Fight Poison with Poison: Detecting Backdoor Poison Samples via Decoupling Benign Correlations

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

Abstract

Deep learning suffers from backdoor poisoning attacks. By injecting only a few poison samples into a training set, adversaries can easily embed stealthy backdoor into the trained models. In this work, we study poison samples detection for defending against backdoor poisoning attacks on deep neural networks (DNNs). A principled idea underlying prior arts on this problem is to *utilize the backdoored models' distinguishable behaviors on poison and clean populations to distinguish between these two different populations themselves and remove the identified poison*. Typically, many prior arts build their detectors upon a latent separability assumption, which states that backdoored models trained on the poisoned dataset will learn separable latent representations for backdoor and clean samples. Although such separation behaviors empirically exist for many existing attacks, there is no control on the separability and the extent of separation can vary a lot across different poison strategies, datasets, as well as the training configurations of backdoored models. Worse still, recent adaptive poison strategies can greatly reduce the "distinguishable behaviors" and consequently render most prior arts less effective (or completely fail). We point out that these limitations directly come from the *passive reliance* on some distinguishable behaviors that are not controlled by defenders. To mitigate such limitations, in this work, we propose the idea of *active defense* — rather than passively assuming backdoored models will have certain distinguishable behaviors on poison and clean samples, we propose to actively enforce the trained models to behave differently on these two different populations. Specifically, we introduce *confusion training* as a concrete instance of active defense. Confusion training separates poison and clean populations by introducing another poisoning attack to the already poisoned dataset, which actively decouples the benign correlations and leave backdoor correlations the only learnable patterns — consequently, only backdoor poison samples can be fitted, while clean samples are underfitted. In short, we literally invite a "defensive poison" in to fight the original backdoor poison we aim to cleanse. By extensive evaluations on both CIFAR10 and GTSRB, we show superiority of active defense across a diverse set of backdoor poisoning attacks.

1 Introduction

The success of deep learning heavily relies on large-scale modern datasets [2, 21, 33]. To build such large datasets, data collection procedures are typically automated and harvest information from the open world, while the data labeling processes are often outsourced to third-party labors. Such pipelines make it **intrinsically challenging to conduct strict supervision over the creation of datasets**, and consequently give rise to the threat of *backdoor poisoning attacks* [4, 12, 20, 27, 41], where adversaries can inject carefully crafted poison samples into the victim training set such that models trained on the contaminated set will be backdoored. Backdoor poisoning attacks are

Preprint. Under review.

considerably risky, because those backdoored models behave almost the same as normal models under standard evaluation metrics but make catastrophic mistakes stealthily triggered by adversaries.

Following the emerging threat of such attacks, a natural question to ask is — *if anyway we have to use a potentially poisoned dataset, is there a principled way for us to* ***distinguish between normal clean samples and potentially abnormal poison samples****?* If so, then we can effectively mitigate the risks of dataset contamination in the first place by simply removing suspicious samples from the raw datasets, before which are used by downstream pipelines. In this work, we look into this question under the threat model of backdoor poisoning attacks. In particular, we study the problem of detecting backdoor poison samples in poisoned datasets.

Limitations of Prior Arts There are a number of prior arts [3, 11, 13, 39, 38] for detecting backdoor poison samples. A principled idea underling these work is to *utilize the backdoored models' distinguishable behaviors on poison and clean populations to distinguish between these two different populations themselves*. Typically, many prior arts [39, 3, 13, 38] consistently build their detectors upon a latent separability assumption, which states that backdoored models trained on the poisoned dataset will learn separable latent representations for backdoor and clean samples and thus poison samples can be identified via cluster analysis in latent representation space. Alternatively, Gao et al. [11] also argue that backdoor models' predictions on poison samples have less entropy under intentional perturbation compared with that of clean samples. Despite the effectiveness of these work in the respective settings they consider, we find that their performances can vary a lot across different poison strategies, datasets and training configurations. Worse still, one recent work [30] further shows adaptive poisoning strategies can be developed to suppress the "distinguishable behaviors" relied by prior arts, rendering them much less effective (or completely fail in most cases). We point out that these limitations are direct consequences of the ***passive strategies*** adopted by these work — they consistently rely on some properties of the backdoored models that are not controlled by defenders.

Our Contributions. In this work, we propose the idea of ***active defense*** — rather than *passively expecting* backdoored models will magically have certain distinguishable behaviors on poison and clean samples, we propose to **actively enforce** the trained models to behave differently on these two different populations. To this end, we introduce ***confusion training*** as a concrete instance of active defense for detecting backdoor poison samples. Confusion training enforces models to have the distinguishable behaviors by introducing another poisoning attack to the already poisoned dataset, such that the benign correlations between semantic features and semantic labels are actively decoupled, while the backdoor correlations between backdoor triggers and target labels are left to be the only learnable patterns — consequently, only backdoor poison samples are fitted while clean samples are underfitted, and one can easily separate samples from these two populations via checking the fitting status. By extensive evaluations on both CIFAR10 [17] and GTSRB [37], we show superiority of our active defense over passive defenses, across a diverse set of backdoor poisoning attacks. Our code is publicly available [1].

2 Background and Related Work

Backdoor Poisoning Attacks. A deep neural network (DNN) model is said to be backdoored, if the model has encoded certain anomalous rules (i.e. backdoors) that are exclusively known to some adversaries while it still behaves normally under standard evaluations. DNN models can be backdoored in various ways, including data poisoning [12, 4], transfer learning [50, 34] and even direct model weights tampering [29, 31], etc. In this work, we focus on data poisoning based backdoor attacks, namely the ***backdoor poisoning attacks***, which are the most commonly considered settings in the literature of backdoor attacks on DNNs. To our best knowledge, backdoor poisoning attacks are first demonstrated by Gu et al. [12]. They manipulate a small number of images in clean training sets, stamp them with a fixed patch pattern and mislabel them to a target class, and observe that models trained on such datasets consistently learn a dominant backdoor correlation between the patch pattern and the target class. Subsequent work further propose different poison strategies for further improving the practicality of the attacks. These improvements include different types of triggers [4, 27], clean label attacks [41, 1], adaptive attacks [30] that can evade certain defenses, etc.

[1] `https://github.com/Unispac/Fight-Poison-With-Poison`

Backdoor Defenses. By exploiting certain properties of some backdoor attacks, various backdoor defenses [3, 10, 11, 16, 19, 22, 25, 39, 42, 44, 46, 49, 15] are also developed. As typical examples, Neural Cleanse [46] and SentiNet [7] propose to reverse engineer the backdoor triggers via searching for universal adversarial perturbations. Fine-pruning [22] suggests to eliminate model backdoors via network pruning. Du et al. [10] and Li et al. [18] propose to apply additional intervention during model training to suppress the influence of poison samples. Xu et al. [49] propose to directly train meta classifiers to predict whether a model is backdoored. Liu et al. [25] and Li et al. [19] attempt to eliminate model backdoors via finetuning the model on a small clean dataset. Udeshi et al. [42], Villarreal-Vasquez and Bhargava [44] introduce input preprocessing to suppress the effectiveness of backdoor triggers during inference stage. There are also some certified defenses [45, 47] that adapt randomized smoothing [8] for the settings of backdoor attacks and achieve nontrivial certified accuracy for backdoor attacks with small ℓ_p bounded triggers. Besides, another line of prior arts [39, 3, 11, 38, 13] look into ***backdoor poison samples detection***. Finally, it's also important to note that some empirical defenses (e.g. [46, 7, 11] etc.) have already known to fail against some more sophisticated poison strategies, while certified defenses lack of utility because of low certifiable accuracy and the restriction to ℓ_p bounded triggers. We refer interested readers to Li et al. [20] for a more comprehensive review.

Backdoor Poison Samples Detection. This work focus on the detection of backdoor poison samples, which aims to eliminate backdoor poison samples from the training set. This line of defenses are particularly attractive for the threat model of backdoor poisoning attacks that we consider in this work. On the one hand, if one can accurately identify poison samples and eliminate them from training sets, the threat of backdoor poisoning attacks can be prevented in the first place and those cleansed datasets can be reliably used for any downstream tasks. On the other hand, even if one can only isolate a certain amount of poison samples, they can still be effectively used to unlearn the backdoor (e.g. [18, 46] etc.) — thus the poison detection can also be used as an important building block for backdoor defense. However, many prior arts on this task passively rely on some properties of backdoored models that can not be controlled by defenders, and consequently suffer from both instability problem and adaptive attack [30]. This work takes one step further by proposing the idea of ***active defense*** and introducing ***confusion training*** as a concrete instance, which significantly improves the performance of poison detection in practice. (We refer interested readers to Appendix A for a more comprehensive review.)

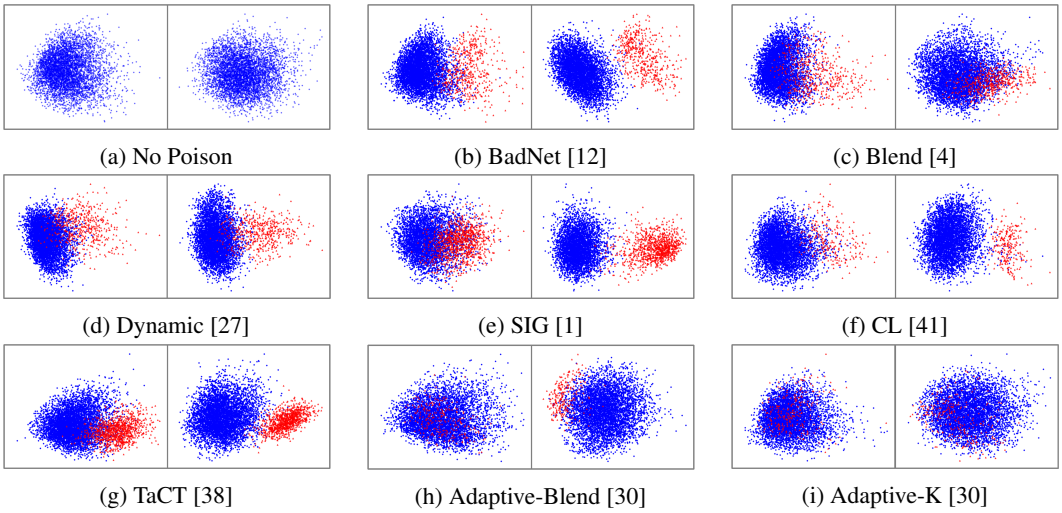
3 Preliminaries

In this section, we first introduce our threat model (Section 3.1) and the basic setting of our defenders (Section 3.2). Then, in Section 3.3, we introduce the latent separability characteristic that is widely used by existing backdoor samples detectors, and discuss problems faced by these prior work.

3.1 Threat Model

Consistent with Tran et al. [39] and Hayase et al. [13], we consider the standard threat model of backdoor poisoning attacks. Specifically, we assume the adversary can manipulate a limited portion of training data, and we also allow the adversary to have knowledge of the victim's network architecture and training algorithm. Besides, the adversary neither controls the training process nor the environment where models are deployed. Thus, the victim can freely train their models on the training data, which though might be contaminated by the adversary.

Moreover, the assumed adversary manipulates the training data with the goal of conducting backdoor attack and two constraints should be satisfied. First, with vanilla training algorithm, models trained on the poisoned dataset should have comparable accuracy to that of models trained on unpoisoned dataset on standard data distribution. Second, models normally trained on the poisoned dataset should be backdoored — when clean data points from non-target classes are corrupted by the attacker-specified trigger, the affected models should misclassify a nontrivial portion of these corrupted points to the target class.



(a) No Poison (b) BadNet [12] (c) Blend [4]

(d) Dynamic [27] (e) SIG [1] (f) CL [41]

(g) TaCT [38] (h) Adaptive-Blend [30] (i) Adaptive-K [30]

Figure 1: **Visualization of the latent representation space (CIFAR10).** Each subfigure (1a~1i) corresponds to one poison strategy. For each poison strategy, two backdoored models are trained with and without data augmentation respectively. The *left-hand* plot in each subfigure corresponds to the model trained without data augmentation, while the *right-hand* plot corresponds to the other. Clean samples are plotted as blue points and poison samples are red.

3.2 Goals and Capability of Our Defense

In this work, our defender considers the task of poison samples detection, which aims to eliminate backdoor poison samples from poisoned datasets and prevent backdoor attacks in the first place. On the other hand, the defender also aims to control the false positive rate of the defense at a low level — that is, by eliminating potential poison samples, only a small portion of clean samples are sacrificed. Besides, the defender requires the performance of defensive techniques to be stable across different datasets, poison strategies and other irrelevant random factors.

Our defender has full control over the training process. Given a potentially poisoned dataset, the defender is allowed to freely use it to train her own model and inspect both the trained model and the dataset. Similar to Tang et al. [38], we also assume the defender has access to a small reserved set of clean data (not necessarily labeled). One can expect that this small clean set is collected with strict and expensive supervision, in contrast to the given training set which is much larger and collected in a much cheaper way. We argue that this a reasonable assumption, and it is also prevalent in many prior work that study backdoor defense (e.g. [24, 22, 23, 40, 51, 19]).

