ASSET: Robust Backdoor Data Detection Across a Multiplicity of Deep Learning Paradigms

Minzhou Pan and Yi Zeng, *Virginia Tech;* Lingjuan Lyu, *Sony AI;*
Xue Lin, *Northeastern University;* Ruoxi Jia, *Virginia Tech*

https://www.usenix.org/conference/usenixsecurity23/presentation/pan

This paper is included in the Proceedings of the
32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the
32nd USENIX Security Symposium
is sponsored by USENIX.

ASSET: Robust Backdoor Data Detection Across a Multiplicity of Deep Learning Paradigms

Minzhou Pan*[1], Yi Zeng*[1], Lingjuan Lyu[2], Xue Lin[3] and Ruoxi Jia[1]

[1]Virginia Tech, Blacksburg, VA 24061, USA
[2]Sony AI, Tokyo, 108-0075, Japan
[3]Northeastern University, Boston, MA 02115, USA

Abstract

Backdoor data detection is traditionally studied in an end-to-end supervised learning (SL) setting. However, recent years have seen the proliferating adoption of self-supervised learning (SSL) and transfer learning (TL), due to their lesser need for labeled data. Successful backdoor attacks have also been demonstrated in these new settings. However, we lack a thorough understanding of the applicability of existing detection methods across a variety of learning settings. By evaluating 56 attack settings, we show that the performance of most existing detection methods varies significantly across different attacks and poison ratios, and all fail on the state-of-the-art clean-label backdoor attack which only manipulates a few training data's features with imperceptible noise without changing labels. In addition, existing methods either become inapplicable or suffer large performance losses when applied to SSL and TL. We propose a new detection method called Active Separation via Offset (ASSET), which actively induces different model behaviors between the backdoor and clean samples to promote their separation. We also provide procedures to adaptively select the number of suspicious points to remove. In the end-to-end SL setting, ASSET is superior to existing methods in terms of consistency of defensive performance across different attacks and robustness to changes in poison ratios; in particular, it is the only method that can detect the state-of-the-art clean-label attack. Moreover, ASSET's average detection rates are higher than the best existing methods in SSL and TL, respectively, by 69.3% and 33.2%, thus providing the first practical backdoor defense for these emerging DL settings.

1 Introduction

Deployment of deep learning (DL) in critical services and infrastructures calls for special emphasis on security, given its susceptibility to erroneous predictions in the presence of attacks [1–3]. Specifically, data-poisoning-based *backdoor attacks* - where attackers manipulate the training data to force certain outputs during testing - pose a significant threat. Successful attacks have been demonstrated on various computer vision tasks and beyond [4]. This paper focuses on the problem of *detecting the poisoned samples within a training set*. An effective detection strategy allows one to mitigate the risk of backdoors by removing suspicious samples from training.

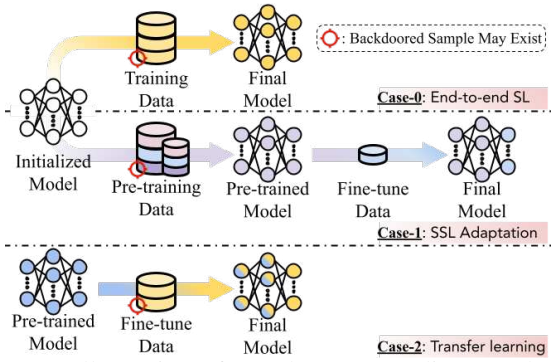

Figure 1: Illustration of popular DL paradigms and corresponding threat models. **Case-0**: traditional end-to-end SL, where one trains a model from scratch. **Case-1**: SSL adaptation, where one first pre-trains a model via SSL using unlabeled pre-training data and then linearly adapts to a small amount of labeled data to obtain the final model. **Case-2**: TL, where one starts with an existing pre-trained model and fine-tunes it. Existing work has demonstrated successful attacks in all three cases under the threat models where datasets marked with red circle indicates poisons. Yet, none of the existing backdoor detection methods is evaluated in all three cases.

Poisoned samples can be regarded as outliers in a training set. However, unlike *arbitrary* outliers considered in the classical outlier detection and robust statistics literature, poisoned samples are special outliers that induce *specific model behaviors*, e.g., misleading the model to predict some target class(es). Hence, recent works on backdoor detection primarily leverage the model trained on the poisoned dataset (backdoored model hereinafter) or information cached during training to help discover poisoned samples [5–11]. For instance, most of the prior work starts by extracting the backdoored model's output [6], intermediate activation patterns [7–10], gradient [11] for each sample, and then separate poisons from clean samples based on the extracted information.

While taking advantage of the information collected from

*Y. Zeng and M. Pan contributed equally. Correspond Y. Zeng or R. Jia.

the downstream learning process provides a clear path to enhancing backdoor detection performance, it also raises the question: *Can these detection methods maintain their performance across different DL settings?* Particularly, existing detection methods are exclusively evaluated in only one learning setting—*end-to-end supervised learning* (SL), where a labeled poisoned dataset is used to train a model *from scratch*. On the other hand, new learning paradigms are increasingly adopted and have demonstrated state-of-the-art prediction performance with reduced annotation costs and computational burden [12–15]. The two most representative and popular paradigms are *self-supervised learning (SSL) adaptation* and *transfer learning (TL)*, as illustrated in Figure 1.

In SSL adaptation, one pre-trains a model on large *unlabeled* data (e.g., through contrastive learning [16–18] or masked autoencoder (MAE) [13]) and then fine-tunes only the last layer using *labeled* data from a specific downstream task. Recent work [19–21] has shown that an attacker can poison the unlabeled dataset to implant backdoors without any control over downstream fine-tuning processes. Thus, it is natural to ask: *Can we detect the poisoned samples within an unlabeled dataset using existing methods?* In TL, one starts with an existing pre-trained model and fine-tunes all layers of the model or just the last layer with labeled data. Despite the importance of TL in practice [22], we lack an understanding of backdoor detection in this setting: *Can we detect the poisoned samples when they are used for fine-tuning an existing model instead of training it from scratch?*

Our first contribution is a comprehensive evaluation of existing detection methods across different DL paradigms. The key findings are summarized as follows.

- **(Case-0) End-to-end SL:** Despite the efficiency demonstrated by prior detection efforts in specific settings, the consistency of efficacy varies a lot across different attacks or poison ratios. In particular, all fail to detect the state-of-the-art clean-label backdoor attack[1] [2] and underperform in the very low or very high poison ratio setting (e.g., 0.05% or 20%).
- **(Case-1) SSL adaptation:** There are no existing methods dedicated to detecting unlabeled poisoned samples in the SSL setting. Yet, some of the existing methods can be adapted to the SSL. For instance, those methods attempting to separate the poisoned samples from clean in the embedding space can employ an embedder learned from unlabeled data to generate the embedding for each sample [2]. However, the performance of these methods after adaptation is limited (e.g., their average detection rate over different attacks all falls below 26%).
- **Case-2 TL:** While prior literature omitted TL in their evaluation, the detection methods can all be applied to it. However, the methods based on embeddings suffer a significant performance loss compared to the end-to-end SL setting because the poisoned samples are less distinguishable from clean ones in a fine-tuned embedding space than a trained-from-scratch one.

