson, Andrew August, Brenda Praggastis, Robert Jasper, Nicole Nichols, and Aaron Tuor. Systematic evaluation of backdoor data poisoning attacks on image classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 788–789, 2020.

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

[53] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. Model agnostic defence against backdoor attacks in machine learning. *arXiv preprint arXiv:1908.02203*, 2019.

[54] Binghui Wang, Xiaoyu Cao, Neil Zhenqiang Gong, et al. On certifying robustness against backdoor attacks via randomized smoothing. *arXiv preprint arXiv:2002.11750*, 2020.

[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] Zhenting Wang, Hailun Ding, Juan Zhai, and Shiqing Ma. Training with more confidence: Mitigating injected and natural backdoors during training. *Advances in Neural Information Processing Systems*, 35:36396–36410, 2022.

[57] Tong Wu, Tianhao Wang, Vikash Sehwag, Saeed Mahloujifar, and Prateek Mittal. Just rotate it: Deploying backdoor attacks via rotation transformation. *arXiv preprint arXiv:2207.10825*, 2022.

[58] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *Proceedings of the IEEE Symposium on Security and Privacy (May 2021)*, 2021.

[59] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2041–2055, 2019.

[60] Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia. Rethinking the backdoor attacks' triggers: A frequency perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16473–16481, 2021.

[61] Yi Zeng, Minzhou Pan, Himanshu Jahagirdar, Ming Jin, Lingjuan Lyu, and Ruoxi Jia. How to sift out a clean data subset in the presence of data poisoning? *arXiv preprint arXiv:2210.06516*, 2022.

[62] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. *arXiv preprint arXiv:2005.00060*, 2020.

[63] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha Dvornek, Sekhar Tatikonda, James Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. *arXiv preprint arXiv:2203.08065*, 2022.

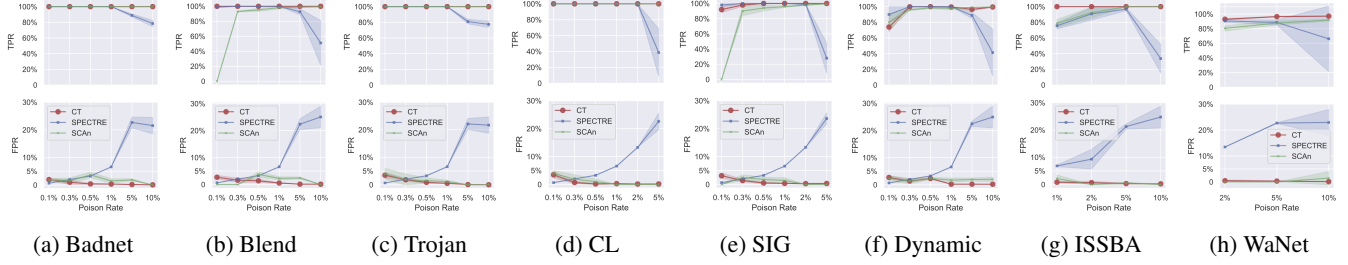| (a) Badnet | (b) Blend | (c) Trojan | (d) CL | (e) SIG | (f) Dynamic | (g) ISSBA | (h) WaNet |

Figure 6: **Ablation experiments on different poison rates.** This figure supplements Figure 3. For most attacks, the poison rates are from 0.1% to 10%. For CL and SIG, since only target class samples are poisoned, their maximum poison rates are 5% instead of 10%, to avoid the entire target class being poisoned, which may lead to severe clean performance loss. For ISSBA and WaNet, the minimum poison rates are 1% and 2%, respectively, in order to guarantee the backdoor is successfully injected.

A Implementations of Confusion Training

A.1 Full Algorithms and Implementations

In Algorithm 2 and 3, we present the algorithmic details. As formulated in Algorithm 3, our implementation of CT defense involves multiple rounds of confusion training. In our implementations for the major evaluations in Sec § 5, there are 6 rounds in total. Specifically, for the parameters in Algorithm 3, we set the number of iteration $K = 6$, the distillation ratios $\{r_1 = \frac{1}{2}, r_2 = \frac{1}{5}, r_3 = \frac{1}{25}, r_4 = \frac{1}{50}, r_5 = \frac{1}{100}\}$, number of confusion iters $m = 6000$, confusion factor $\lambda = 20$. The loss function ℓ is the standard cross entropy loss. By default, we use a reserved clean set D_{reserve} with 2000 samples. We use standard stochastic gradient descent with a weight decay of 10^{-4}, a momentum of 0.7, and a learning rate of 0.001. We set the threshold $u = 2$ in Algorithm 2.