3.3 Latent Separability and Adaptive Backdoor Poisoning Attacks

One principled idea for detecting backdoor poison samples is to utilize the backdoored models' distinguishable behaviors on poison and clean populations to distinguish between these two populations themselves. **Arguably, the most popular and successful characteristic is the latent separability phenomenon** first observed by Tran et al. [39]. The basic observation is that backdoored models trained on poisoned datasets tend to learn separable latent representations for poison and clean samples — thus poison samples and clean samples can be separated via a cluster analysis in the latent representation space of the backdoored models. Most state-of-the-art backdoor samples detectors [39, 3, 38, 13] are also explicitly built upon this characteristic.

However, one aspect that has not been seriously considered by previous work is — **across different settings, the extent of the latent separation may not always be as significant as what the defenders ideally expect**, and consequently the cluster analysis in latent representation space will also fail to identify the poison populations. For example, the latent separation characteristic might just be intrinsically weaker for some poison strategies than that of others. In the most extreme cases, motivated adversaries may even develop **adaptive backdoor poisoning attacks** that can intentionally suppress the latent separation characteristic. This is not just a conceptual assumption. Very recently, Qi et al.

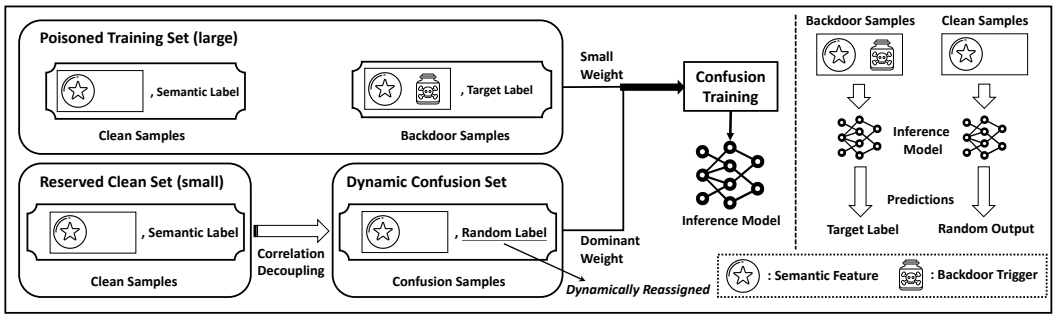


Figure 2: An overview of the confusion training pipeline.

[30] indeed come up with such adaptive backdoor poisoning attacks and successfully circumvent state-of-the-art backdoor samples detectors built on the latent separability. Besides, **other factors** like the training configurations (e.g. whether to apply data augmentation) of the base backdoored models may also impact the separation characteristics.

To illustrate this problem, we pick a diverse set of poison strategies [12, 4, 27, 1, 41, 38, 30] (including adaptive ones), and in Figure 1 we visualize (refer Apppendix C for detailed configurations) the latent representation space of their corresponding backdoored models. As shown, although the latent separation is notable in many cases, the characteristics can also be hard-to-identify when adaptive poison strategies (Figure 1h,1i) are used. Besides, in many cases (Figure 1e,1g,1h), models trained with (right-hand plots) and without (left-hand plots) data augmentation also exhibit very different extent of separation. We refer interested readers to Appendix B for a more thorough discussion.

4 Towards Active Defense

In this section, we first motivate the general idea of active defense (Section 4.1), and then we will describe (Section 4.2) a concrete instance of active defense, namely the confusion training, that we propose in this work.

4.1 From Passive Defense to Active Defense

Distinguishable characteristics of backdoor samples enable defenses to identify them. However, as we have discussed in Section 3.3, such characteristics can vary and even be intentionally suppressed by adaptive attacks [30]. Consequently, performance of existing detectors that are built on such characteristics can also vary or catastrophically degrade. We point out that such a situation is a direct consequence of the ***passive strategies*** adopted by these work. Take the latent separability characteristic for example, although it plays a central role for backdoor samples detection, prior designs only passively assume that such characteristics will naturally arise in backdoor learning. That is to say, the fundamental building block of the defensive construction is actually not under control of the constructor.

In this work, we take one step further by proposing the idea of ***active defense***. Our basic philosophy is that — rather than *passively assuming* backdoored models will naturally have distinguishable behaviors on poison and clean populations, one should actively enforce the models trained on poisoned set to behave differently on the two populations. To illustrate this methodology, we introduce ***confusion training*** as a concrete instance in the next subsection.

4.2 Confusion Training

In Figure 2, we sketch an overview of the confusion training pipeline. To be straightforward, we also abstract the procedure of confusion training in Algorithm 1. As illustrated, our defender initially has a small **reserved clean set** at hand, which is collected with strict supervision. From this small clean set, the defender will further construct a **dynamic confusion set** by mislabeling all the reserved clean samples to random labels. By "dynamic", we mean the random labels of this set will be dynamically reassigned when the set is used in the downstream of the pipeline. Conceptually, such a dynamic

random labeling process completely **decouples the benign correlations** between *semantic features* and *semantic labels* in the confusion set. Moreover, its dynamic nature will also **prevent models from naively memorizing**.

Algorithm 1: Sketch of Confusion Training

Input: Randomly initialized model $F(\cdot;\theta)$, loss function L, dataset $D_{train} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ to be cleansed, reserved clean set $D_{clean} = \{\hat{\mathbf{x}}_j, y_j\}_{j=1}^{M}$, confusion training strength λ, number of confusion batches I

Output: The cleansed dataset D^*

$\theta \leftarrow$ pretrain $F(.;\theta)$ on D_{train};
for $i = 1, ..., I$ **do**
  $(\mathbf{X}_i, \mathbf{Y}_i) \leftarrow$ random batch from D_{train};
  $(\widehat{\mathbf{X}}_i, \widehat{\mathbf{Y}}_i) \leftarrow$ random batch from D_{clean};
  $\widehat{\mathbf{Y}}_i \leftarrow$ random reassignment;
  loss $\leftarrow L(F(\mathbf{X}_i;\theta), Y_i) + \lambda \cdot L(F(\widehat{\mathbf{X}}_i;\theta), \widehat{Y}_i)\Big)$;
  $\theta \leftarrow$ one step gradient descent on loss;
$D^* \leftarrow D_{train}$;
for $i = 1, ..., N$ **do**
  $\tilde{y}_i \leftarrow F(\mathbf{x}_i;\theta)$
  if $\tilde{y}_i == y_i$ **then**
    $D^* \leftarrow$ remove $(\mathbf{x}_i, y_i)$ from D^*

Confusion training (Algorithm 1) works by training an **inference model** jointly on the poisoned training set and the dynamic confusion set. Specifically, during the training, the confusion set is assigned a **dominant weight** which is much larger than that assigned to the poisoned training set. As a result, the **confusion set will serve as an another strong poison** and dominate the training. On the one hand, this will significantly disrupt models' fitting on clean samples. On the other hand, since the confusion set does not contain any information about the backdoor trigger, it will not disrupt the fitting of backdoor correlation. Thus, at the end of confusion training, the resulting inference model fail to fit most clean samples but still perfectly fit the poison samples. **This is exactly a typical example how an active defender can intentionally induce the distinguishable behavior.** As illustrated in Figure 2, one can expect that the inference model will output a random label given a clean input, while always correctly output the target label given a backdoor input. Thus, at the end of the day, the task of **identifying backdoor poison samples** is reduced to a simple task of finding samples whose labels are consistent with the predictions of the inference model, which is similar to the process of a naive label-only membership inference [6]. Note that, in practice, our implementation involves multiple rounds of confusion training for better performance, which is more sophisticated than the sketch version we present in Algorithm 1. However the key philosophy is faithfully reflected by the sketch description. We refer interested readers to Appendix D for more technical details.

5 Empirical Evaluation

In this section, we present our experiment results. We first describe the setup of our experiments (Section 5.1), and present the main evaluation results (Section 5.2) on confusion training pipeline and other baseline defenses.

5.1 Setup

Datasets. We consider backdoor poisoning attacks on both the CIFAR10 [17] and the GTSRB [37] datasets. For each dataset, we follow the default training/test set split by Torchvision [26]. Recall that our defender is assumed to have a small reserved clean set at hand (as defined in Section 3.2). To implement this setting, for both datasets, we randomly pick 2000 samples from the test split to simulate the reserved clean set, and leave the rest part of the test split for evaluation.

Models. Similar to Tran et al. [39], we use ResNet20 [14] as the base architecture for implementing all experiments presented in this section. Specifically, for each poison strategy we consider, we will use ResNet20 to train base backdoored models on the corresponding poisoned training set. The training follows standard training scripts that are commonly used by prior work, and the trained base models are used as building blocks for baseline detectors. As for our confusion training pipeline, ResNet20 is also consistently used for building inference models. See Appendix E for training details.

Attacks. We consider eight different backdoor poisoning attacks in our evaluation. These attacks correspond to a diverse set of poisoning strategies. BadNet [12] (Figure 1b) and Blend [4] (Figure 1c) correspond to typical all-to-one dirty-label attacks with patch-like trigger and blending based trigger respectively. Dynamic [27] (Figure 1d) corresponds to the strategy that use sample-specific triggers in place of single universal trigger. SIG [1] (Figure 1e) and CL [41] (Figure 1f) correspond to two typical clean-label poisoning strategies. TaCT [38] (Figure 1g) focus on source-specific attack. Finally, Adaptive-Blend (Figure 1h) and Adaptive-K (Figure 1i) are adaptive poisoning strategies suggested by Qi et al. [30] that can suppress the latent separability characteristic. During implementing these attacks, we follow the protocols and suggested default configurations of their original papers and open-source implementations. On CIFAR10, all of the eight attacks are implemented and evaluated. On GTSRB, we omit CL since the original paper only releases poison set for CIFAR10. In Appendix E, we present the detailed configurations of all the considered attacks.

Defenses. To illustrate the superiority of active defense, we compare confusion training with five prior arts that also work on the task of backdoor poison samples detection, including three typical techniques ([39, 3, 11]) that are commonly considered in the literature and two very recent state-of-the-art methods [38, 13]. As have been mentioned at the beginning of this work, these prior arts all belong to the paradigm of passive defense — all of them passively rely on certain "distinguishable behaviors" of backdoored models that they do not control. Specifically, Strip [11] assumes that backdoor models' predictions on poison samples have less entropy under intentional perturbation, while the rest four methods all rely on the latent separability characteristic that we introduce in Section 3.3. As for our confusion training pipeline, the technical details are discussed in Appendix D. All the configurations are also detailed in Appendix E.

Evaluation Protocols. We report four metrics in our evaluation. For each defense, we report its elimination rate (true positive rate) and sacrifice rate (false positive rate) against each poison strategy that we evaluate. Elimination rate denotes the ratio of poison samples that are correctly identified by the detector, while sacrifice rate denotes the ratio of clean samples that are mistakenly diagnosed as poison. After applying a detector to scan a dataset, we cleanse the dataset by removing all those samples that are identified as potential poison. Then, we train a new model from scratch on the cleansed dataset, and report the clean accuracy and the attack success rate (ASR) of this new model. Clean accuracy denotes the accuracy on clean test inputs, while ASR denotes the success rate of activating the backdoor with backdoor inputs. Considering the randomness that exist in both attacks and defenses, we **repeat all evaluations for three times** with different random seeds, and finally report the average results across the three repeated evaluations.

Data Augmentation During our evaluation of baseline defenses, we also seriously consider the effects of data augmentation. Specifically, for each poison strategy, we train two base backdoored models both with and without data augmentation, and evaluate performance of baseline poison detectors built on these two base models respectively. Note that, for our confusion training, we do not use any data augmentation.

5.2 Performance of Defenses

During our empirical study, we observe that data augmentation can improve the performance of those baseline poison detectors in many cases, however, it can also make the performance worse in some cases. Thus, **there is no generally consistent conclusion that can claim data augmentation is always preferable (or undesirable)**. We will further discuss this issue in Appendix F.2. For now, in this subsection, we summarize the performance of the evaluated defenses. For conciseness, when we summarize our evaluation results for baseline defenses, we always pick the best result out of the both cases (w/ and w/o data augmentation). These numbers reflect upper bounds of the baseline defenses — in practice, the defender may not know whether data augmentation would be better or not for a specific poisoned dataset, and thus can only choose one of the two options. On the other hand, since our confusion training does not involves data augmentation, the reported numbers for our defense are from a single consistent configuration. Still, we will see, although we consider the best possible results for baselines, our defense still achieves consistently superior performance.