[1] Clean-label attacks refer to those where the poisoned samples appear to be correctly labeled to a human inspector.

[2] We will elaborate on the adaptation techniques in Section 5.1.

	Spectral [7]	Spectre [8]	Beatrix [9]	AC [10]	ABL [11]	Strip [6]	CT [23]	Ours
Applicable to Labeled Data	✓	✓	✓	✓	✓	✓	✓	✓
Applicable to Unlabeled Data	○	○	○	○	○	○	×	✓
Robust to Diff. Triggers	×	×	×	×	×	×	×	✓
Robust to Diff. Poison Ratios	×	×	×	×	×	×	×	✓

Table 1: A summary and comparison of representative works in the detection of backdoored samples. ○ denotes partially satisfactory (i.e., requiring additional adaptation).

The limitations of existing methods per our evaluation are summarized in Table 1. Overall, there still lacks a detection method that is effective across different learning paradigms.

Our second contribution is the development of a robust, generic approach to backdoor detection that applies to the three representative learning paradigms discussed above. Like most existing literature [6, 8, 9], our approach also assumes that the defender has an extra set of clean samples (referred to as a *base set* hereinafter) with a size much smaller compared to the training set. In practice, these clean samples can be obtained through manual inspection or automatic screening [24]. However, unlike the previous works, we do not require the base set to be labeled.

The key idea of our approach is to induce different model behaviors between poisoned samples and clean ones. To achieve this, we design a two-step optimization process: we first *minimize* some loss on the clean base set; then, we attempt to *offset* the effect of the first minimization on the clean distribution by *maximizing* the same loss on the entire training set including both clean and poisoned samples. The outcome of this two-step process is a model which returns high loss for poisoned samples and low loss for clean ones. Hence, we can decide whether a sample is poisoned or clean based on the corresponding loss value.

We found that the two-step optimization-based offset idea achieves strong detection performance except in settings where the poison ratio is low, or the learning of the poisoned samples happens slowly—at roughly the same speed as learning of clean samples. As we will explicate later in the paper, in these cases, the effect of the second maximization significantly outweighs that of the first minimization; as a result, both poisoned and clean samples achieve large losses and become inseparable.

To tackle the challenge, we propose a strengthened technique that involves *two nested offset procedures*, and the inner offset reinforces the outer one. Specifically, we use the inner offset procedure to identify the points most likely to be poisoned and mark them as suspicious; the outer offset procedure still minimizes some loss on the clean base set, but the maximization will now be performed on the points marked to be suspicious by the inner offset, instead of the entire poisoned

dataset. As the proportion of clean samples within the suspicious set is much smaller than that within the entire poisoned set, the small loss of clean samples obtained from the first minimization would be impacted much less by the second maximization. This nested design effectively improves the separability between clean and poisoned samples.

Our third contribution is the provision of techniques that can adaptively set the loss threshold to discern poisoned samples. Some of the prior works [7, 8] assume the knowledge of poison ratio and mark a fixed number of samples as poisoned ones based on their respective criteria. Moreover, the poisoned and clean samples often do not have a clear separation based on their criteria (see examples in Figure 5); as a result, their detection performance is very sensitive to the estimated poison ratio. We argue that in practice, it is challenging to have an accurate estimate of the poison ratio. Hence, it is preferable to adapt detection to the data characteristics rather than relying on a fixed estimate. Herein, we design two adaptive thresholding techniques tailored to specific requirements imposed by inner and outer offset procedures (i.e., prioritizing precision vs. prioritizing true positive rate).

We conduct extensive experiments in comparison with seven representative or state-of-art backdoor data detection methods over 56 different attack settings across various DL paradigms and show that our proposed method, ASSET, is the only one that can provide reliable detection consistently across all the evaluated settings. This work is also the first practical backdoor detection for the SSL and the TL settings[3].

2 Background & Related Work

End-to-end supervised learning & transfer learning. The objective of end-to-end SL is to train a classifier $f(\cdot|\theta): \mathcal{X} \to [k]$, which predicts the label $y \in [k]$ of an input $x \in \mathcal{X}$. θ denotes the parameters of the classifier $f(\cdot|\theta)$. The standard end-to-end SL (**Case-0**) consists of two stages: training and testing. In the training stage, a learning algorithm is provided with a set of training data, $D = \{(x_i, y_i)\}_{i=1}^{N}$, consisting of examples from k classes. Then, the learning algorithm seeks the model parameters, θ, that minimize the empirical risk:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}\left(f\left(x_i|\theta\right), y_i\right). \quad (1)$$

When $f(\cdot|\theta)$ is a deep neural network, the corresponding empirical risk is a non-convex function of θ, and finding a global minimum is generally impossible. Hence, the standard practice is to look for a local minimum. Algorithmically, the model is initialized with random parameters and updated iteratively via stochastic gradient descent [25]. In the test stage, the trained model $f(\cdot|\theta^*)$ takes input test examples and serves up predictions. TL (**Case-2**) shares the same optimization goal as the end-to-end SL. However, TL initializes the optimization with a pre-trained backbone model instead of random parameters. Within the scope of this paper, we consider two of the most popular TL schemes: (1) FT-all: the entire pre-trained model gets updated during training (e.g., [13, 14, 26]); (2) FT-last (or linear adaptation): only the last fully-connected layer is updated (e.g., [15, 16, 27]). In the context of TL, we will refer to solving the optimization (1) as *fine-tuning* and D as the *fine-tuning data*.

[3] Open-source: https://github.com/ruoxi-jia-group/ASSET

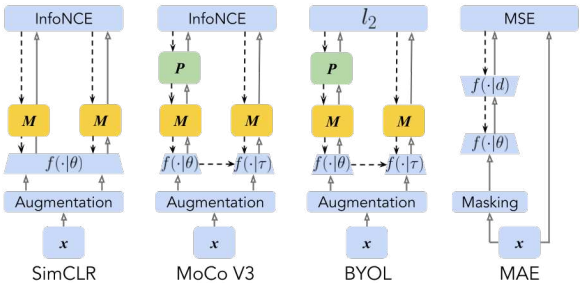

Figure 2: Illustration of representative SSL methods: SimCLR [12], MoCo V3 [12], BYOL [18], and the Masked Auto-Encoder [13]. The solid gray arrow indicates forward propagation, and the dashed black arrow indicates backpropagation.

Self-supervised learning. SSL usually consists of two phases: pretext training and fine-tuning. Pretext training aims to train an encoder $f(\cdot|\theta): \mathcal{X} \to \mathcal{Z}$ that can map the input $x \in \mathcal{X}$ into the embedding $z \in \mathcal{Z}$. θ denotes the parameters of the encoder $f(\cdot|\theta)$. This paper focuses primarily on two of the most recent SSL schemes: contrastive learning and masked auto-encoder (MAE). Their training processes are illustrated in Figure 2, where M is a multi-layer perceptron (MLP) used to reduce the dimension of features, and P is a predictor. The fundamental idea of contrastive learning, e.g., SimCLR [12], MoCo V3 [17], and BYOL [18], is to learn an encoder by bringing the embeddings corresponding to the augmentations of the same image (a.k.a. positive pairs) closer and distancing its embeddings from other images (a.k.a. negative pairs). All three methods pre-train $f(\cdot|\theta)$, M and P (if applicable) on large amounts of unlabeled data, and differ in how they generate positive and negative pairs and in the loss functions they use for training. We refer interested readers to [28] for more details. By contrast, the recently proposed SSL method, MAE [13], trains the encoder $f(\cdot|\theta)$ by masking a portion of pixels in an image x (the masked image is denoted by x') and then using $f(x'|\theta)$ with a decoder $d(\cdot)$ to restore x. For all the aforementioned SSL methods, after the pretext training, the acquired encoder parameters θ^* will be adapted to a downstream task similarly to TL using the fine-tuning data.