A.2 Computational Cost

The primary computational cost of the full CT poison detection pipeline (Algorithm 3) comes from the pretraining procedure in Line 3. Given that K rounds of pretraining are conducted over sequentially shrinking datasets $\{D_k\}_{k=1}^K$, the computational overhead is projected to approximate $1 + \sum_{k=1}^{K-1} r_i \approx 2$ times that of a full pretraining (specific to the hyper-parameters discussed in Appendix § A.1). Furthermore, the m inner iterations (Line 4) resemble several epochs of fine-tuning and consume significantly fewer computational resources than pretraining. In addition, computational costs associated with subsequent steps, spanning from Line 17 to Line 22, are relatively negligible compared to the whole procedure. In summary, the complete CT poison detection framework necessitates roughly twice the computational resources required in standard pretraining and in conventional post-hoc approaches [4, 13, 47, 50] (since the latter also primarily involves pre-training). To illustrate this, we executed our confusion training pipeline and the classical poison detection baseline Spectral Signature [50] on a single GPU in our workstation, which is equipped with 48 Intel Xeon Silver 4214 CPU cores, 384 GB RAM, and 8 GeForce RTX 2080 Ti

GPUs. We document the execution time in Table 9. Note that obtaining a ready-to-deploy model necessitates further training a downstream model on the cleansed dataset, of which the computational cost depends on the model architecture and training algorithm (as discussed in Sec § 2.3).

Algorithm 2 Identify Potential Target Classes

Input: Dataset $\widetilde{D} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^n$, Inference Model θ_{ct}, The Number of Classses C, Gaussian Density Function $N(\cdot; \mu, \sigma)$, a threshold u
Output: $T \subseteq \{1, \ldots, C\}$, indicating potential target classes

1:   $T \leftarrow \{\}$
2:   **for** $c = 1, \ldots, C$ **do**
3:     $H \leftarrow \{h(\tilde{x}; \theta_{ct}) | (\tilde{x}, \tilde{y}) \in \widetilde{D} \wedge \tilde{y} = c\}$ ▷ Latent Vectors
4:     $H_0 \leftarrow \{h(\tilde{x}; \theta_{ct}) | (\tilde{x}, \tilde{y}) \in \widetilde{D} \wedge \tilde{y} = c \wedge F(\tilde{x}; \theta_{ct}) \neq c\}$
5:     $H_1 \leftarrow H \setminus H_0$
6:     $U, \Lambda, V \leftarrow SVD_2(H)$ ▷ Dimension Reduction
7:     $S \leftarrow \{U^T h | h \in H\}$ ▷ Top-2 PCA Directions
8:     $S_0, S_1 \leftarrow \{U^T h | h \in H_0\}, \{U^T h | h \in H_1\}$
9:     $\widehat{\mu}, \widehat{\sigma} \leftarrow$ empirical mean and covariance of S
10:    $\widehat{\mu}_0, \widehat{\sigma}_0 \leftarrow$ empirical mean and covariance of S_0
11:    $\widehat{\mu}_1, \widehat{\sigma}_1 \leftarrow$ empirical mean and covariance of S_1
12:    $L \leftarrow \sqrt[|S|]{\frac{\prod_{s_0 \in S_0} N(s_0; \widehat{\mu}_0, \widehat{\sigma}_0) \cdot \prod_{s_1 \in S_1} N(s_1; \widehat{\mu}_1, \widehat{\sigma}_1)}{\prod_{s \in S} N(s; \widehat{\mu}, \widehat{\sigma})}}$
13:    **if** $L > u$ **then**
14:      $T \leftarrow T \cup \{c\}$
15:    **end if**
16: **end for**
17: **return** T

B Comparison with Additional Defenses

A line of work aims to design robust training algorithms that intend to prevent models from learning any backdoor during training [16, 18, 48, 56]. The conceptual underpinning that propels these methodologies bears considerable relevance to our study, given that they also strive to proactively devise training procedures that are robust enough to counter back-

Algorithm 3 Full Algorithm of CT Poison Detection

Input: Dataset $\widetilde{D} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{n}$ to be cleansed, reserved clean set D_{reserve}, loss function ℓ, confusion factor λ, number of confusion iterations m, number of confusion training rounds K, distillation ratios $\{r_i\}_{i=1}^{K-1}$
Output: The cleansed dataset D^*