We present the summarized results in Table 1 and Table 2 for CIFAR10 and GTSRB respectively, and we defer the full results to Appendix F. We consider a defense is successful against an attack,

	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	CL [41]	SIG [1]	TaCT [38]	Adap-Blend [30]	Adap-K [30]
Without Defense	ASR	/	99.9	96.7	99.4	97.5	72.9	99.1	53.9	97.3
	Clean Accuracy	92.0	91.4	91.6	91.6	91.7	91.5	91.8	91.7	91.6
Spectral Signature [39]	Elimination Rate	/	91.1	68.9	79.7	99.7	94.3	93.5	61.3	11.3
	Sacrifice Rate	15.0	14.3	14.5	14.3	7.0	28.7	28.7	7.3	7.5
	ASR	/	63.8	94.3	96.6	2.3	12.1	69.4	29.8	98.8
	Clean Accuracy	90.9	90.9	90.8	90.6	91.7	89.9	89.7	91.5	91.2
Activation Clustering [3]	Elimination Rate	/	99.1	87.3	92.2	100.0	96.4	99.8	0.0	43.1
	Sacrifice Rate	12.3	0.8	3.3	5.4	0.9	2.5	1.9	0.0	37.0
	ASR	/	1.2	85.1	93.7	1.4	1.8	1.7	53.9	95.4
	Clean Accuracy	91.8	91.9	91.6	91.6	91.6	91.7	91.5	91.7	90.4
Strip [11]	Elimination Rate	/	100.0	50.6	99.4	100.0	83.5	77.3	1.1	10.0
	Sacrifice Rate	9.9	10.4	10.3	10.1	10.5	10.8	10.5	10.3	10.5
	ASR	/	1.1	96.1	10.7	1.5	68.0	98.3	55.0	96.8
	Clean Accuracy	91.7	91.6	91.5	91.6	91.5	91.6	91.2	91.4	91.1
SCAn [38]	Elimination Rate	/	100.0	93.0	97.4	100.0	95.5	100.0	0.0	0.9
	Sacrifice Rate	0.0	0.0	2.6	1.4	2.9	0.0	0.0	0.0	3.1
	ASR	/	1.1	26.7	33.7	1.5	2.3	1.2	53.9	97.1
	Clean Accuracy	92.0	91.5	91.8	91.5	91.6	91.6	91.7	91.7	90.5
SPECTRE [13]	Elimination Rate	/	100.0	97.4	99.5	100.0	90.4	99.9	95.1	22.5
	Sacrifice Rate	1.5	0.5	0.5	0.5	0.3	1.2	1.0	0.3	0.7
	ASR	/	1.2	8.0	10.7	1.6	20.6	1.8	2.7	71.6
	Clean Accuracy	91.6	91.8	91.6	91.6	91.7	91.2	91.7	91.8	91.3
Confusion Training (Ours)	Elimination Rate	/	100.0	100.0	99.9	100.0	100.0	100.0	99.7	92.4
	Sacrifice Rate	0.0	0.5	0.9	0.8	0.6	0.7	1.0	1.2	2.7
	ASR	/	1.1	1.8	5.7	1.5	0.1	0.9	0.8	3.5
	Clean Accuracy	92.0	91.8	91.6	91.5	91.6	91.4	91.5	91.6	91.2

Table 1: **Performance of backdoor poison samples detectors on CIFAR10.** (average)

	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	SIG [1]	TaCT [38]	Adap-Blend [30]	Adap-K [30]
Without Defense	ASR	/	99.8	96.0	100.0	63.1	99.9	48.9	63.1
	Clean Accuracy	98.1	97.7	98.1	98.0	98.1	98.4	97.7	97.8
Spectral Signature [39]	Elimination Rate	/	100.0	77.1	93.6	66.9	95.9	61.1	74.4
	Sacrifice Rate	39.1	38.3	38.6	38.4	49.7	48.3	25.5	25.4
	ASR	/	0.1	92.0	45.8	57.5	33.2	30.8	19.4
	Clean Accuracy	96.8	96.9	96.8	96.8	95.3	96.8	96.7	96.7
Activation Clustering [3]	Elimination Rate	/	99.7	81.5	81.3	28.6	100.0	0.0	0.0
	Sacrifice Rate	0.5	0.7	1.3	0.6	1.5	2.7	0.1	0.3
	ASR	/	0.1	90.3	99.7	48.9	0.0	49.2	67.3
	Clean Accuracy	97.6	97.6	97.6	97.3	98.1	97.3	97.6	96.3
Strip [11]	Elimination Rate	/	100.0	73.9	100.0	43.8	37.7	3.8	4.8
	Sacrifice Rate	8.8	9.5	9.3	7.8	9.7	11.3	9.7	9.8
	ASR	/	0.1	93.9	1.2	60.0	99.9	52.5	68.1
	Clean Accuracy	97.2	97.2	97.2	97.7	97.7	98.0	97.0	96.6
SCAn [38]	Elimination Rate	/	100.0	81.3	89.0	85.8	100.0	28.3	35.6
	Sacrifice Rate	1.6	3.2	2.2	2.9	2.6	2.2	2.3	4.6
	ASR	/	0.1	87.1	91.8	10.1	0.0	33.6	47.4
	Clean Accuracy	97.7	97.1	97.4	97.4	97.7	97.0	97.4	97.5
Confusion Training (Ours)	Elimination Rate	/	100.0	99.5	99.6	99.7	99.6	98.0	87.2
	Sacrifice Rate	0.1	0.3	1.0	0.7	0.2	2.2	0.8	3.5
	ASR	/	0.1	1.6	0.5	0.1	0.2	0.8	0.7
	Clean Accuracy	98.0	97.8	97.9	98.1	97.7	98.0	97.9	96.8

Table 2: **Performance of backdoor poison samples detectors on GTSRB.** (average)

if the ASR is reduced below 20%; otherwise we say the defense is unsuccessful. In the tables, for each (attack, defense) pair, we highlight the attack success rate (ASR) with green color for successful cases and red color for unsuccessful cases.

Confusion training is robust across different poison strategies. As shown, consistent with the intuitions that we illustrate in Figure 1, on both datasets, all baseline defenses (built on the passive defense paradigm) are *susceptible to* at least one of the *adaptive backdoor poisoning attacks* [30] (Adap-Blend, Adap-K), and the elimination rates catastrophically degrade. Moreover, even for non-adaptive attacks, performance of these baseline defenses still vary a lot across different poison strategies. As we have discussed in prior sections, these are exactly the limitations of passive defense. In contrast, *on both datasets, our confusion training pipeline (built on the active defense paradigm) can always eliminate almost all poison samples against all poison strategies (including adaptive ones) we consider*, and models trained on the cleansed dataset only have negligible ASR.

Confusion training is robust across different datasets. Another interesting observation is that, even for a same (attack, defense) pair, the elimination rate often baseline defenses can also vary. For example, on CIFAR10, activation clustering can eliminate 96.4% of the poison samples on average against SIG, however, the number drops to 28.6% for GTSRB. A more extreme example is SPECTRE.

On CIFAR10, it exhibits state-of-the-art results among baselines and can even defeat one of the adaptive attacks (Adap-Blend). However, we find that it completely fails in most cases on GTSRB, because the dataset is highly unbalanced and consequently it often can not correctly identify the target class. For this reason, we omit it (refer Appendix F for the failure results) in Table 2. In comparison, as shown, our confusion training pipeline consistently performs well on both datasets.

Confusion training has low sacrifice rate. Moreover, we highlight that our confusion training pipeline has low sacrifice rate (false positive rate). In all evaluated cases, the sacrifice rates are consistently kept at a low level (below 1% in most cases). Note that, advantage of low sacrifice rate is not obviously reflected by the clean accuracy. Because, both GTSRB and CIFAR10 have a large samples size, and even if more clean data are sacrificed, the drop of clean accuracy is only within several percents. However, we point out that the low sacrifice rate itself has its own merit. For example, one can use all the detected suspicious samples to do unlearning (e.g. [18, 46]). If the sacrifice (false positive) rate is low, then the backdoor correlation can be unlearned without hurting clean accuracy too much. This can be a useful follow-up remedy when the elimination rate is low.

6 Discussion

In Section 5.2, we witness a big win of our confusion training pipeline over prior arts, across different datasets and poison strategies. Even though, **we keep a conservative attitude to oversell the security of the confusion training technique itself**. After all, there is still no strict guarantee that the technique would be reliable in all settings. Rather, we deem the success of confusion training as a strong evidence for the **superiority of our active defense methodology** that is underlying the design of our techniques. As we have discussed in Section 3.3, illustrated in Figure 1, and also empirically validated in Section 5.2, it is suboptimal to just passively hope that some properties will automatically arise in backdoored models and then build defenses that heavily rely on such properties. Although this conclusion is very straightforward, it is surprising to see that a lot of prior works including very recently published state-of-the-arts are still built on such a passive paradigm. By our work, we want to promote awareness of the active defense methodology, and we encourage future work on backdoor samples detection to incorporate this idea into their designs.

Finally, we also note that, since we reveal many failure cases of prior defenses. These information might potentially be used to threat systems built on these defenses.

7 Conclusion

In this work, we look into the problem of backdoor poison samples detection as a defense against backdoor poisoning attacks. We analyze limitations of prior arts on this problem, including inconsistent performance across different settings and susceptibility to adaptive poison strategies. We attribute these limitations to the passive defense paradigm that these prior arts follow — that they passively assume backdoored models will naturally have distinguishable behaviors on poison and clean populations. As an alternative, we propose the methodology of active defense, which suggests that one should actively enforce the models trained on poisoned set to behave differently on the two populations. To illustrate this methodology, we introduce the technique of confusion training as a concrete instance. Confusion training introduces a dynamic confusion set as another strong poison to interrupt models' fitting on clean samples, such that the trained models only fit poison samples while clean samples are underfitted. This naturally induces a separation between poison samples and clean samples. Then, we compare our confusion training pipeline with other baseline defenses across different poison strategies and datasets, and validate the superiority of our method and the active defense methodology underling its design. By our work, we point out that active defense is a promising direction for mitigating backdoor poisoning attacks. We encourage future work to incorporate this idea into their designs.

References

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

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

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

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

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

[6] 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] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1310–1320, Long Beach, California, USA, 2019.

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

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

[16] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 301–310, 2020.

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

[21] 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.

[22] 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.

[23] 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.

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

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

[26] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

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

[28] 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.

[29] Xiangyu Qi, Tinghao Xie, Ruizhe Pan, Jifeng Zhu, Yong Yang, and Kai Bu. Towards practical deployment-stage backdoor attack on deep neural networks. *arXiv preprint arXiv:2111.12965*, 2021.

[30] Xiangyu Qi, Tinghao Xie, Saeed Mahloujifar, and Prateek Mittal. Circumventing backdoor defenses that are based on latent separability. Technical report, Princeton University, May 2022.

[31] 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.

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

[33] 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.

[34] 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.

[35] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing gaussian mixture models with em using equivalence constraints. *Advances in neural information processing systems*, 16, 2003.

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

[37] 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.

[38] 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.

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

[40] 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.

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

[42] 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.

[43] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[44] Miguel Villarreal-Vasquez and Bharat Bhargava. Confoc: Content-focus protection against trojan attacks on neural networks. *arXiv preprint arXiv:2007.00711*, 2020.

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

[46] 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.

[47] Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. *arXiv preprint arXiv:2003.08904*, 2020.

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

[49] 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.

[50] 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.

[51] 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.

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

A A Supplementary Review on Backdoor Poison Samples Detection

A.1 The Important Role of Backdoor Poison Samples Detection

This work looks into the problem of backdoor poison samples detection. More precisely, we focus on offline detection that aims to identify backdoor poison samples in a fixed training set. **This line of defenses are particularly attractive for the threat model of backdoor poisoning attacks** that we consider in this work, for two reasons:

- From a utility perspective: if one can accurately identify poison samples and eliminate them from training sets, the threat of backdoor poisoning attacks can be prevented in the first place and those cleansed datasets can be reliably used for any downstream tasks. Thus, the threat of backdoor poisoning can be resolved once and for all. On the other hand, even if poison samples can not be fully eliminated — as long as one can accurately isolate a certain amount of poison samples, they can still be effectively used to unlearn the backdoor (e.g. [18, 46] etc.) — thus the poison detection can also be used as an important building block for backdoor defense.
- From a methodological perspective: we point out that **a "good" backdoor defense is always approximately equivalent to a "good" backdoor poison samples detector**. On the one hand, given a "good" detector that can eliminate (almost) all poison samples without sacrificing too many clean samples, models trained on the cleansed dataset will have negligible ASR (attack success rate) but still keep high clean accuracy (e.g. see our confusion training in Table 1,2) — **thus, it is a "good" defense**. On the other hand, if one defense (not necessarily based on poison detection) is "good", the resulting model should have negligible ASR and have high clean accuracy. So, the model's predictions on most poison samples will be inconsistent with their labels, while the predictions on most clean samples would be consistent. Based on this result, **one can equivalently build a "good" poison detector with this model**, by simply removing all training samples whole labels are not consistent with the predictions — most poison samples are removed while clean samples are kept. Thus, in some sense, **the seemingly larger problem of backdoor defense is equivalent to the sub-problem of backdoor poison samples detection**.

In short, above discussions suggest the fundamental role of backdoor poison samples detection underlying backdoor defense, and it also justifies why we are particularly interested in this problem.

A.2 The Distinguishable Behaviors

There are a number of prior arts for detecting backdoor poison samples. A **principled idea** underling most of these work is to *utilize the backdoored models' **distinguishable behaviors** on poison and clean populations to distinguish between these two different populations themselves*.

Typically, many prior arts [39, 3, 13, 38] consistently build their detectors upon a **latent separability assumption**, which states that backdoored models trained on the poisoned dataset will learn separable latent representations for backdoor and clean samples and thus poison samples can be identified via cluster analysis in latent representation space.