Backdoor attacks. Backdoor attacks have been extensively studied in the end-to-end SL setting and can be categorized into dirty-label and clean-label attacks. Dirty-label backdoor attacks manipulate both label and feature of a sample. These attacks have developed from using a sample-independent visible pattern as the trigger [1, 29–31] to more stealthy and powerful attacks with sample-specific [32] or visually imperceptible triggers [33–38]. Clean-label backdoor attacks ensure that the manipulated features are semantically consistent with corresponding labels. Existing attacks in this category range from inserting arbitrary triggers [39–41] to optimized trig-

gers [2]. Most of the above backdoor attacks can be easily adapted to TL settings without modifications. There are also backdoor attacks specifically designed for TL settings, e.g., the hidden trigger backdoor attack [40].

With the thriving development of SSL, especially contrastive learning (e.g., SimCLR [12, 16], MoCo [17, 42, 43], BYOL [18]) and the MAE [13], backdoor attacks targeting SSL have also been explored. Recent work mainly applies existing dirty-label backdoor triggers studied in SL to the targeted category of samples [19, 20]. However, attacks' efficacy are limited (ASR below 10% on CIFAR-10 even with an in-class poison ratio set to 50%, as shown in our experiment, Section 5.3). A recent attack [21] exploits the "representation invariance" property of contrastive learning and instantiate a symmetric trigger via manipulation in the frequency domain, achieving much higher ASR with a lower poison ratio (e.g., in-class poison ratio of 10%).

Backdoor sample detection. Note that no existing backdoored sample detection methods have been considered nor evaluated over cases other than **Case-0**. In particular, there is no practical defense under the SSL, and the study in TL is overlooked. Many of the existing works identify poisoned samples by examining their difference from clean ones in the embedding space, such as using singular value decomposition (SVD) [7, 8], Gram matrix [9], K-Nearest-Neighbors [44], and feature decomposition [5]. In addition to embeddings, intermediate neural activation [10, 45] and gradients [46, 47] extracted from samples can also be adopted for backdoored sample detection. Past work has also examined other differentiating properties of backdoor samples, such as trigger's resistance to augmentations [6], high-frequency artifacts [36], low contribution to the training task [48, 49], or backdoor samples may achieve lower loss at the early stage of training [11].

A recent work [23] proposed a confusion training procedure, which trains a model on a weighted combination of the randomly-labeled clean base set and the poisoned set. Introducing a randomly-labeled clean set into training prevents the model from fitting to the clean portion of the poisoned data, thereby allowing the identification of poisoned samples whose labels are consistent throughout the training process. Our experiment found that the effectiveness of [23] highly relies on the hyperparameter tuning of the weighted combined-training process and the performance varies significantly with poison ratios. Additionally, the fundamental assumption is that decoupling the benign correlations between semantic features and semantic labels does not influence the learnability of the correlations between backdoor triggers and target labels. However, some advanced clean-label backdoor attack trigger [2] strongly entangles with the semantic features of the target class; therefore, [23] falls short of detecting the trigger. At a high level, confusion training shares a similar idea to ours in the sense that we both leverage a clean base set to induce different detector behaviors between clean and poisoned samples. However, there are several key differences in the method design: our approach induces different behaviors by optimizing opposite optimization objectives on the base set and the poisoned set, whereas confusion training relies on random labeling to disrupt the learning of the clean samples. Importantly, we design a nested procedure that can effectively deal with the failure cases of [23]. Moreover, our method distinguishes itself from [23] by providing additional important advantages: (1) our approach does not require the poisoned set to be labeled, thereby enabling applications in SSL settings; and (2) our approach is robust to different poison ratios without ratio-specific tuning and can effectively detect attacks generating triggers entangled with semantic features.

3 Attacker & Defender Models

This section discusses standard threat models and assumptions about defender knowledge for different DL paradigms.

Case-0 End-to-end SL: In this setting, the attacker performs the backdoor attack by injecting a set of poisoned samples into the training dataset. The defender has access to the poisoned training dataset and the downstream learning algorithm. The defender's goal is to identify the poisoned samples within the training set and further remove the identified samples to prevent backdoor attacks from taking effect.

Case-1 SSL Adaptation: Under this setting, the attacker performs the backdoor attack by poisoning the unlabeled dataset [20, 21]. Following prior attack literature, we assume that the attacker does not have access to the fine-tuning task—the dataset or algorithm. Thus, the dataset used for fine-tuning is clean, and the attack only affects the unlabeled dataset. The defender has access to the complete training data, including both the data for SSL as well as the data for fine-tuning. In addition, the defender knows the algorithm for SSL and fine-tuning. The goal of the defender is to identify and remove the poisoned samples from the unlabeled dataset. Other attack settings target multi-modal contrastive learning, such as attacking the CLIP [19], is not considered in this case, as training CLIP requires additional text input supervision [50].

Case-2 Transfer Learning: The attacker performs the backdoor attack by poisoning a labeled dataset used for fine-tuning an existing pre-trained model. The defender knows the pre-trained model, the entire fine-tuning dataset (whose size often cannot support training a model from scratch), as well as the fine-tuning algorithm. The goal of the defender is to detect the poisoned samples within the fine-tuning dataset.

In all three cases, we assume the attacker can poison no more than half of the training dataset. We also assume that the defender has a small set of clean, unlabeled samples (the base set) to help with detection. These clean samples can be manually or automatically screened [24]. Compared with most recent detection methods [23], which require a *labeled* clean base set of at least 2000 samples, our method *relaxes* the requirement on the label information.

4 Proposed Method

4.1 Key Idea

Our goal is to enforce distinguishable model behaviors on poisoned and clean samples actively. The key idea is to design two optimizations that induce *opposite* model behaviors on *the poisoned dataset* (including its clean and poisoned portion) and *the clean base set*. Specifically, the two optimizations are performed simultaneously, where the first one *minimizes* a certain loss function on the clean base set and the second one *maximizes* the same loss on the entire poisoned training dataset. Note that the clean portion of the poisoned dataset and the clean base set are both drawn from the same clean distribution. Hence, the effect of the second optimization on the clean samples will be *offset* by the first optimization, and the loss on clean samples after the two optimizations is closer to the loss before. By contrast, the poisoned samples only go through the second optimization; therefore, the loss on the poisoned samples is maximized. Overall, as a result of the two optimizations, poisoned and clean samples will produce different loss values, thus becoming separable. The single offset's effect on clean samples and poisoned samples is illustrated in Figure 3 (**a**).