1:   $D_1 \leftarrow \widetilde{D}$
2:   **for** $k = 1, \ldots, K$ **do**
3:      $\theta_{\text{ct}} \leftarrow \mathcal{A}(D_k)$ ▷ Pretrain
4:      **for** $i = 1, \ldots, m$ **do**   ▷ Confusion Training
5:          $(\widetilde{\mathbf{X}}_i, \widetilde{\mathbf{Y}}_i) \leftarrow$ a random batch from D_k
6:          $\mathbf{X}'_i \leftarrow$ a random batch from the unlabeled D_{reserve}
7:          $\mathbf{Y}'_i \leftarrow F(\mathbf{X}'_i; \widetilde{\theta})$ ▷ Pseudo Labels
8:          $\mathbf{Y}^* \leftarrow$ random mislabels other than $\mathbf{Y}'_i$
9:          $\ell_{ct} \leftarrow \frac{\ell(f(\widetilde{\mathbf{X}}_i; \theta_{ct}), \widetilde{\mathbf{Y}}_i) + (\lambda - 1) \cdot \ell(f(\mathbf{X}'_i; \theta_{ct}), \mathbf{Y}^*)}{\lambda}$
10:         $\theta_{\text{ct}} \leftarrow$ one step gradient descent on ℓ_{ct}
11:      **end for**
12:      **if** $k < K$ **then**
13:          $D_{k+1} \leftarrow$ sort samples in $\widetilde{D}$ according to their loss values $\ell(f(\cdot; \theta_{ct}), y)$ and select the first $\lfloor r_k \cdot n \rceil$ samples with least loss values   ▷ Iterative Poison Distillation
14:      **end if**
15:   **end for**
16:   $D^* \leftarrow D$
17:   $T \leftarrow$ Apply Algorithm 2 with θ_{ct}
18:   **for** $i = 1, \ldots, n$ **do**   ▷ Poison Detection
19:      **if** $\tilde{y}_i \in T \wedge \tilde{y}_i = F(\tilde{x}_i; \theta_{ct})$ **then**
20:          $D^* \leftarrow$ remove $(\tilde{x}_i, \tilde{y}_i)$ from D^*
21:      **end if**
22:   **end for**
23:   **return** D^*

---

door attacks. In Table 2, we compared CT with ABL [18] from this category of defenses. For a more comprehensive comparison with these defenses, we evaluate three additional defenses (DBD [16], MOTH [48] and NONE [56]) from this category and evaluate them on CIFAR10 in Table 8. Our implementations strictly follow the original configurations of their open-source repositories, and our evaluation follows the same setup used for Table 2. Notably, CT continues exhibiting stronger resilience than the three additional baselines.

Table 9: **Execution time on a single RTX 2080 Ti GPU.**

(minutes)	Spectral Signature [50]	CT (ours)
CIFAR10	77	150
GTSRB	41	93

C   Full Ablation Results on Poison Rates

We hereby present ablation results on poison rates against other attacks in Figure 6, which supplement Figure 3.

Table 7: **Hyperparameters of Baseline Attacks**

	CIFAR10 [17]	GTSRB [45]	ImageNet [8]
BadNet [12]	Poison Rate = 0.3%	Poison Rate = 1.0%	Poison Rate = 1.0%
Blend [5]	Poison Rate = 0.3%	Poison Rate = 1.0%	Poison Rate = 1.0%
Trojan [26]	Poison Rate = 0.3%	Poison Rate = 1.0%	Poison Rate = 1.0%
CL [52]	Poison Rate = 0.3%	/	/
SIG [3]	Poison Rate = 2.0%	Poison Rate = 2.0%	/
Dynamic [31]	Poison Rate = 0.3%	Poison Rate = 0.3%	/
ISSBA [21]	Poison Rate = 2.0%	/	/
WaNet [30]	Poison Rate = 5.0% Cover Rate = 10.0%	Poison Rate = 5.0% Cover Rate = 10.0%	/
TaCT [47]	Poison Rate = 0.3% Source Class = 1 Cover Classes = 5,7 Cover Rate = 0.3%	Poison Rate = 0.5% Source Class = 1 Cover Classes = 5,7 Cover Rate = 0.5%	/
Adap-Patch [36]	Poison Rate = 0.3% Cover Rate = 0.3%	Poison Rate = 0.3% Cover Rate = 0.3%	/
Adap-Blend [36]	Poison Rate = 0.3% Cover Rate = 0.6%	Poison Rate = 0.5% Cover Rate = 1.0%	/