To our best knowledge, Tran et al. [39] is the first to observe the latent separation phenomenon. Specifically, they train a backdoored model on the poisoned dataset and use it to map all training samples to the latent representation space. Then, for samples of each class, they project their representations to their top PCA [28] directions. Interestingly, for all non-target classes that only contain clean samples, their latent representations regularly distribute around a single center; however, for the target class that contains both clean and poison samples, the latent representations clearly form a bimodal distribution — poison and clean samples form their own modal respectively. Thus, poison samples can then be eliminated by simply dropping the poison modal. Later, Chen et al. [3] also independently observe this separation phenomenon and propose to use K-means to separate the clean and poison clusters in the latent space. Note that, both Tran et al. [39] and Chen et al. [3] launch their cluster analysis in an unsupervised manner and this may not work well in more challenging cases. Thus, more recently, Tang et al. [38] and Hayase et al. [13] further propose to take use of the knowledge of clean distribution which significantly improves the cluster analysis.

Despite the effectiveness of these latent separability based techniques in the respective settings they consider, as we have discussed and illustrated in our main text (Section 3.3, Section 5.2), their performances can vary a lot across different poison strategies, datasets and training configurations. Worse still, adaptive attacks that can intentionally suppress this separability characteristic can make then degrade catastrophically.

Beside the latent separability, **other heuristic characteristics** are also utilized to detect backdoor poison samples. For localized triggers, Sentinet [7] use backdoored models to compute the saliency maps of each sample and locate suspicious sample via checking whether their saliency maps contain small connected regions of high salience. Strip [11] also observe that when samples are intentionally superimposed with random image patterns, backdoored models' predictions on poison samples have smaller randomness compared with that of clean samples. However, these characteristics are not as successful as the latent separability. Sentinet [7] is designed only for local trigger, while Strip [11] is not effective when triggers pattern are more complicated or when backdoor correlations are less dominant [38].

We point out that all these limitations we mentioned above are direct consequences of the ***passive strategies*** adopted by these work — they consistently rely on some "distinguishable behaviors" of the backdoored models that are not controlled by defenders. This work takes one step further by proposing the idea of ***active defense*** (Section 4.1) and introducing ***confusion training*** (Section 4.2) as a concrete instance, which significantly improves the performance of poison detection in practice (Section 5.2).

B On the Latent Separability of Backdoor Poison Samples

B.1 Latent Separability

One principled idea for detecting backdoor poison samples is to utilize the backdoored models' distinguishable behaviors on poison and clean populations to distinguish between these two populations themselves. **Arguably, the most popular and successful characteristic is the latent separability phenomenon** first observed by Tran et al. [39]. The basic observation is that backdoored models trained on poisoned datasets tend to learn separable latent representations for poison and clean samples — thus poison samples and clean samples can be separated via a cluster analysis in the latent representation space of the backdoored models. Commonly used backdoor samples detectors (e.g. Spectral Signature [39], Activation Clustering [3]) and some recently published state-of-the-art work (e.g. SCAn [38], SPECTRE [13]) are consistently built on this characteristic.

As a sanity check of the latent separability characteristic, we train a group of backdoored models with a diverse set of poison strategies [12, 4, 27, 1, 41, 38] on CIFAR10, and visualize (refer Apppendix C for detailed configurations) their latent representation space (of target class) in Figure 1. Although some information are lost due to the dimension reduction, we can still see that poison and clean samples consistently exhibit different distributions in all the considered cases from Figure 1b to 1g. This is consistent to observations reported by prior arts [39, 3, 38, 13] and it also accounts for the effectiveness of backdoor samples detectors built on latent space cluster analysis. On the other hand, however, we find one aspect that has not been well studied by previous work is that **the extent of the latent separation can vary a lot** (at least in the low dimensional PCA space) across different poison strategies and training configurations. For example, as can be observed in Figure 1, poison samples of BadNet (Figure 1b) exhibit stronger latent separation (less fusion) than that of Blend (Figure 1c). Meanwhile, models trained with (right-hand plot) and without (left-hand plot) data augmentation also lead to different extent of separations — models trained with data augmentation usually lead to stronger separation (e.g. Figure 1e,1f,1g) than that of models trained without data augmentation, however, the trend can also be inverse (e.g. Figure 1c). Note that, for the visualization results, we are also aware of the randomness that arise from the stochastic nature of model training. To assure readers that the variations across different cases we discussed above are not simply random noise, we repeat the same experiments for 3 times with different random seeds (see Figure 3 in Appendix C), and it turns out that all the three groups of results generated with different random seeds are qualitatively consistent. This indicates that the variations we discussed above are indeed a reflection of inherent properties of these different cases rather than random noise. As we can see in Section 5.2, the varying extent of separation directly results in varying performance of many backdoor samples detectors built on the separation characteristic — while they can work well against some poison strategies, **their performances can be less satisfactory against some others**.

B.2 Adaptive Backdoor Poisoning Attacks

Motivated by the varying extent of separation, perhaps a more fundamental question to ask is: ***Is the latent separability between poison and clean populations really an unavoidable characteristic of backdoored models?*** The implication of this question is that motivated adversaries might attempt to design adaptive backdoor attacks such that the backdoored models behave indistinguishably on both clean and poison samples. Such *potential* adaptive attacks can pose a fundamental challenge to existing backdoor samples detectors, because they completely overturn the principal assumption (that distinguishable behaviors should exist) underlying their designs.

Answers to this question depend on specific threat models and defensive settings we consider. Under a strong threat model where adversaries can fully control the training process, a series of recent work [36, 48, 9, 32, 5, 52] show that the latent representations of poison and clean samples can be made indistinguishable by explicitly encoding the indistinguishability objective into the training loss of the backdoored model.

On the other hand, as for the weaker data poisoning based threat model that we consider in this work, the problem appears to be harder, because there is still a huge gap for understanding why a deep model tend to learn separate latent representations for backdoor poison samples. One very recent work by Qi et al. [30] looks into this problem. Motivated by some heuristic insights, they propose two principled adaptive poisoning strategies that empirically can well suppress the separation characteristic in the latent space of victim models. The basic idea underlying their adaptive strategies is to introduce a set of "cover" samples (different from poison samples) which also contain backdoor triggers but are still correctly labeled to their semantic groundtruth (other than the target class). Intuitively, these "cover" samples work as regularizers that can penalize models for learning overwhelmingly strong backdoor signals in the latent representation space. In this work, we consider two adaptive strategies, namely adaptive-blend and adaptive-k, suggested in their work. Figure 1h and Figure 1i give out a straightforward sense of their "adaptiveness". In Section 5.2, we also see that **performance of existing detectors can degrade catastrophically against these adaptive backdoor poisoning attacks**.

C Visualization of Latent Representation Space

In this section, we present our full visualization results of latent representation space on CIFAR10, for a set of backdoored models attacked by different poison strategies.

Procedure of The Visualization. For each poison strategy, we first construct the resulting poisoned dataset following the configuration in Appendix E.3. Then, we train a backdoored model on this poisoned dataset with a standard training (see Appendix E.2). Next, we take out all samples that are **labelled as the target class** from the poisoned set and use the trained backdoored model to project them into the latent representation space. Finally, for visualization purpose, we project these latent representations to low dimensional visual planes.

PCA [28] Projection. In Figure 1 that we present in the main text, we use PCA [28] to project these latent representations into the top-2 principal directions. In Figure 3, we present the full results produced by PCA projection, where the experiments are repeated for three times.

t-SNE [43] Projection. Considering that PCA may not well capture the local similarity structure, alternatively, we also use t-SNE [43], which is designed specifically for preserving local structure and arguably one of the best dimensionality reduction technique. In Figure 4, we present the visualization results with two-dimensional t-SNE projection.

Oracle Projection. Note that, both PCA and t-SNE are unsupervised methods for discovering structures. The low dimension projection generated by these two methods may still not faithfully reflect the real extent of separation between poison and clean samples — these unsupervised dimensionality reduction methods may only keep other irrelevant structures and throw the information about the separability. Thus, following the same practice in Qi et al. [30], we also incorporate oracle knowledge (in the experimental simulation, we know which one is poison and which one is not) about poison and clean samples to improve the visualization. Specifically, we model the problem of poison

samples detection as a binary classification problem, where poison samples form the positive class and clean samples form the negative class. Then, we use Support Vector Machine (SVM) to fit these samples. Intuitively, the fitted hyperplane by SVM is the approximately optimal linear boundary between the two classes. Finally, we compute the (signed) distance between each point and the fitted hyperplane, and plot the distance histogram in Figure 5. Conceptually, if the poison and clean populations are very well separated in the latent representation space, then the two groups should stay in the opposite side of the SVM hyperplane, and thus the distance histograms of these two different groups will also be far well separated. This is true for the naive BadNet (Figure 5a). However, **for adaptive poison strategies, one can see (Figure 5g,5h) that even with oracle knowledge, it is hard to separate the two groups with a linear boundary**.

D Technical Details of Confusion Training

In Section 4.2, we illustrate the key idea of confusion training. In this section, we will first introduce several useful engineering techniques (Appendix D.1) that are used for implementing our confusion training pipeline. Then we will describe the detailed algorithms (Appendix D.2) for confusion training that we use in our implementation.

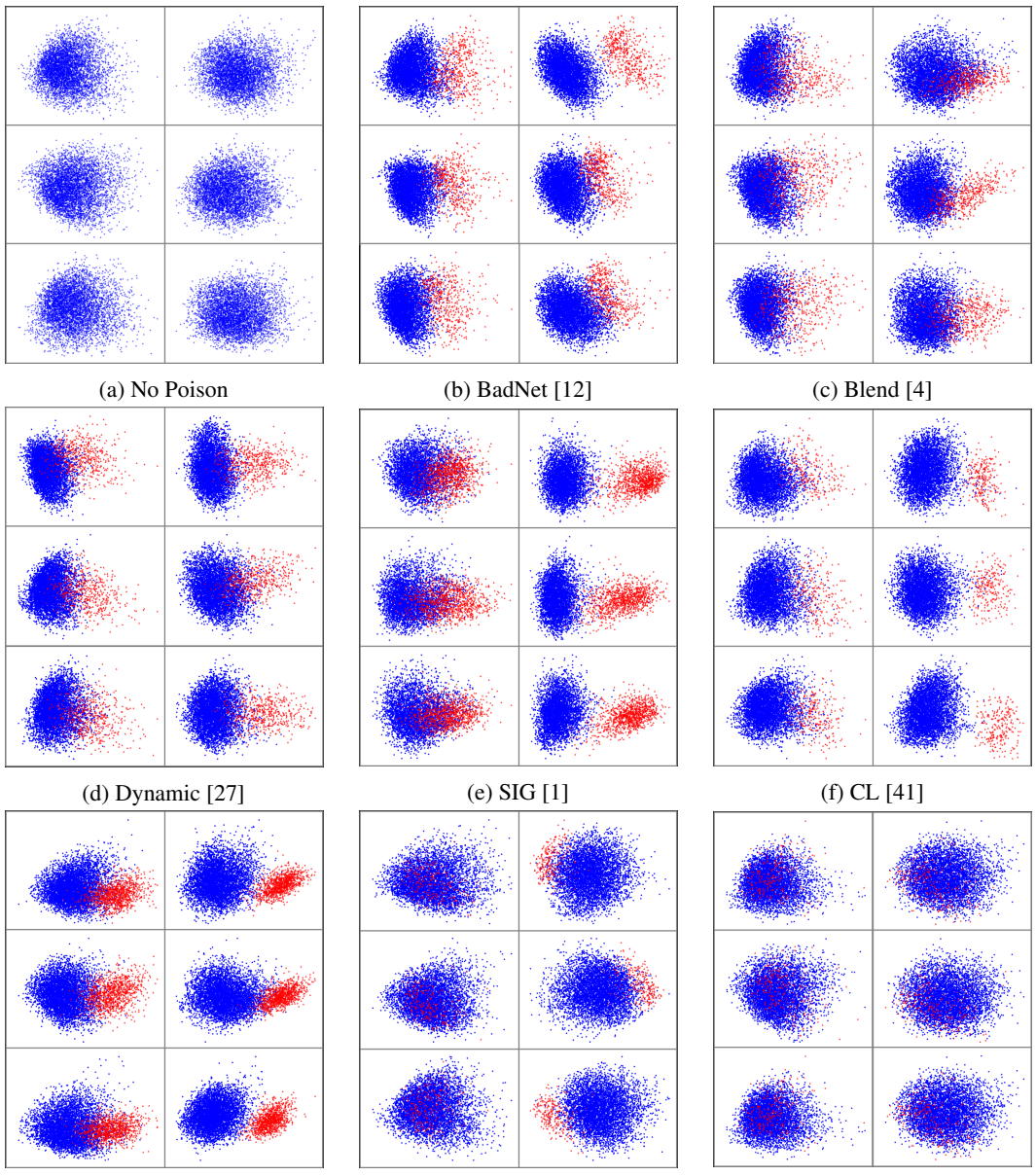
D.1 Practical Techniques for Confusion Training

Mitigate Overfitting. In practice, due to the strong expressivity of deep models, if we use a static confusion set, the deep model may still overfit both the confusion set and the poisoned training set as long as the training time is sufficiently long. This deviates our expectation that the inference model should separate backdoor poison samples and clean samples by only fitting the poison samples. To avoid such situation, in our implementation, we maintain a **dynamic confusion set** during confusion training. For every training batch, we will reassign a new set of random labels to samples from the confusion set. This leads to constantly changing correlations for clean semantics, and effectively prevents overfitting — because it can not even be fitted!