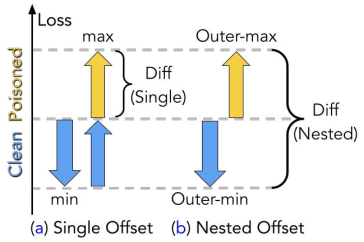

Figure 3: Illustration of **(a)** single offset procedure and **(b)** how the power of differentiating between clean and poison improves when the single offset is replaced by a nested loop.

Intuition on the distinguishability of poisons. Poisoning, whether through additive triggers [1], generative models [32], affine transformations [35], or even adaptive perturbation techniques [36], introduces a distributional shift from clean data. The resulting poisoning distribution and the original clean distribution have disjoint support, and thus the total variation (TV) distance between the two distributions is one. The Le Cam's lower bound, a classic result in statistical learning (refer to Chapter 15 in [51]), states that the minimum error over all detectors that classify the samples from two distributions, P_1 and P_2, is equal to $1/2(1-\|P_1-P_2\|_{\mathrm{TV}})$. Hence, there exists a detector achieving zero error probability for distinguishing between poisons and non-poisons. Le Cam's bound guarantees the existence of a good detector as long as poisons do not naturally appear in the clean distribution, and our method to be introduced is an effort to find such a detector based on the information of a clean base set.

4.2 Detection via Offset

Now, we formalize the offset idea for poisoned sample detection. Let D_{b} denote the clean base set and D_{poi} denote the poisoned training set. Formally, we can characterize the process of inducing distinguishable behaviors on poisoned and clean samples as a multi-objective optimization:

$$\theta^* \in \arg\min_{\theta} \frac{1}{|D_{\mathrm{b}}|} \sum_{x_{\mathrm{b}} \in D_{\mathrm{b}}} \mathcal{L}_{\min}\left(f(x_{\mathrm{b}}|\theta)\right) - \frac{1}{|D_{\mathrm{poi}}|} \sum_{x_{\mathrm{poi}} \in D_{\mathrm{poi}}} \mathcal{L}_{\max}\left(f(x_{\mathrm{poi}}|\theta)\right). \quad (2)$$

When discussing the high-level idea of our method, we assume that the minimization and maximization employ the same objective, i.e., $\mathcal{L}_{\min} = \mathcal{L}_{\max}$. However, these two functions can also be different; as long as minimizing $\mathcal{L}_{\min}$ and maximizing $\mathcal{L}_{\max}$ induce different model behaviors, one optimization will mitigate the effect of the other on the clean distribution.

In the implementation, we do not directly solve the optimization with two optimizations at the same time due to the instability of the corresponding optimization path; instead, we *loop* between two objectives:

1. We first minimize $\mathcal{L}_{\min}$ by taking a *gradient descent* step on a mini-batch drawn from the base set;
2. Then, we utilize the resulting model as the initializer for maximizing $\mathcal{L}_{\max}$ and perform a *gradient ascent* step on a mini-batch drawn from the poisoned set;
3. Repeat the above two steps.

We empirically observe the alternating procedure is stable. As the focus of the paper is to develop practical detection methods, we will defer the theoretical analysis of this procedure—an interesting open problem—for future work.

Next, we will discuss which loss function we shall use to instantiate $\mathcal{L}_{\min}$ and $\mathcal{L}_{\max}$. With the goal of detecting unlabeled poisoned data in mind, we propose a loss function, which calculates the variance of the logits. Let $f(x|\theta)$ denote the output logit of a model that is parameterized by θ and takes x as input. For a model that performs k-class classification, $f(x|\theta) \in \mathbb{R}^k$ and the i-th class logit is denoted by $f(x|\theta)_i$. Furthermore, let $\overline{f(x|\theta)}$ denote the average of the output logits of all classes. Then, our proposed loss function can be expressed as

$$\mathcal{L}_{\mathrm{var}}(f(x|\theta)) = \frac{1}{k}\sum_{i=0}^{k}\left(f(x|\theta)_i - \overline{f(x|\theta)}\right)^2. \quad (3)$$

When the detection is performed on the *unlabeled* data, we can instantiate both $\mathcal{L}_{\min}$ and $\mathcal{L}_{\max}$ to be $\mathcal{L}_{\mathrm{var}}$ defined above, because calculating $\mathcal{L}_{\mathrm{var}}$ does not require label information. As the result of minimizing $\mathcal{L}_{\min}$, the clean samples are forced to have a *flat* logit pattern. Then, the maximization optimization maximizes the same loss on the poisoned dataset, which induces high-variance logits for poisoned samples. For clean samples, the effects of maximization and minimization are roughly canceled out. Therefore, clean samples are expected to produce lower-variance logits than poisoned samples.

When the detection is performed on a labeled poisoned dataset, we find that instantiating $\mathcal{L}_{\max}$ with the cross-entropy-based prediction loss $\mathcal{L}_{\mathrm{ce}}$ achieves a good detection performance faster than $\mathcal{L}_{\mathrm{var}}$:

$$\mathcal{L}_{\mathrm{ce}}(f(x|\theta), y) = -\sum_{i=1}^{k} y_i \log \sigma(f(x|\theta))_i, \quad (4)$$

where $\sigma(x)_i$ denotes the i-th output of the softmax and y represents the one-hot encoding of x's label.

It is worth mentioning that we fix the minimization loss to be $\mathcal{L}_{\text{var}}$ regardless of whether the base set is labeled or unlabeled. We found that even when the label information is available, this choice still leads to better detection performance than using $\mathcal{L}_{\text{ce}}$ as the minimization goal. This is because learning through minimizing $\mathcal{L}_{\text{var}}$ will make the model extract class-independent features. A mini-batch of the base set may be class-imbalanced or sometimes contain only partial classes due to random sampling. Hence, $\mathcal{L}_{\text{var}}$ can be more steadily minimized than $\mathcal{L}_{\text{ce}}$ via mini-batch gradients.

4.3 Strengthened Detection via Nested Offset

Weakness of a single offset. Despite the neatness of the offset idea, directly solving the two optimizations with the proposed loss functions is limited in tackling attacks with *low poison ratio* and the settings where *poisoned samples take effect slowly during training* (i.e., attacks need many epochs of training to obtain a high enough success rate; examples of such attacks include [2,21]). The reasons are as follows. In the low poison ratio setting, mini-batches naturally contain very small amounts of poisoned samples; on the other hand, each gradient ascent step takes a step towards reducing the *average* loss over a mini-batch and tends to overlook the minorities. Hence, the loss of poisoned samples would be increased by less with a lower poison ratio. To explain the second limitation, note that θ is an over-parameterized model (e.g., ResNet-18 and Vision Transformer). If an attack takes many epochs to take effect, then we need to train θ for long enough. The model after long training will end up "memorizing" all the samples from the base set and the poisoned set, i.e., all the samples from the base set achieve a low value of $\mathcal{L}_{\text{min}}$ and all the samples from the poisoned set (including both clean and poisoned samples) to achieve a high value of $\mathcal{L}_{\text{max}}$. In that case, the poisoned portion and the clean one are inseparable.