Table 8: **Comparing CT with 3 additional backdoor defenses on CIFAR10.**

(%)	mean (std)		No Defense	DBD	MOTH	NONE	CT (ours)
	No Poison	ACC	94.1 (0.1)	92.2 (0.5)	91.8 (0.3)	93.3 (0.1)	92.2 (0.3)
		ASR	-	-	-	-	-
	BadNet	ACC	93.8 (0.1)	92.2 (0.4)	91.7 (0.5)	93.3 (0.2)	93.2 (0.1)
		ASR	100 (0)	1.8 (0.1)	1.2 (0.3)	1.0 (0)	0.8 (0.2)
	Blend	ACC	93.6 (0.1)	92.0 (0.4)	91.2 (0.2)	93.2 (0.2)	93.1 (0.2)
		ASR	93.5 (0.2)	4.4 (0.3)	89.9 (1.5)	89.0 (0.8)	1.6 (0.3)
	Trojan	ACC	93.8 (0.1)	92.3 (0.3)	90.9 (0.2)	93.1 (0.1)	92.9 (0.1)
C		ASR	99.9 (0.1)	2.7 (0.2)	7.6 (3.3)	2.4 (0.2)	1.7 (0.4)
I	CL	ACC	93.6 (0)	93.1 (0.3)	91.3 (0.3)	93.3 (0.1)	93.5 (0.2)
F		ASR	99.8 (0.1)	89.8 (0.1)	1.3 (0.4)	1.2 (0.1)	0.9 (0)
A	SIG	ACC	93.8 (0)	91.5 (0.4)	90.9 (0.2)	93.2 (0)	93.2 (0)
R		ASR	80.2 (0.6)	49.8 (1.1)	98.2 (0.5)	86.4 (1.0)	0.1 (0)
10	Dynamic	ACC	93.8 (0)	92.0 (0.3)	91.5 (0.5)	93.5 (0.2)	93.2 (0)
		ASR	99.3 (0.4)	89.3 (0.1)	84.3 (7.8)	4.4 (0.4)	3.7 (1.6)
	ISSBA	ACC	93.7 (0)	90.6 (0.8)	91.0 (0.2)	93.4 (0.2)	93.2 (0)
		ASR	100 (0)	2.4 (0.1)	42.7 (27.5)	62.2 (24.7)	0.6 (0)
	WaNet	ACC	93.0 (0.5)	90.9 (0.3)	90.7 (0.2)	92.8 (0.1),	93.0 (0.2)
		ASR	93.9 (0.8)	2.1 (0.2)	5.4 (3.2)	91.5 (0.6)	1.0 (0.1)
	TaCT	ACC	93.6 (0)	91.9 (0.4)	91.3 (0.1)	93.3 (0.1)	92.5 (0.5)
		ASR	99.6 (0.2)	2.0 (0.2)	35.6 (23.8)	5.2 (2.7)	1.2 (0.1)
	Adap-Patch	ACC	93.5 (0.2)	91.8 (0.3)	91.4 (0.2)	93.0 (0.3)	92.4 (0.1)
		ASR	100 (0)	2.6 (0.3)	29.9 (15.4)	6.2 (0.7)	1.4 (0.1)
	Adap-Blend	ACC	94.0 (0)	92.7 (0.2)	91.5 (0.2)	93.1 (0.2)	92.8 (0.2)
		ASR	84.8 (1.1)	2.4 (0.1)	9.9 (7.1)	47.6 (0.7)	2.2 (0.1)

D   Configurations of Baseline Attacks

For attacks on image datasets, we follow the default configurations of their original papers and open-source implementations. In Table 7, we summarize the hyperparameters used for each poison strategy. For constrained and unconstrained attacks on Ember, we follow the strategies proposed in the original paper [42]. The constrained attack confines backdoor triggers to those features that are really editable in PE files, while the unconstrained attack allows modifying all features during poisoning. The poison rate is 1% for both attacks. For the constrained attack, the trigger watermark size is 17, with attack strategy "LargeAbsSHAP x MinPopulation"; for the unconstrained attack, the trigger watermark size is 32, with attack strategy "Combined Feature Value Selector".