Iterative Poison Distillation. In our implementation, we will iteratively run confusion training for multiple rounds. By our design, after a round of confusion training, most clean samples in the poisoned set should have high loss values due to the disruption of confusion set, while most poison samples should concentrate in the low loss region. Thus, after each round of confusion training, we can abandon the part of training samples with highest loss values and reasonably expect that most poison samples are still be kept in the rest part, then, the rest part will be further used for the next round. This is analogous to an iterative distillation process. By iteratively removing impurities (clean samples) separated by distillation, density of the desired substances (backdoor poison samples) will be increasingly higher. Intuitively, such a process will keep amplifying the backdoor correlations meanwhile weakening the benign correlations in the distilled poisoned training set. Consequently, quality of the resulting inference model will also be keep improved.

Conditional Random Labeling. For every sample in the confusion set, we always rule out their semantic ground truth during random labeling. This can make the inference model even worse than random classifiers on clean samples, and helps to reduce false positive rate for identify backdoor poison samples. Moreover, in our implementation, we do not really perform random labeling. For each training batch with the sequential batch id b and the clean batch (X_b, Y_b), we just shift its labels to $(Y_b + b)\%C$. If $b\%C$ equals to zero, we shift the labels to $(Y_b + b + 1)\%C$ instead. We view this as a simple identical way for implementing the conditional random labeling.

Cluster Analysis for Identifying The Target Class. Given the final inference model generated by confusion training, a naive way for eliminating backdoor poison samples is to directly remove all samples whose labels are consistent with the inference model's predictions (i.e. all fitted training samples). False positives (clean samples that are mistakenly identified as poison samples) can arise, because usually there are still a small amount of clean samples (usually several percent) being fitted by the inference model despite the disruption of confusion set. Following the practice of previous work [3, 38], a simple technique for reducing false positives is to only scan samples from suspicious classes that are likely to be target classes. Since the confusion training actively separates the poison and clean populations, just like previous work, we can still identify suspicious classes by applying cluster analysis on the latent representation space of the inference model. Recall that, although the naive inference procedure we mentioned at the beginning may have higher false positive rate, it can leave out a set of samples that (almost) only contain clean samples. This gives us rich information about the distribution of clean samples' latent representations generated by the inference model. Thus,

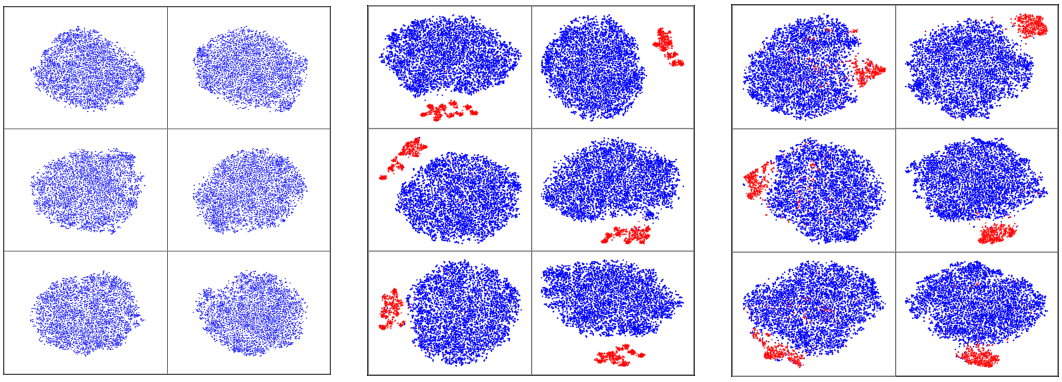


(a) No Poison (b) BadNet [12] (c) Blend [4]

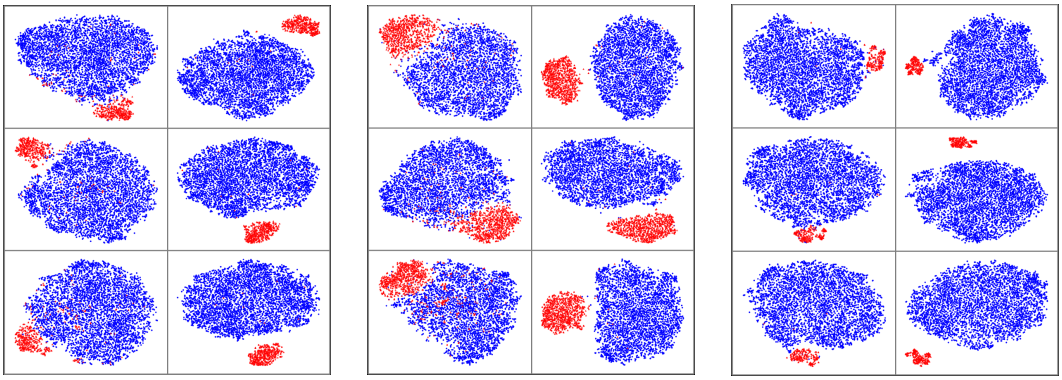
(d) Dynamic [27] (e) SIG [1] (f) CL [41]

(g) TaCT [38] (h) Adaptive-Blend [30] (i) Adaptive-K [30]

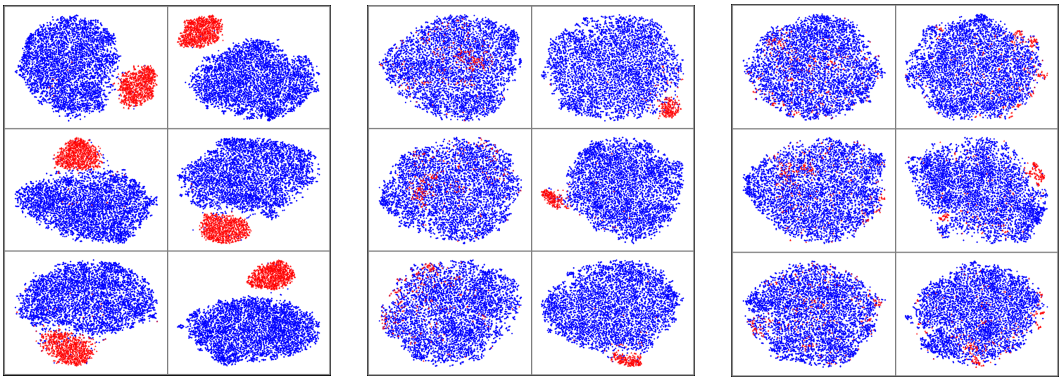
Figure 3: [PCA] Visualization of the latent representation space (CIFAR10), with base models trained with three different random seeds respectively (corresponding to three rows in each subfigure). This figure servers as a supplement to Figure 1. Each row in every subfigure corresponds to a different random seed, and the first row of each subfigure is exactly the one presented in Figure 1. As shown, the three different rows in each subfigure exhibit qualitatively consistent results. This assures readers that the variations across different configurations (i.e. different poison strategies, training configurations) are not simply random noise, instead, they are induced by the inherent properties of these different settings. Red points correspond to poison samples and blue points correspond to clean samples.



(a) No Poison (b) BadNet [12] (c) Blend [4]

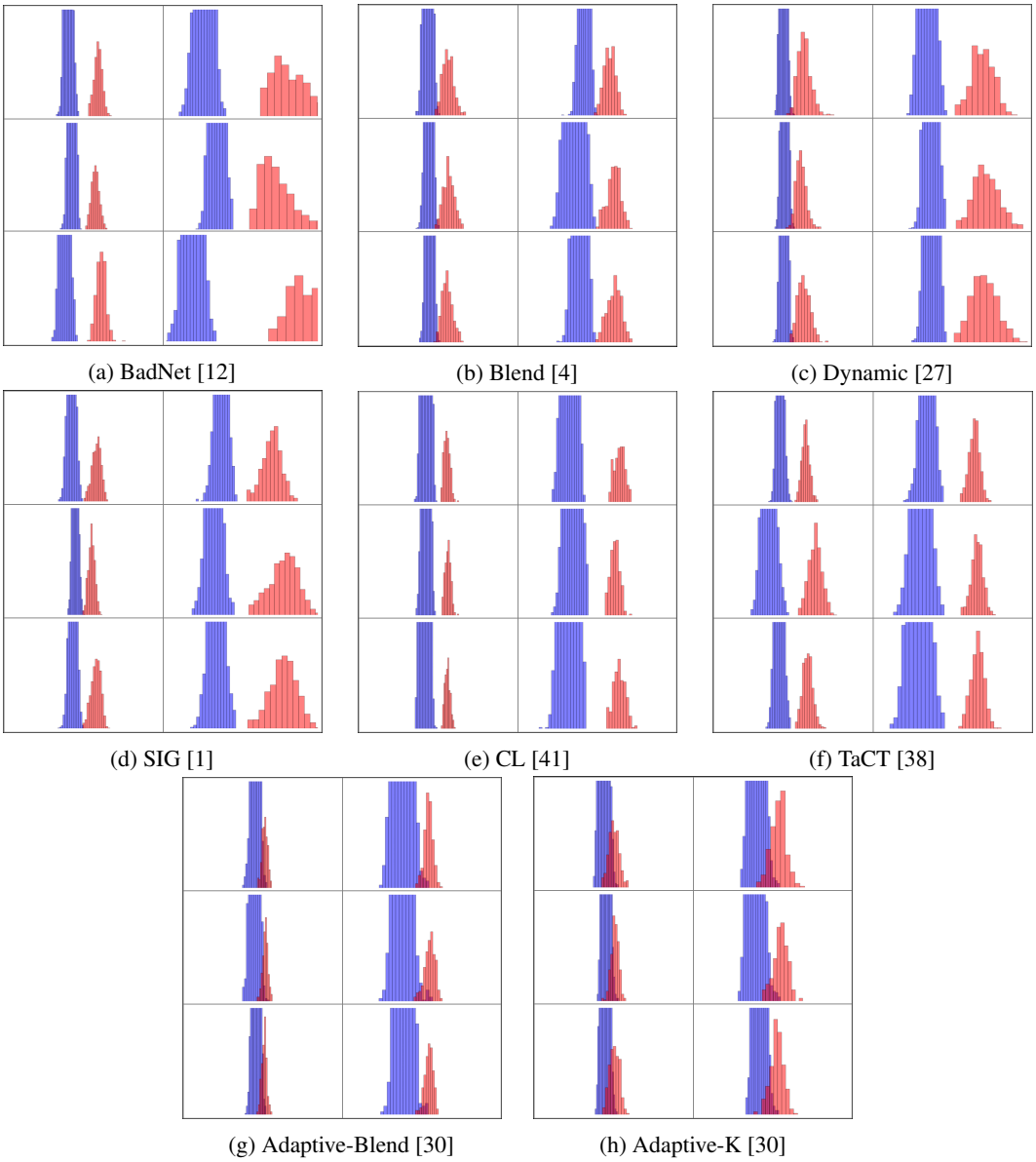


(d) Dynamic [27] (e) SIG [1] (f) CL [41]



(g) TaCT [38] (h) Adaptive-Blend [30] (i) Adaptive-K [30]

Figure 4: [t-SNE] Visualization of the latent representation space (CIFAR10). Configurations are the same to Figure 3, except that two-dimensional t-SNE projection is used instead of PCA.



(a) BadNet [12] (b) Blend [4] (c) Dynamic [27]

(d) SIG [1] (e) CL [41] (f) TaCT [38]

(g) Adaptive-Blend [30] (h) Adaptive-K [30]

Figure 5: [Oracle] Visualization of the latent representation space (CIFAR10). Each plot is a histogram, where the X-coordinate is the **signed** distance to the oracle hyperplane (approximated by SVM) and Y-coordinate denotes the number of samples for the corresponding distance level. The X-coordinate range of all histograms is [-3,3]. For BadNet, Blend and Dynamic, the Y-coordinate range is [0, 150]. For CL, Adaptive-Blend and Adaptive-K, the Y-coordinate range is [0, 75]. For SIG and TaCT, the Y-coordinate range is [0, 250]. Bars that exceed the maximum Y range are directly cutoff at the maximum height, since the information is not important. Similar to the point plots in Figure 3,4, red bars correspond to poison samples and blue bars correspond to clean samples.

we can simply identify suspicious classes via checking whether there are obvious outliers that deviate the clean distribution. Specifically, we encode our knowledge about clean distribution into a set of equivalence constraints (the knowledge that a set of samples must be in the same cluster) and use the semi-supervised EM algorithm proposed by Shental et al. [35] to compute Gaussian Mixture Models (GMM). We identify a class as a potential target class, if the two-clusters GMM explanation has significantly higher likelihood than that of single-mode Gaussian model directly estimated from the clean cluster.

D.2 Algorithms of Confusion Training

We formulate our algorithms for the confusion training pipeline. Overall, there are two main steps. With Algorithm 2, we run iterative confusion training to concentrate poison samples in a condensed poisoned set D_S and use it to generate the final inference model $F(\cdot, \theta_S)$ with another run of confusion training (iteration S). Then, with this inference model, Algorithm 3 will identify potential poison samples with a label-only inference by scanning samples from suspicious classes. As discussed in Appendix D.1, in the stage of poison identification, $\mathcal{S} = \{1, \ldots, C\}$ corresponds to the naive label-only inference, while a better $\mathcal{S}$ from cluster analysis can further help to reduce the false positive rate.