How to mitigate these failure cases? To illustrate our idea, let us think about a hypothetical design, assuming one can perfectly pinpoint a set of poisoned samples. In this design, we keep the first step minimizing on the clean base set, but the second maximization is performed on purely poisoned samples instead of the poisoned training set, which generally contains a large portion of clean samples and only a small portion of poisoned samples. This hypothetical design would be able to solve the two failure cases above. For the first case, since mini-batches for maximization contain solely poisoned samples, the poisoned samples would still have their loss increased and thus is distinguishable from the clean ones. For the second case, while long training can lead to memorization but with the hypothetical design, it is just the poisoned samples that get memorized and are assigned with high loss; therefore, the poisoned samples and the clean ones are still separable.

While having access to a set of purely poisoned samples is not realistic, this thought experiment inspires an idea to improve an offset-based detection approach, which is to replace the poisoned training set (dominated by clean samples) with a set dominated by poisoned samples in the second maximization. To form such a poison-dominated set, we can leverage a new offset loop (referred to as the *inner* offset loop) to mark a set of the most suspicious samples. Then, we use those samples to perform maximization of the original offset loop (referred to as the *outer* offset loop).

How to design the inner offset loop that provides a poison-condensed set? First, it is not ideal to reuse the design of the outer loop for this inner one, because in that case the inner would suffer the same "memorization" issue. Instead, we aim to avoid "overparameterized" models and perform the inner loop with a simple model. On the other hand, a simple model could be incapable of extracting complex features to support the detection of poisoned samples. Our solution is to use the poisoned model (i.e., the downstream model trained on the poisoned dataset) to extract features from the poisoned set and the base set and then optimize a simple model to detect the poisoned samples in the feature space.

Note that the embedding space of a poisoned model has been shown to be informative to detect many but not all backdoor attacks (detailed in Section 5). Although the poisoned and clean samples are not perfectly separable based on the embeddings—as illustrated in Figure 6—the reason why these methods underperform in many cases, the poisoned model still provides a well-trained embedding space and some imperfect signals for selecting a poison-condensed set.

Detailed design of the inner offset loop. The inner offset loop is executed *inside* the previous offset loop (Eqn. 2). It condenses the poison in a mini-batch sampled by the maximization step of the outer offset loop. Specifically, the inner offset loop will return a set of samples marked as poison. We will use this poison-condensed subset of the original mini-batch to perform the outer maximization. When the inner loop is relatively precise in gathering a poison-condensed subset, the outer loop will maximize the outer loss of poisoned samples without introducing much offset effect on clean samples. As a result, the poisoned and clean samples become more distinguishable in terms of the outer loss compared to a single offset loop via Eqn. 2. An intuitive explanation of the improvement is illustrated by Figure 3 (**b**).

Let $f(x|\theta^*_{\text{poi}})$ denote the poisoned model, and its parameters are given by θ^*_{poi}. Let $M(\cdot|w)$ be a mapping from the logits to a real value in the range $[0,1]$, and w denotes its parameters. The inner offset can be characterized by

$$w^* = \arg\min_{w} \underbrace{\frac{1}{|B_{\text{b}}|}\sum_{x_{\text{b}}\in B_{\text{b}}} \mathcal{L}_{\text{BCE}}\left(M(f(x_{\text{b}}|\theta^*_{\text{poi}})|w),0\right)}_{\mathcal{L}_1} + \underbrace{\frac{1}{|B_{\text{poi}}|}\sum_{x_{\text{poi}}\in B_{\text{poi}}} \mathcal{L}_{\text{BCE}}\left(M(f(x_{\text{poi}}|\theta^*_{\text{poi}})|w),1\right)}_{\mathcal{L}_2}, \quad (5)$$

where $\mathcal{L}_{\text{BCE}}(p,q) = -p\log q + (1-p)\log(1-q)$, representing the binary cross entropy loss and B_{b} and B_{poi} stand for a mini-batch drawn from the clean base set and the poisoned training set, respectively.

The first minimization objective will encourage learning a mapping M such that the mini-batch from the clean base set is labeled as "0"; the second objective will further promote M to label the mini-batch from the poisoned set as "1". By minimizing the two objectives simultaneously, the effect on the clean data gets canceled. As a result, the clean samples will be predicted as "1" with low confidence, yet the poisoned ones will be predicted as "1" with high confidence. Then, we can mark the samples with the highest confidence or the lowest BCE loss for predicting "1" as the suspicious poisoned samples. In practice, M is implemented as a two-layer, full-connected network with 128 hidden neurons. Again, to avoid stability issues, in the implementation, we first take a gradient descent step to minimize $\mathcal{L}_1$ and then take a gradient ascent step to minimize $\mathcal{L}_2$, and alternate between the two steps.

The pseudo-code for the inner offset loop is provided in Algorithm 1, termed *Poison Concentration*.

Algorithm 1: Poison Concentration

Input: θ^*_{poi} (Poisoned feature extractor);
B_{poi} (Poisoned training mini-batch);
B_{b} (Base set mini-batch);
Output: B_{pc} (Poison concentrated mini-batch);
Parameters: $\mathcal{N}$ (Total inner loop iteration number);
$\gamma > 0$ (Step size);
λ (Threshold);

/* 1.Dynamic training of M */
for *each iteration* j *in* $(0, \mathcal{N}-1)$ **do**
  $M'_j \leftarrow M_j - \gamma \frac{1}{|B_{\text{b}}|} \sum_{x_{\text{b}} \in B_{\text{b}}} \frac{\partial \mathcal{L}_{\text{BCE}}\left(M\left(f(x_{\text{b}}|\theta^*_{\text{poi}})\right),0\right)}{\partial M}$;
  $M_{j+1} \leftarrow M'_j - \gamma \frac{1}{|B_{\text{poi}}|} \sum_{x_{\text{poi}} \in B_{\text{poi}}} \frac{\partial \mathcal{L}_{\text{BCE}}\left(M'\left(f(x_{\text{poi}}|\theta^*_{\text{poi}})\right),1\right)}{\partial M'}$;
/* 2.Get output values */
$V \leftarrow M_{\mathcal{N}}\left(f(B_{\text{poi}}|\theta^*_{\text{poi}})\right)$;
/* 3.Using AO to determine outliers */
$B_{\text{pc}} \leftarrow B_{\text{poi}}[\mathbf{AO}(V) \geq \lambda]$;
return B_{pc}

Adaptive thresholding for the inner offset. The last step of Poison Concentration is to select the subset marked as poison based on the confidence score output by M. We will elaborate on how to adaptively choose the size of this subset. First, directly adopting a fixed threshold to identify the most likely poisoned samples is impractical, as different mini-batches may contain different amounts of poisons. To tackle this problem, we adopt Adjusted Outlyingness (AO) [52] to adaptively determine the number of most suspicious samples within each mini-batch. AO maps the BCE losses into a scale such that a fixed threshold can effectively identify the most suspicious samples. Note that AO does not aim to filter out as many poisoned samples as possible within the mini-batch; instead, it is adopted to achieve high precision, i.e., identifying a subset of the mini-batch that is dominated by poisoned samples. In the evaluation, we threshold the output of AO with 2. By the nature of AO, we are essentially adopting an adaptive threshold despite using a fixed output value (see Figure 8).