Algorithm 2: Inference Model Generation

Input: A randomly initialized network network $F(\cdot;\theta)$, the loss function L, the number of classes C, the dataset $D_{train} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ to be cleansed, a small reserved clean set $D_{clean} = \{\hat{\mathbf{x}}_j, y_j\}_{j=1}^{M}$, confusion training strength λ, iteration steps S, number of confusion batches I, distillation rates $\{r_k\}_{k=1}^{S}$

Output: The final inference model θ_S

$D_1 \leftarrow D_{train}$;
for $s = 1, ..., S$ **do**
  `//standard pre-training`
  $\theta_s \leftarrow$ start from the network $F(\cdot;\theta)$ and use D_s to perform model training with a vanilla optimizer;
  `//confusion training`
  rounder $\leftarrow 1$;
  for $i = 1, ..., I$ **do**
    $(\mathbf{X}_i, \mathbf{Y}_i) \leftarrow$ a random batch sampled from D_s;
    $(\hat{\mathbf{X}}_i, \hat{\mathbf{Y}}_i) \leftarrow$ a random batch sampled from D_{clean};
    `//make sure the shifted labels are always wrong`
    if *rounder*$\%C == 0$ **then**
      rounder $\leftarrow$ rounder $+ 1$
    `//dynamically shift the labels of clean batch`
    $\hat{\mathbf{Y}}_i \leftarrow (\hat{\mathbf{Y}}_i + \text{rounder})\%C$;
    loss $\leftarrow \Big(L(F(\mathbf{X}_i;\theta_s), Y_i) + \lambda \cdot L(F(\hat{\mathbf{X}}_i;\theta_s), \hat{Y}_i)\Big) \Big/ (1+\lambda)$;
    $\theta_s \leftarrow$ optimize $F(\cdot;\theta_s)$ on the batch loss;
    rounder $\leftarrow$ rounder+1;
  $\theta \leftarrow \theta_s$
  `//poison distillation`
  $D_{s+1} \leftarrow$ sort samples in D_{train} according to their loss values $L(F(\mathbf{x};\theta_s), y)$ and select the first $\lfloor r_s \cdot N \rfloor$ samples with least loss values

E Detailed Configurations of Our Experiments

E.1 Computation Resources and Datasets

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

Algorithm 3: Poison Samples Elimination

Input: Inference Model $F(\cdot;\theta)$, the dataset $D_{train} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ to be cleansed, the set of suspicious classes $\mathcal{S}$

Output: The cleansed dataset $D_{cleansed}$

$D_{cleansed} \leftarrow D_{train}$;

for $i = 1, ..., N$ **do**

```
//if a sample is not from suspicious classes, skip it
if y_i is not in S then
    continue;
//label-only inference for identifying poison samples
```

$\tilde{y}_i \leftarrow F(\mathbf{x}_i;\theta)$

if $\tilde{y}_i == y_i$ **then**

$D_{cleansed} \leftarrow$ remove $(\mathbf{x}_i, y_i)$ from $D_{cleansed}$

Amounts of Computation. For baseline evaluations, two models are trained (w/ and w/o data augmentation) on the cleansed dataset for each (attack, defense) pair. Thus, for each run, $6 \times 9 \times 2 = 108$ modes are trained for CIFAR10 and $6 \times 8 \times 2 = 96$ models are trained for GTSRB. Since all experiments are repeated for 3 times with different random seeds, in total, 612 models are trained for baseline evaluations. With a single GPU (GeForce RTX 2080 Ti), roughly 50 minutes are needed for training our model on CIFAR10 (200 epochs), and 15 minutes are needed for GTSRB (100 epochs) — about **342 GPU hours in total**. For confusion training, we first launch the defense pipeline for each attack. With a single GPU, training an inference model for the defense roughly takes 50 minutes for CIFAR10 and 35 minutes for GTSRB. Similarly, after datasets are cleansed by the confusion training pipeline, we will also train two base models (w/ and w/o data augmentation) on the cleansed set for each run. The costs for base models are the same to that of baseline defenses. In total, to evaluate the confusion training, **about** 67.5 **GPU hours are needed for CIFAR10 and** 26 **GPU hours for GTSRB**.

Datasets. We use two public datasets in our evaluations — CIFAR10 [17] and GTSRB [37]. CIFAR10 is under MIT license, while GTSRB is licensed under the CC0 1.0 Universal (CC0 1.0) Public Domain Dedication. For each dataset, we first follow the default training/test set split by Torchvision [26]. Recall that our defender is assumed to have a small reserved clean set at hand (as defined in Section 3.2). To implement this setting, for both datasets, we further randomly pick 2000 samples from the test split to simulate the reserved clean set, and leave the rest part of the test split for evaluation.

E.2 Detailed Configurations for Training Backdoored Models

For poisoned datasets and cleansed datasets, we will train models on them, to evaluate attack success rate (ASR) and clean accuracy. To simulate the setting that a victim is training his/her model with on a poisoned dataset, we use the vanilla training procedures for training all base models.

Specifically, for all base models, we use ResNet20 [14] as the architecture. SGD with a momentum of 0.9, a weight decay of 10^{-4}, and a batch size of 128, is used for optimization. Initially, we set the learning rate to 0.1. On CIFAR10, we follow the standard 200 epochs stochastic gradient descent procedure, and the learning rate will be multiplied by a factor of 0.1 at the epochs of 100 and 150. On GTSRB we use 100 epochs of training, and the learning rate is multiplied by 0.1 at the epochs of 40 and 80.

E.3 Detailed Configurations for Baseline Attacks

We consider eight different backdoor poisoning attacks in our evaluation. These attacks correspond to a diverse set of poisoning strategies. BadNet [12] (Figure 1b) and Blend [4] (Figure 1c) correspond to typical all-to-one dirty-label attacks with patch-like trigger and blending based trigger respectively. Dynamic [27] (Figure 1d) corresponds to the strategy that use sample-specific triggers in place of single universal trigger. SIG [1] (Figure 1e) and CL [41] (Figure 1f) correspond to two typical

	BadNet [12]	Blend [4]	Dynamic [27]	SIG [1]	CL [41]	TaCT [38]	Adap-Blend [30]	Adap-K [30]
CIFAR10 [17]	Target Class = 0 Poison Rate = 1%	Target Class = 0 Poison Rate = 1%	Target Class = 0 Poison Rate = 1%	Target Class = 0 Poison Rate = 2%	Target Class = 0 Poison Rate = 0.5%	Target Class = 0 Poison Rate = 2% Source Class = 1 Cover Classes = 5,7 Cover Rate = 1%	Target Class = 0 Poison Rate = 0.5% Cover Rate = 0.5%	Target Class = 0 Poison Rate = 0.5% Cover Rate = 0.5%
GTSRB [37]	Target Class = 2 Poison Rate = 1%	Target Class = 2 Poison Rate = 1%	Target Class = 2 Poison Rate = 1%	Target Class = 2 Poison Rate = 2%	/	Target Class = 2 Poison Rate = 2% Source Class = 1 Cover Classes = 5,7 Cover Rate = 1%	Target Class = 2 Poison Rate = 0.5% Cover Rate = 0.5%	Target Class = 2 Poison Rate = 0.5% Cover Rate = 0.5%

Table 3: **Hyperparameters of Baseline Attacks**

clean-label poisoning strategies. TaCT [38] (Figure 1g) focus on source-specific attack. Finally, Adaptive-Blend (Figure 1h) and Adaptive-K (Figure 1i) are adaptive poisoning strategies suggested by Qi et al. [30] that can suppress the latent separability characteristic. On CIFAR10, all of the eight attacks are implemented and evaluated. On GTSRB, we omit CL since the original paper only releases poison set for CIFAR10.

During implementing these attacks, we follow the protocols and suggested default configurations of their original papers and open-source implementations. In table 3, we summarize the hyperparameters that are used for each poison strategy. Specifically, *Target Class* is the class that the backdoor trigger is correlated to, *Poison Rate* denotes the portion of training samples that are stamped with the trigger and labeled to the target class. In addition, TaCT [38] only chooses samples from a *Source Class* to construct poison samples, and they will also randomly pick a portion (*Cover Rate*) of samples from some *Cover Classes* and also plant triggers to them while keeping them still correctly labeled as their semantic labels. For Adap-Blend and Adap-K [30], they also keep a portion (*Cover Rate*) of samples planted with triggers but still correctly labeled. Besides, for all these attacks we implement, we use the same trigger patterns to those suggested by their original papers.

E.4 Detailed Configurations for Baseline Defenses

To illustrate the superiority of active defense, we compare confusion training with five prior arts that also work on the task of backdoor poison samples detection, including three typical techniques ([39, 3, 11]) that are commonly considered in the literature and two very recent state-of-the-art methods [38, 13]. As have been mentioned at the beginning of this work, these prior arts all belong to the paradigm of passive defense — all of them passively rely on certain "distinguishable behaviors" of backdoored models that they do not control. Specifically, Strip [11] assumes that backdoor models' predictions on poison samples have less entropy under intentional perturbation, while the rest four methods all rely on the latent separability characteristic that we introduce in Section 3.3.

Spectral Signature [39] removes $1.5 * \rho_p$ suspected samples from every class, while SPECTRE [13] removes $1.5 * \rho_p$ suspected samples only from the class with the highest QUE score. Activation Clustering [3] cleanses classes with silhouette scores over a threshold (0.15 for CIFAR10 and 0.25 for GTSRB). Similar is SCAn [38], whose threshold is e. Strip [11] first estimates the entropy distribution of clean samples on a validation set, selects an entropy threshold with a 10% false positive rate, and eventually removes all training samples with entropy below this threshold.

Note that, all these baseline defenses start from a base backdoored model (generated in the way we described in Appendix E.2), and identify poison samples according to the behaviors of the underlying base backdoored model.

E.5 Detailed Configurations for Confusion Training

As mentioned in Appendix D, our defense pipeline works by iteratively running confusion training for multiple rounds, and each round we will rule out some samples from the training set to increase the density of poison. In our implementation, there are 4 rounds in total. In the first round, we use the full poisoned set for confusion training. In the second round, we only take the top 50% samples with smallest loss values. In the third round, we only take top 25%. In the final round, we only take top 2000 samples from the poisoned training set. Formally, in Algorithm 2, we take $r_1 = 0.5, r_2 = 0.25, r_3 = 2000/N$.

For each round of confusion training, we initialize the model with weights generated by the last round. Then the model will be first roughly pretrained with the distilled poisoned set for 60 epochs.

Intuitively, this will set up a good priors for the backdoor correlation. Then, another 4000 iterations of confusion training will be applied to decouple the benign correlations. A large weight will be assigned to the confusion batch, such that it will dominate the learning on benign features. Specifically, we take $\lambda = 24$ for CIFAR10, and $\lambda = 14$ for GTSRB.

F Full Experiment Results

F.1 Full Results of the Three Rounds of Evaluations

Table 4-6 and Table 7-9 show our three repeated experiments on CIFAR10 and GTSRB, respectively. As shown, our Confusion Training consistently removes all (or many enough) poison training samples, and all models retrained on our cleansed sets are backdoor-free.

	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	CL [41]	SIG [1]	TaCT [38]	Adaptive-Blend [30]	Adaptive-k [30]
Without Defense	ASR	/	100.0	97.0	99.6	96.9	72.2	99.1	54.9	93.3
	Clean Accuracy	92.0	91.7	91.5	92.0	91.7	91.6	91.8	91.7	91.4
Spectral Signature [39]	Elimination Rate	/	99.0	66.6	83.8	99.6	95.8	95.1	70.0	9.6
	Sacrifice Rate	15.0	14.2	14.5	14.3	7.0	28.7	28.7	7.2	7.5
	ASR	/	1.4	96.1	94.7	4.1	4.7	69.9	25.5	98.9
	Clean Accuracy	90.6	90.8	90.9	90.9	91.9	89.7	89.6	91.6	91.2
Activation Clustering [3]	Elimination Rate	/	100.0	87.6	90.6	100.0	96.6	99.8	0.0	48.4
	Sacrifice Rate	12.3	0.0	1.9	1.5	2.1	3.8	4.1	0.0	37.2
	ASR	/	0.9	84.1	97.3	1.2	0.8	2.2	54.9	90.9
	Clean Accuracy	91.9	92.0	91.7	92.0	91.7	91.5	91.5	91.7	90.3
Strip [11]	Elimination Rate	/	100.0	53.8	99.2	100.0	88.0	77.1	1.6	5.6
	Sacrifice Rate	10.1	10.0	10.9	10.2	10.4	10.6	10.5	10.4	10.6
	ASR	/	1.1	96.5	13.4	1.5	71.9	99.0	52.1	91.5
	Clean Accuracy	91.4	91.7	91.5	92.0	91.4	91.6	91.1	91.4	91.3
SCAn [38]	Elimination Rate	/	100.0	92.4	97.6	100.0	95.6	100.0	0.0	0.0
	Sacrifice Rate	0.0	0.0	3.7	0.0	0.0	0.0	0.0	0.0	0.0
	ASR	/	1.2	27.2	13.8	1.6	3.6	0.7	56.3	93.3
	Clean Accuracy	92.0	91.5	91.6	91.7	91.8	91.5	91.6	91.6	91.4
SPECTRE [13]	Elimination Rate	/	100.0	96.2	99.8	100.0	83.0	99.8	94.0	67.6
	Sacrifice Rate	1.5	0.5	0.5	0.5	0.2	1.4	1.0	0.3	0.4
	ASR	/	1.3	7.7	8.8	1.7	50.1	3.2	2.7	17.3
	Clean Accuracy	91.8	91.7	91.7	91.3	91.9	91.1	91.7	92.0	91.4
Confusion Training (Ours)	Elimination Rate	/	100.0	100.0	99.8	100.0	100.0	100.0	100.0	88.8
	Sacrifice Rate	0.0	1.0	0.9	0.8	1.0	0.6	1.1	2.0	4.5
	ASR	/	1.2	2.0	5.8	1.1	0.0	0.6	0.5	4.0
	Clean Accuracy	92.0	91.7	91.8	91.6	91.4	91.1	91.5	91.4	91.5