4.4 Overall Workflow

The overall algorithm of ASSET with two offset loops is presented in Algorithm 2. Functionally speaking, the inner loop condenses the poison within each mini-batch drawn from the poisoned dataset, the outer loop induces different model behaviors on clean samples and poisoned samples. At each iteration of the outer, we minimize $\mathcal{L}_{\text{var}}$ by taking mini-batch gradient descent with samples from the clean base set; then, we perform the poison concentration step: the inner returns subset of samples most likely to be poisoned; then proceed to the maximization step of the outer $\mathcal{L}_{\max}$ by doing gradient ascent with the suspicious points returned by the inner. In the end, we can obtain a detector model $f(\cdot|\theta_I)$ with parameters θ_I obtained after I outer iterations and this model induces different values of $\mathcal{L}_{\max}$ between clean and poisoned samples.

Algorithm 2: ASSET Backdoor Detection

Input: θ_0 (Initialized detector);
θ^*_{poi} (Poisoned feature extractor);
D_{poi} (Poisoned training set);
D_{b} (Base set);
Output: S_{poi} (Indexes of the detected poisoned samples);
Parameters: I (Total outer loop iteration number);
$\alpha > 0$ (Step size);

for *each iteration* i *in* $(0, I-1)$ **do**
  /* 1. Obtaining mini-batches */
  $B^i_{\text{poi}} \leftarrow B^i_{\text{poi}} \in D_{\text{poi}}$;
  $B^i_{\text{b}} \leftarrow B^i_{\text{b}} \in D_{\text{b}}$;
  /* 2. Minimization */
  $\theta' = \leftarrow \theta_i - \alpha \frac{1}{|B^i_{\text{b}}|} \sum_{x^i_{\text{b}} \in B^i_{\text{b}}} \frac{\partial \mathcal{L}_{\text{var}}\left(f(x^i_{\text{b}}|\theta_i)\right)}{\partial \theta_i}$;
  /* 3. Poison Concentration */
  $B^i_{\text{pc}} \leftarrow$ **Poison Concentration** $\left(B^i_{\text{poi}}, B^i_{\text{b}}, \theta^*_{\text{poi}}\right)$;
  /* 4. Maximization */
  $\theta_{i+1} \leftarrow \theta'_i + \alpha \frac{1}{|B^i_{\text{pc}}|} \sum_{x^i_{\text{pc}} \in B^i_{\text{pc}}} \frac{\partial \mathcal{L}_{\max}\left(f(x^i_{\text{pc}}|\theta')\right)}{\partial \theta'}$;
/* 5.Get output loss values */
$V \leftarrow \mathcal{L}_{\max}\left(f(D_{\text{poi}}|\theta_I)\right)$;
/* 6.Detection result via adaptive GMM */
$S_{\text{poi}} \leftarrow$ **adaptive GMM** (V);
return S_{poi}

Adaptive thresholding for the outer loop. With the trained detector model, θ_I, we now discuss how to identify the poisoned samples. Similar to the inner, we propose an adaptive thresholding method for the outer as well. Note that the threshold of the inner and outer loop has distinct goals. The inner loop aims to identify a subset with a high density of poisons, while the outer loop aims to adaptively conduct a split between the clean and poisoned loss distribution that helps the detector to remove as many poisons as possible while maintaining a low false positive, i.e., high precision is prioritized for the inner yet high recall is prioritized for the latter.

As will be shown later, after the overall optimization, $f(\cdot|\theta_I)$ will output distinct loss distribution for the clean and poisoned samples. One might be tempted to directly fit a Gaus-

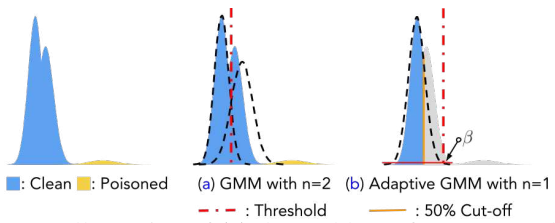

Figure 4: Illustration of **(a)** the problem of GMM over long-tailed cases where the attacks are of low poison ratio and **(b)** how the proposed adaptive GMM can help.

sian Mixture Model (GMM) with two components. However, doing so is problematic, as depicted in Figure 4. Since there are usually much fewer poisoned samples than clean ones, the GMM tends to split the multiple-modal clean distribution into two Gaussian distributions instead of fitting two Gaussians respectively to the clean and poison distributions.

To tackle this problem, we propose a simple twist of GMM, termed **adaptive GMM**. We first abandon half of the samples achieving the highest values of $\mathcal{L}_{\text{max}}$, which will remove all the poisoned samples (we assume the attacker can poison no more than half of the training dataset, Section 3). Then, we fit a Gaussian to the remaining points. Since the optimized detector model largely centers the clean samples' loss close to $\mathcal{L}_{\text{var}} = 0$ or $\mathcal{L}_{\text{ce}} = -\log(\frac{1}{k})$, the Gaussian fitted on the remaining samples remains similar to the Gaussian fitted on all the non-poisons (see Figure 4 (b)). Lastly, we set a small threshold on the Gaussian density, β, to cut off the samples that are unlikely to be generated from the fitted Gaussian. In practice, we set the cut-off threshold as $\beta = 10^{-6}$, which equivalently keeps the lowest-loss samples with a probability higher than $> 99.99\%$ being generated from the fitted Gaussian (for any Gaussian distribution with a variance smaller than 10).

5 Evaluation

Our evaluation aims to answer the following questions.

- **Case-0** (Section 5.2): How does ASSET compare with other methods in end-to-end SL setting? Is detection effective when multiple attacks exist simultaneously? How does the detection performance vary over different attacks and poison ratios?
- **Case-1** (Section 5.3): Can ASSET robustly detect attacks in SSL settings? How does the knowledge about downstream tasks affect the defense's effect?
- **Case-2** (Section 5.4): Can ASSET provide reliable backdoor sample detection in TL settings? What are the limitations of other defenses in this setting?
- **Adaptive Attack** (Section 5.5): Is it possible to adaptively evade ASSET's detection?
- **Ablation Study** (Appendix 6.4): How do different design choices affect the final performance of ASSET?

5.1 Settings

Evaluation metrics. There are two key aspects throughout our evaluation: (1) How accurately can the poisoned samples be detected (***upstream evaluation***)? (2) After the suspicious points are removed, how would a downstream model learn from the remaining data perform (***downstream evaluation***)?

For upstream evaluation, we utilize two metrics, namely, True Positive Rate (**TPR**), $TPR = TP/(TP+FN)$, and False Positive Rate (**FPR**), $FPR = FP/(FP+TN)$, where TP, FP, TN, and FN denote the number of true positives, false positives, true negatives, and false negatives, respectively[4]. TPR depicts how well a specific backdoor detection method filters out the backdoored samples. A higher TPR (closer to 100%) denotes a stronger filtering ability. FPR depicts how precise the filtering is: when a specific method achieves TPR that is high enough, FPR helps us to understand the trade-off, i.e., how many clean samples are wasted and wrongly flagged as backdoored during the detection. A lower FPR shows that fewer clean samples are wasted, and more clean data shall be kept and available for downstream usage.