Table 4: **Performance of backdoor poison samples detectors on CIFAR10.** (seed #1)

	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	CL [41]	SIG [1]	TaCT [38]	Adaptive-Blend [30]	Adaptive-k [30]
Without Defense	ASR	/	99.8	95.9	99.5	96.5	73.2	99.3	54.7	99.6
	Clean Accuracy	92.1	91.3	91.8	91.3	91.9	91.4	92.1	91.6	91.7
Spectral Signature [39]	Elimination Rate	/	88.8	69.4	72.6	100.0	93.1	88.9	56.8	12.8
	Sacrifice Rate	15.0	14.4	14.5	14.4	7.0	28.7	28.8	7.3	7.5
	ASR	/	90.4	92.9	99.1	1.1	15.1	82.5	32.0	98.7
	Clean Accuracy	90.8	91.1	90.6	90.3	91.7	90.0	90.0	91.7	91.4
Activation Clustering [3]	Elimination Rate	/	98.4	86.6	93.2	100.0	96.0	99.7	0.0	42.0
	Sacrifice Rate	12.3	1.1	6.4	5.9	0.5	0.0	1.2	0.0	37.2
	ASR	/	1.3	84.2	96.3	1.3	1.3	1.6	54.7	97.7
	Clean Accuracy	91.6	91.7	91.5	91.5	91.7	91.9	91.4	91.6	90.1
Strip [11]	Elimination Rate	/	100.0	37.2	99.8	100.0	78.7	82.6	0.8	14.0
	Sacrifice Rate	10.4	10.5	10.1	10.0	10.1	11.5	10.0	9.8	10.1
	ASR	/	1.1	97.7	8.8	1.7	72.3	97.2	56.3	99.7
	Clean Accuracy	92.0	91.4	91.9	91.2	91.4	91.6	91.1	91.4	90.8
SCAn [38]	Elimination Rate	/	100.0	93.6	98.2	100.0	96.6	100.0	0.0	2.8
	Sacrifice Rate	0.0	0.0	0.0	4.1	4.4	0.0	0.0	0.0	9.4
	ASR	/	0.9	23.8	13.5	1.7	1.5	1.3	54.7	99.1
	Clean Accuracy	92.1	91.4	92.1	91.3	91.5	91.5	91.7	91.6	88.4
SPECTRE [13]	Elimination Rate	/	100.0	98.4	99.6	100.0	94.6	100.0	94.8	0.0
	Sacrifice Rate	1.5	0.5	0.5	0.5	0.3	1.1	1.0	0.3	0.8
	ASR	/	1.2	7.6	9.8	2.0	1.1	0.1	2.3	98.0
	Clean Accuracy	91.4	91.6	91.7	92.0	91.7	91.1	91.7	91.3	91.1
Confusion Training (Ours)	Elimination Rate	/	100.0	100.0	100.0	100.0	100.0	100.0	99.6	97.2
	Sacrifice Rate	0.0	0.5	0.8	1.3	0.3	0.7	1.7	0.9	2.1
	ASR	/	1.1	1.2	4.0	2.0	0.3	0.7	1.1	3.1
	Clean Accuracy	92.1	91.8	91.4	91.4	91.6	91.5	91.5	91.7	91.2

Table 5: **Performance of backdoor poison samples detectors on CIFAR10.** (seed #2)

F.2 Effects of Data Augmentation

Table 10-13 show the effects of data augmentation on passive defenses based on the already trained models. Obviously, such passive backdoor defenses suffer from instability with or without data

	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	CL [41]	SIG [1]	TaCT [38]	Adaptive-Blend [30]	Adaptive-k [30]
Without Defense	ASR	/	100.0	97.3	99.1	99.0	73.4	99.0	52.1	99.0
	Clean Accuracy	91.9	91.2	91.4	91.6	91.5	91.4	91.5	91.8	91.7
Spectral Signature [39]	Elimination Rate	/	85.4	70.8	82.8	99.6	94.1	96.5	57.2	11.6
	Sacrifice Rate	15.0	14.3	14.4	14.3	7.0	28.7	28.6	7.3	7.5
	ASR	/	99.6	94.0	95.9	1.6	16.4	55.9	32.0	98.7
	Clean Accuracy	91.3	90.7	90.8	90.7	91.5	90.1	89.4	91.2	90.9
Activation Clustering [3]	Elimination Rate	/	98.8	87.8	92.8	100	96.6	100	0.0	38.8
	Sacrifice Rate	12.3	1.4	1.5	8.7	0.1	3.8	0.3	0.0	36.5
	ASR	/	1.4	87.1	87.5	1.8	3.2	1.2	52.3	97.7
	Clean Accuracy	91.9	91.9	91.7	91.4	91.5	91.6	91.7	91.9	90.7
Strip [11]	Elimination Rate	/	100.0	60.8	99.2	100.0	83.8	72.3	0.8	10.4
	Sacrifice Rate	9.2	10.7	9.9	10.1	10.9	10.2	10.9	10.6	10.7
	ASR	/	1.1	94.0	9.9	1.4	59.9	98.8	56.6	99.3
	Clean Accuracy	91.6	91.8	91.1	91.5	91.7	91.5	91.3	91.3	91.3
SCAn [38]	Elimination Rate	/	100.0	93.0	96.4	100.0	94.3	100.0	0.0	0.0
	Sacrifice Rate	0.0	0.0	4.1	0.0	4.2	0.0	0.0	0.0	0.0
	ASR	/	1.1	29.0	73.9	1.2	1.8	1.5	52.1	99.0
	Clean Accuracy	91.9	91.6	91.6	91.5	91.6	91.8	91.8	91.8	91.7
SPECTRE [13]	Elimination Rate	/	100.0	97.6	99.2	100.0	93.7	99.9	96.4	0.0
	Sacrifice Rate	1.5	0.5	0.5	0.5	0.3	1.1	1.0	0.3	0.8
	ASR	/	1.1	8.7	13.6	1.0	10.7	2.1	3.1	99.5
	Clean Accuracy	91.6	92.0	91.5	91.5	91.6	91.5	91.7	92.1	91.5
Confusion Training (Ours)	Elimination Rate	/	100.0	100.0	99.8	100.0	100.0	100.0	99.6	91.2
	Sacrifice Rate	0.0	0.1	1.1	0.3	0.6	0.8	0.3	0.8	1.5
	ASR	/	1.0	2.1	7.2	1.3	0.0	1.5	0.7	3.4
	Clean Accuracy	91.9	91.9	91.7	91.6	91.9	91.5	91.4	91.6	90.9

Table 6: **Performance of backdoor poison samples detectors on CIFAR10.** (seed #3)

	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	SIG [1]	TaCT [38]	Adap-Blend [30]	Adap-K [30]
Without Defense	ASR	/	100.0	95.9	100.0	65.2	100.0	48.7	72.3
	Clean Accuracy	97.9	97.7	98.1	97.9	98.1	98.4	97.6	97.7
Spectral Signature [39]	Elimination Rate	/	100.0	78.2	92.9	66.7	99.6	56.4	85.0
	Sacrifice Rate	39.1	38.3	38.6	38.4	49.7	48.2	25.5	25.3
	ASR	/	0.1	89.5	69.2	57.1	0.0	29.1	3.7
	Clean Accuracy	96.8	97.0	97.3	97.0	93.1	95.7	96.7	97.0
Activation Clustering [3]	Elimination Rate	/	99.6	85.7	80.5	0.0	100.0	0.0	0.0
	Sacrifice Rate	0.4	0.6	0.3	0.6	0.8	3.5	0.0	0.2
	ASR	/	0.1	86.5	100.0	63.3	0.0	48.7	69.9
	Clean Accuracy	98.6	97.9	97.7	97.1	98.0	97.7	97.6	96.9
Strip [11]	Elimination Rate	/	100.0	79.3	100.0	64.5	51.5	4.5	5.3
	Sacrifice Rate	8.4	8.6	8.5	8.7	8.6	10.9	10.3	9.6
	ASR	/	0.0	94.9	0.2	60.8	100.0	47.9	57.7
	Clean Accuracy	97.0	97.4	96.9	98.4	97.4	98.1	97.5	97.4
SCAn [38]	Elimination Rate	/	100.0	84.6	90.6	83.1	100.0	85.0	12.8
	Sacrifice Rate	0.5	2.4	3.1	0.5	2.5	3.5	6.4	8.0
	ASR	/	0.1	82.1	98.8	6.5	0.0	2.2	79.8
	Clean Accuracy	97.8	97.1	97.4	98.0	98.1	96.5	97.2	97.3
SPECTRE [13]	Elimination Rate	/	100.0	0.0	98.5	0.0	0.0	0.0	0.0
	Sacrifice Rate	0.2	0.5	0.2	0.5	0.2	0.2	0.2	0.1
	ASR	/	0.0	86.5	2.2	64.5	99.7	34.3	53.1
	Clean Accuracy	98.3	96.5	96.9	96.2	98.5	97.6	97.5	98.1
Confusion Training (Ours)	Elimination Rate	/	100.0	100.0	99.6	99.2	100.0	96.2	86.5
	Sacrifice Rate	0.1	0.0	1.9	0.1	0.2	1.9	0.9	2.3
	ASR	/	0.0	0.8	0.0	0.1	0.2	1.3	0.1
	Clean Accuracy	97.7	97.8	98.0	98.3	97.5	97.9	98.1	97.3

Table 7: **Performance of backdoor poison samples detectors on GTSRB.** (seed #1)

augmentation. Passive defenders do not know whether data augmentation would benefit their defenses – data augmentation is preferred (by defenders) in some cases, but the opposite in other cases.

	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	SIG [1]	TaCT [38]	Adap-Blend [30]	Adap-K [30]
Without Defense	ASR	/	100.0	96.7	100.0	63.3	99.8	46.4	60.2
	Clean Accuracy	98.5	98.0	98.2	98.1	98.2	98.3	98.0	98.0
Spectral Signature [39]	Elimination Rate	/	100.0	77.8	95.9	67.5	99.4	54.1	60.9
	Sacrifice Rate	39.1	38.3	38.6	38.4	49.6	48.2	25.5	25.4
	ASR	/	0.2	91.6	4.6	50.2	0.7	30.3	53.7
	Clean Accuracy	97.2	97.2	96.9	97.2	97.4	97.2	96.7	95.0
Activation Clustering [3]	Elimination Rate	/	100.0	80.1	84.6	85.9	100.0	0.0	0.0
	Sacrifice Rate	0.5	0.9	1.1	0.9	2.9	2.6	0.0	0.4
	ASR	/	0.1	92.5	100.0	21.2	0.0	46.4	60.8
	Clean Accuracy	97.2	97.9	98.4	96.8	98.0	96.5	98.0	96.0
Strip [11]	Elimination Rate	/	100.0	69.9	100.0	39.8	38.2	3.0	5.3
	Sacrifice Rate	9.3	8.7	9.4	7.7	10.3	11.0	9.2	10.0
	ASR	/	0.1	95.2	2.6	60.1	99.7	54.5	84.2
	Clean Accuracy	96.9	97.6	97.7	97.4	98.3	97.9	97.3	95.7
SCAn [38]	Elimination Rate	/	100.0	87.2	90.2	87.2	100.0	0.0	94.0
	Sacrifice Rate	0.4	4.6	0.0	3.5	2.9	0.0	0.4	2.9
	ASR	/	0.1	84.5	76.6	23.1	0.0	47.2	0.0
	Clean Accuracy	97.6	97.2	97.3	96.9	97.2	97.1	97.4	98.5
SPECTRE [13]	Elimination Rate	/	0.0	0.0	100.0	0.0	0.0	0.0	0.0
	Sacrifice Rate	0.1	0.2	0.2	0.5	0.1	0.1	0.1	0.2
	ASR	/	100.0	92.7	0.0	64.8	99.5	34.5	77.8
	Clean Accuracy	98.2	97.6	97.5	97.6	98.5	98.22	97.7	94.2
Confusion Training (Ours)	Elimination Rate	/	100.0	99.6	100.0	100.0	100.0	98.5	88.0
	Sacrifice Rate	0.1	1.0	0.5	1.8	0.1	2.8	0.6	2.7
	ASR	/	0.4	1.6	0.3	0.2	0.1	0.1	0.2
	Clean Accuracy	98.5	97.8	98.2	97.8	97.6	97.7	98.2	96.5

Table 8: **Performance of backdoor poison samples detectors on GTSRB.** (seed #2)