One thing worth noting is that no detection method can reliably remove all the poisoned samples. However, *the remained backdoor samples that go unnoticed by a successful defense should be small enough to deactivate attacks*. Thus, we evaluate the backdoor attacks' Attack Success Rate (**ASR**) on the downstream model trained using the filtered dataset to study whether the detection is good enough to stop attacks. ASR measures the proportion of backdoored test samples being classified into target classes. Additionally, we evaluate the downstream model's Clean Accuracy (**ACC**). A high ACC means that the detection method is able to maintain a large enough clean set to support the model performance.

Dataset & models. We incorporate three standard computer vision benchmark datasets into our evaluation: CIFAR-10 [53] (main text), STL-10 [54] (Appendix 6.3), and ImageNet [55] (a randomly selected 100-class subset, Appendix 6.3). To ensure the effectiveness of the baselines and fair comparison, we set the base set size as 1000 for all the settings. We will later show that our method is robust to different choices of the base set size in the ablation study, Appendix 6.4. We obtain a 1000-size clean base set for each dataset by randomly selecting the samples from the test set and removing their label information. All the upstream evaluation metrics (i.e., TPR and FPR) are evaluated on the respective training sets, i.e., the training set of **Case-0**, the fine-tuning set of **Case-2**, and the unlabeled pre-training set for **Case-1**. For **Case-0**, we adopt all the remaining data from the test set for evaluation of the downstream metrics (i.e., ACC and ASR). For **Case-1** and **Case-2**, we split the remaining test set into half being fine-tuning set and half being the downstream metric evaluation set. ResNet-18 [56] is adopted on the CIFAR-10. ViT-Small/16 [14] is adopted on STL-10 and ImageNet (Appendix 6.3). For **Case-1**, we incorporate four state-of-the-art SSL training methods, i.e., SimCLR [12], MoCo V3 [17], BYOL [18], and the MAE [13], for evaluation. For **Case-2**, we consider the two most popular transfer learning cases, namely, FT-all and FT-last (detailed in Section 2). The pre-trained model parameters for fine-tuning are loaded from the timm library[5].

[4] Note that poison is considered positive and clean is considered negative.

[5] https://timm.fast.ai/

	Dirty-Label Backdoor Attacks								*Clean-Label Backdoor Attacks*						Average		Worst-Case	
	BadNets (5%)		**Blended (5%)**		**WaNet (10%)**		**ISSBA (1%)**		**LC (1%)**		**SAA (1%)**		**Narci. (0.05%)**					

(a) Upstream Evaluation

	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
Spectral	95.6	2.86	99.8	2.64	0.64	16.6	0.00	1.52	80.2	0.71	87.6	0.63	0.00	0.08	51.9	3.58	0.00	16.6
Spectre	96.9	0.28	99.8	2.64	1.00	16.6	80.2	0.71	99.8	0.51	99.4	0.51	0.00	0.07	68.2	3.05	0.00	16.6
Beatrix	93.8	1.81	67.9	3.04	82.2	0.53	73.4	1.31	91.2	0.29	69.8	1.58	12.0	1.97	70.4	1.50	12.0	3.04
AC	90.5	40.1	65.4	44.9	8.30	41.5	11.0	41.1	91.2	0.41	75.6	21.5	0.00	34.3	48.9	32.0	0.00	44.9
ABL	85.4	3.40	93.4	2.98	28.1	13.5	55.2	0.96	87.2	0.63	73.4	0.77	0.00	0.07	60.4	3.19	0.00	13.5
Strip	25.4	11.5	17.3	12.1	5.08	10.0	68.8	9.34	100	0.85	63.4	1.22	0.00	0.05	40.0	6.44	0.00	10.0
CT	99.0	3.72	98.1	4.53	95.8	2.64	96.6	4.37	100	9.01	95.2	5.44	0.00	5.54	83.5	5.03	0.00	9.01
Ours	99.5	0.55	100	0.00	90.7	8.09	95.6	0.36	96.2	0.75	96.6	0.39	92.0	0.34	**95.8**	**1.49**	**90.7**	**8.09**

(b) Downstream Evaluation

	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
No Def.	96.5	93.4	94.9	93.5	99.4	93.5	92.6	94.1	100	94.7	76.7	94.4	99.7	94.9	94.3	94.1	100	93.4
Spectral	48.4	94.5	10.7	94.1	98.9	90.0	93.0	94.1	10.6	94.8	3.11	94.2	99.7	94.8	52.1	93.8	99.7	90.0
Spectre	34.8	94.5	6.57	94.1	100	89.6	14.0	94.3	100	94.7	0.86	94.4	99.8	94.9	50.9	93.8	100	89.6
Beatrix	55.6	93.8	94.9	93.8	2.13	94.1	17.0	94.2	4.12	94.8	8.64	94.3	90.4	94.5	39.0	94.2	94.9	93.8
AC	81.3	76.9	93.3	82.1	99.7	83.1	83.5	81.3	4.31	94.8	7.63	87.7	100	90.7	67.1	85.0	100	76.9
ABL	88.6	92.5	94.2	88.7	90.2	93.1	30.6	94.2	6.32	94.7	7.63	94.4	99.3	94.9	59.6	93.2	99.3	88.7
Strip	76.9	85.3	93.8	87.1	98.6	91.7	25.5	91.0	0.38	94.8	9.63	94.4	99.8	94.9	57.8	91.3	99.8	81.3
CT	3.42	93.1	31.3	91.2	0.53	92.5	1.12	93.2	0.44	91.1	2.16	93.2	100	94.1	19.9	92.6	100	91.1
Ours	2.68	94.9	0.44	95.2	1.89	93.1	1.55	94.8	1.16	94.9	1.14	94.4	9.68	94.9	**2.65**	**94.6**	**9.68**	**93.1**

Table 2: (a) Upstream and (b) Downstream evaluation and comparison results under **Case-0**, CIFAR-10. We list the poison ratio of each attack at the top of each column, which follows the original work that proposed these attacks. We highlight the ASR below 20% in **blue** as a success defense, the ASR above 20% in **red** as a failed defense case.

Baseline defenses. Referring to Table 1, we incorporate a wide range of existing backdoor detection for comparison, including both standard baselines used in prior work as well as state-of-the-art ones. In particular, we consider Spectral [7], Spectre [8], and the Beatrix [9]; we include AC [10] as a representative work that utilizes intermediate neural activation; ABL [11], which was originally a robust training defense and repurposed as a detection method based on output losses; Strip [6] as a representative detection approach based on model outputs; and CT [23], the most recent work reported achieving state-of-the-art performance on end-to-end SL settings based on confusion training. All the implementations and hyperparameters follow the original papers. For methods that rely on or can be boosted by an additional base set, e.g., Spectre, Beatrix, Strip, CT, we use the same 1000-size base set as ours. We note that this comparison setting might not be fair, as compared to these baselines, our method relaxes the requirement on label information; in addition, AC and ABL cannot be adapted to use the base set. We want to show that even without label information, our method can still achieve comparable or much better results with stronger robustness than the other baselines. Detailed explanations of the defense settings and how we adapted them to **Case-1** and **Case-2** are provided in Appendix 6.1.