	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	SIG [1]	TaCT [38]	Adap-Blend [30]	Adap-K [30]
Without Defense	ASR	/	99.4	95.4	100.0	60.7	100.0	51.5	56.8
	Clean Accuracy	97.9	97.4	97.9	97.9	98.1	98.6	97.6	97.6
Spectral Signature [39]	Elimination Rate	/	100.0	75.2	92.1	66.5	88.7	72.9	77.4
	Sacrifice Rate	39.1	38.3	38.6	38.4	49.7	48.4	25.4	25.4
	ASR	/	0.0	94.8	63.5	65.2	99.0	33.0	0.7
	Clean Accuracy	96.3	96.5	96.2	96.2	95.4	97.6	96.8	98.0
Activation Clustering [3]	Elimination Rate	/	99.6	78.6	78.9	0.0	100.0	0.0	0.0
	Sacrifice Rate	0.6	0.6	2.5	0.4	0.7	2.0	0.3	0.3
	ASR	/	0.0	91.8	99.0	62.2	0.0	52.6	71.1
	Clean Accuracy	97.0	97.1	96.7	98.0	98.3	97.8	97.1	95.9
Strip [11]	Elimination Rate	/	100.0	72.6	100.0	27.1	23.3	3.8	3.8
	Sacrifice Rate	8.6	11.3	10.0	7.1	10.1	11.9	9.6	9.8
	ASR	/	0.1	91.6	0.8	59.0	100.0	55.1	62.5
	Clean Accuracy	97.8	96.6	97.1	97.3	97.5	97.9	96.3	96.6
SCAn [38]	Elimination Rate	/	100.0	72.2	86.1	87.0	100.0	0.0	0.0
	Sacrifice Rate	4.0	2.7	3.4	4.6	2.3	3.1	0.0	2.9
	ASR	/	0.2	94.8	100.0	0.6	0.0	51.5	62.5
	Clean Accuracy	97.8	97.1	97.6	97.3	97.7	97.5	97.6	96.6
SPECTRE [13]	Elimination Rate	/	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Sacrifice Rate	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1
	ASR	/	100.0	98.9	100.0	60.1	100.0	48.7	85.7
	Clean Accuracy	98.2	97.2	97.8	97.6	96.4	96.7	96.2	97.2
Confusion Training (Ours)	Elimination Rate	/	100.0	98.9	99.2	100.0	98.7	99.2	87.2
	Sacrifice Rate	0.0	0.0	0.5	0.2	0.4	1.9	1.0	5.4
	ASR	/	0.0	2.5	1.2	0.1	0.4	0.9	1.9
	Clean Accuracy	97.9	97.7	97.6	98.1	97.9	98.4	97.3	96.5

Table 9: **Performance of backdoor poison samples detectors on GTSRB.** (seed #3)

w/o aug : w/ aug	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	Clean Label [41]	SIG [1]	TaCT [38]	Adaptive-Blend [30]	Adaptive-k [30]
Spectral Signature [39]	Elimination Rate	/	87.1 / 81.6	66.1 / 59.9	71.3 / 77.4	82.0 / 99.7	58.5 / 94.3	74.9 / 93.5	1.9 / 61.3	7.9 / 11.2
	Sacrifice Rate	15.0 / 15.0	14.0 / 14.4	14.5 / 14.6	14.4 / 14.4	7.1 / 7.0	29.4 / 25.4	29.1 / 28.7	7.5 / 7.3	7.5 / 7.5
Activation Clustering [3]	Elimination Rate	/	99.1 / 33.0	87.3 / 0.0	61.3 / 59.9	33.2 / 100.0	0.0 / 96.4	65.0 / 99.8	0.0 / 0.0	43.1 / 0.0
	Sacrifice Rate	12.3 / 13.2	0.8 / 0.0	3.3 / 0.0	2.5 / 4.2	0.8 / 0.9	2.7 / 2.5	3.5 / 1.8	0.0 / 0.0	37.0 / 31.6
Strip [11]	Elimination Rate	/	100.0 / 100.0	42.7 / 50.6	92.6 / 99.4	98.9 / 100.0	80.1 / 80.5	68.6 / 70.7	0.5 / 0.7	2.4 / 10.0
	Sacrifice Rate	10.8 / 10.0	10.6 / 10.8	10.3 / 10.3	10.5 / 10.1	10.7 / 10.5	10.8 / 10.6	10.5 / 10.3	10.5 / 10.2	10.6 / 10.5
SCAn [38]	Elimination Rate	/	99.7 / 100.0	89.8 / 93.0	88.9 / 97.4	66.7 / 100.0	94.9 / 95.5	99.5 / 100.0	0.0 / 0.0	0.0 / 0.9
	Sacrifice Rate	0.0 / 0.0	1.2 / 0.0	5.5 / 2.6	0.0 / 1.4	2.3 / 2.9	1.6 / 2.3	0.0 / 0.0	4.7 / 0.0	2.4 / 3.1
SPECTRE [13]	Elimination Rate	/	100.0 / 100.0	95.5 / 97.4	96.8 / 99.5	100.0 / 100.0	51.2 / 90.4	99.1 / 99.8	6.8 / 95.1	16.4 / 22.5
	Sacrifice Rate	1.5 / 1.5	0.5 / 0.5	0.5 / 0.5	0.5 / 0.5	0.3 / 0.3	2.0 / 1.2	1.0 / 1.0	0.8 / 0.3	0.7 / 0.7

Table 10: **Without Augmentation vs. With Augmentation (CIFAR10).** (average)

w/o aug : w/ aug	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	Clean Label [41]	SIG [1]	TaCT [38]	Adaptive-Blend [30]	Adaptive-k [30]
Spectral Signature [39]	Elimination Rate	/	87.2 / 99.0	66.6 / 43.2	75.0 / 83.8	79.2 / 99.6	59.9 / 95.8	73.8 / 95.1	1.2 / 70.0	9.6 / 9.2
	Sacrifice Rate	15.0 / 15.0	13.4 / 14.2	14.5 / 14.7	14.4 / 14.3	7.1 / 7.0	29.4 / 28.7	29.1 / 28.7	7.5 / 7.2	7.5 / 7.5
Activation Clustering [3]	Elimination Rate	/	100.0 / 99.0	87.6 / 0.0	90.6 / 87.0	0.0 / 100.0	0.0 / 96.6	97.0 / 99.8	0.0 / 0.0	48.4 / 0.0
	Sacrifice Rate	12.3 / 13.2	0.0 / 0.0	1.9 / 0.0	1.5 / 0.2	0.0 / 2.1	4.3 / 3.8	9.4 / 4.1	0.0 / 0.0	37.2 / 32.7
Strip [11]	Elimination Rate	/	100.0 / 100.0	48.8 / 53.8	92.6 / 99.2	100.0 / 100.0	88.0 / 83.1	77.1 / 60.3	1.6 / 0.4	3.2 / 5.6
	Sacrifice Rate	10.6 / 10.4	10.7 / 10.0	9.6 / 10.9	10.5 / 10.2	10.4 / 10.5	10.6 / 10.5	10.5 / 10.3	10.4 / 10.1	10.1 / 10.6
SCAn [38]	Elimination Rate	/	99.6 / 100.0	89.6 / 92.4	89.0 / 97.6	100.0 / 100.0	95.6 / 95.6	99.8 / 100.0	0.0 / 0.0	0.0 / 0.0
	Sacrifice Rate	0.0 / 0.0	0.0 / 0.0	3.6 / 3.7	0.0 / 0.0	0.0 / 0.0	0.0 / 7.0	0.0 / 0.0	4.0 / 0.0	7.1 / 0.0
SPECTRE [13]	Elimination Rate	/	100.0 / 100.0	95.4 / 96.2	96.8 / 99.8	100.0 / 100.0	76.0 / 83.0	99.8 / 99.6	20.4 / 94.0	49.2 / 67.6
	Sacrifice Rate	1.5 / 1.5	0.6 / 0.5	0.5 / 0.5	0.5 / 0.5	0.2 / 0.2	1.5 / 1.4	1.0 / 1.0	0.7 / 0.3	0.5 / 0.4

Table 11: **Without Augmentation vs. With Augmentation (CIFAR10).** (seed #1)

w/o aug : w/ aug	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	Clean Label [41]	SIG [1]	TaCT [38]	Adaptive-Blend [30]	Adaptive-k [30]
Spectral Signature [39]	Elimination Rate	/	88.8 / 78.2	61.0 / 69.4	72.6 / 66.2	83.2 / 100.0	60.7 / 93.1	82.7 / 88.9	1.6 / 56.8	6.8 / 12.8
	Sacrifice Rate	15.0 / 15.0	14.3 / 14.4	14.5 / 14.5	14.4 / 14.5	7.1 / 7.0	29.4 / 18.7	28.9 / 28.8	7.5 / 7.3	7.5 / 7.5
Activation Clustering [3]	Elimination Rate	/	98.4 / 0.0	86.6 / 0.0	93.2 / 0.0	99.6 / 100.0	0.0 / 96.0	98.0 / 99.7	0.0 / 0.0	42.0 / 0.0
	Sacrifice Rate	12.3 / 13.1	1.1 / 0.0	6.4 / 0.0	5.9 / 3.7	2.5 / 0.5	3.9 / 0.0	1.1 / 1.2	0.0 / 0.0	37.2 / 33.5
Strip [11]	Elimination Rate	/	100.0 / 100.0	36.4 / 37.2	90.4 / 99.8	96.8 / 100.0	78.7 / 74.5	56.4 / 82.6	0.0 / 0.8	1.6 / 14.0
	Sacrifice Rate	10.9 / 10.4	10.5 / 11.2	10.7 / 10.1	10.1 / 10.0	10.5 / 10.1	11.5 / 11.0	10.2 / 10.0	10.5 / 9.8	10.6 / 10.1
SCAn [38]	Elimination Rate	/	100.0 / 100.0	90.6 / 93.6	84.6 / 98.2	0.0 / 100.0	95.1 / 96.6	99.6 / 100.0	0.0 / 0.0	0.0 / 2.8
	Sacrifice Rate	0.0 / 0.0	3.5 / 0.0	12.8 / 0.0	0.0 / 4.1	0.0 / 4.4	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 9.4
SPECTRE [13]	Elimination Rate	/	100.0 / 100.0	95.0 / 98.4	96.0 / 99.6	100.0 / 100.0	0.0 / 94.6	100.0 / 100.0	0.0 / 94.8	0.0 / 0.0
	Sacrifice Rate	1.5 / 1.5	0.5 / 0.5	0.6 / 0.5	0.5 / 0.5	0.3 / 0.3	3.1 / 1.1	1.0 / 1.0	0.8 / 0.3	0.8 / 0.8

Table 12: **Without Augmentation vs. With Augmentation (CIFAR10).** (seed #2)

w/o aug : w/ aug	(%)	No Poison	BadNet [12]	Blend [4]	Dynamic [27]	Clean Label [41]	SIG [1]	TaCT [38]	Adaptive-Blend [30]	Adaptive-k [30]
Spectral Signature [39]	Elimination Rate	/	85.4 / 67.6	70.8 / 67.0	66.2 / 82.3	83.6 / 99.6	55.0 / 94.0	68.3 / 96.5	2.8 / 57.2	7.2 / 11.6
	Sacrifice Rate	15.0 / 15.0	14.3 / 14.5	14.4 / 14.5	14.5 / 14.3	7.1 / 7.0	29.5 / 28.7	29.2 / 28.6	7.5 / 7.3	7.5 / 7.5
Activation Clustering [3]	Elimination Rate	/	98.8 / 0.0	87.8 / 0.0	0.0 / 92.8	0.0 / 100.0	0.0 / 96.6	0.0 / 100.0	0.0 / 0.0	38.8 / 0.0
	Sacrifice Rate	12.3 / 13.2	1.4 / 0.0	1.5 / 0.0	0.0 / 8.7	0.0 / 0.0	0.0 / 3.8	0.0 / 0.0	0.0 / 0.0	36.5 / 28.6
Strip [11]	Elimination Rate	/	100.0 / 100.0	43.0 / 60.8	94.8 / 99.2	100.0 / 100.0	73.5 / 83.8	72.3 / 69.1	0.0 / 0.8	2.4 / 10.4
	Sacrifice Rate	10.8 / 9.2	10.7 / 11.3	10.5 / 9.9	11.0 / 10.1	11.2 / 10.9	10.2 / 10.2	10.9 / 10.6	10.7 / 10.6	11.2 / 10.7
SCAn [38]	Elimination Rate	/	99.4 / 100.0	89.2 / 93.0	93.0 / 96.4	100.0 / 100.0	94.1 / 94.3	99.0 / 100.0	0.0 / 0.0	0.0 / 0.0
	Sacrifice Rate	0.0 / 0.0	0.0 / 0.0	0.0 / 4.1	0.0 / 0.0	7.0 / 4.2	4.7 / 0.0	0.1 / 0.0	10.2 / 0.0	0.0 / 0.0
SPECTRE [13]	Elimination Rate	/	100.0 / 100.0	96.0 / 97.6	97.6 / 99.2	100.0 / 100.0	77.7 / 93.7	97.4 / 99.9	0.0 / 96.4	0.0 / 0.0
	Sacrifice Rate	1.5 / 1.5	0.5 / 0.5	0.5 / 0.5	0.5 / 0.5	0.3 / 0.3	1.5 / 1.1	1.1 / 1.0	0.8 / 0.3	0.8 / 0.8

Table 13: **Without Augmentation vs. With Augmentation (CIFAR10).** (seed #3)