Backdoor attack settings. For **Case-0** we incorporate seven standard or state-of-the-art attacks, including four dirty-label and clean-label ones. For dirty-label backdoor attacks, we incorporate localized backdoor attack BadNets [1], global-wised blended trigger Blended [29], wrapping-based invisible backdoor attack WaNet [35], and the state-of-the-art sample-specific invisible backdoor attack, ISSBA [32]. For clean-label attacks, we include the standard Label Consistent (LC) attack [39], the state-of-the-art feature-collision-based hidden trigger backdoor, Sleeper Agent Attack (SAA) [41], and the state-of-the-art optimization-based Narcissus attack (Narci.) [2]. For **Case-1**, only limited existing work has explored the attack over SSL's unlabeled training set. We incorporate the Checkerboard trigger (C-brd) used in [19], the Colored Square trigger (C-squ) used in [20], and the state-of-the-art YCbCr frequency-based invisible trigger used in CTRL [21]. In particular, CTRL has been shown to achieve a magnitude higher attacking efficacy than [20]. For **Case-2**, directly implementing some of the attacks from end-to-end SL may not lead to effective attacks, e.g., the Blended attack cannot achieve high ASR under the FT-all settings. Thus, we consider attacks that can maintain effectiveness for each TL setting. BadNets and the SAA are adopted for evaluation under the FT-all case. Blended and the hidden trigger backdoor attack (HTBA) [40] are adopted for the evaluation under the FT-last case. All the incorporated attacks' settings, such as trigger design and trigger strength, all follow their original papers. Appendix 6.2 details the specifics of these attacks' setups under each learning paradigm and visual examples of the poisoned samples we intend to detect.

5.2 Case-0: End-to-end SL

Detection performance against different attacks in SL. Table 2 presents the upstream and downstream evaluation results under the end-to-end SL setting on the CIFAR-10 dataset with the ResNet-18 model trained from scratch for 200 epochs. For each different attack, we adopt the poison ratio following each original paper, which is listed at the top of each column. We have included the row of "No Defense" in Table 2 (b)

	Poison Ratio%	Spectral [7]	Spectre [8]	Beatrix [9]	AC [10]	ABL [9]	Strip [6]	CT [9]	Ours
BadNets	0.05%\25	25	23	13	22	7	19	1	**0**
	1%\500	37	23	13	416	32	446	**0**	17
	5%\2500	109	109	155	238	365	1866	20	**13**
	20%\10000	817	170	113	1086	4590	330	16	**7**
	50%\25000	7963	158	264	774	1944	1001	25000	**4**
Blended	0.05%\25	25	25	22	25	19	23	**2**	4
	1%\500	86	44	53	49	41	413	16	**2**
	5%\2500	6	5	803	866	1023	2068	33	**0**
	20%\10000	226	27	31	306	4965	1669	3659	**13**
	50%\25000	9568	1023	2386	1514	13959	10736	25000	**8**

Table 3: # poisons remained in the filtered training set after defense (**Case-0**, CIFAR-10). **Bolded** results denote the smallest value. red to highlight failed defenses where more than 30 poisoned samples remain as we find this amount of poisons still enables ASRs greater than 30%.

to show the attack effects without any backdoor detection defense in place. Existing methods are able to achieve decent detection effects on some specific attacks, but they experience large performance variations when defending different attacks. These methods either solely rely on the embedding space of a poisoned model that may change with different trigger designs or rely on some detection rule that may not apply to specific backdoor designs. For example, ABL assumes that backdoor samples achieve the lowest loss at the early stage of training. However, the Narci. clean-label poisoned samples' losses do not meet the assumption; thus, ABL is not effective on the Narci. The recently proposed CT achieves the highest detection rate and the most consistent performance among all baselines, but it still fails to detect the state-of-the-art clean-label attack, Narci. Notably, no existing detection method obtains satisfying results as Narci. introduces optimized features as robust as the semantic features of the target class [2]. Regarding the upstream evaluation in Table 2 (a), our method reliably achieves a TPR above 90% for all the evaluated settings and significantly improves the state-of-the-art in terms of the average and worse-case defensive performance over different attacks. Regarding the downstream evaluation in Table 2 (b), we find that ASSET is the only defense that gives rise to robust models over all the evaluated poisoned datasets, i.e., all ASRs drop below random guessing rate, i.e., 10%. In particular, our method is the only effective method to mitigate Narci. Moreover, the downstream models trained over ASSET filtered datasets achieve the highest average ACC. Notably, the average ACC of our method is slightly higher than using the original poisoned dataset (which contains more clean samples). Results for multiple attacks introduced simultaneously are provided in Appendix 6.3, with similar observations.

Unlike ASSET, the existing methods do not have an active process to induce differentiating behaviors between clean samples and poisoned ones. Thus clean and poisoned samples often have overlapping behaviors and cannot be easily separated. We illustrate the separation between clean and poisoned samples using different detection methods and their threshold in Figure 5, emphasizing the importance of the proposed active offset process.

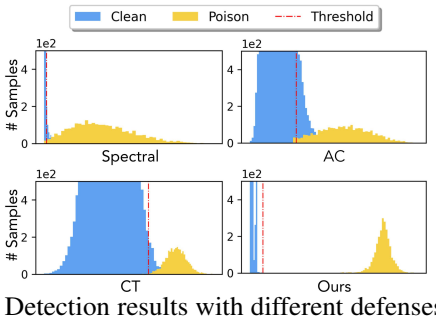

Figure 5: Detection results with different defenses in distribution histograms (CIFAR-10, Blended attack, 5%, **Case-0**). We emphasize the effects and the necessity of adaptive thresholding and the process of actively pushing the distribution of clean and poison away from each other.

Impact of poison ratios. In Table 3, we study the effects of poison ratio on different detection methods against two standard attacks, namely, BadNets, and the Blended attack. Most existing detection works better for small poison ratios but fails as the ratio increases. One reason is that many works, such as Spectral, Spectre, and AC, are based on the feature distribution of the poison dataset. However, an increased poisoning rate will cause the clean feature distribution to be closer to the poisoned one, making them less separable. CT is the most robust baseline in the previous evaluation, but it also fails for very large ratios like 20% (10000 poisons) or 50% (25000 poisons). The reason could be that their detector uses fixed hyperparameters that are fine-tuned on small poison ratios. Our defense is robust to poison ratio changes, even for extreme cases where half of the samples in the training set are poisoned or only 25 (0.05%) samples are poisoned.

5.3 Case-1: SSL Adaptation

	C-brd (0.5%)		C-Squ (0.5%)		CTRL (1%)		Average		Worst-Case	
	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
Spectral	0.08	7.89	0.44	7.87	1.20	1.50	0.57	5.75	0.08	7.89
Spectre	0.64	7.86	2.36	7.77	0.40	1.51	0.64	5.71	0.40	7.86
Beatrix	0.88	7.85	2.76	7.75	73.4	2.79	25.7	6.13	0.88	7.85
AC	6.72	28.6	9.88	28.4	36.8	21.3	17.8	26.1	6.72	28.6
ABL	5.76	7.59	6.24	7.57	20.2	1.31	10.7	5.49	5.76	7.59
Ours	91.5	0.67	96.0	0.25	97.4	0.69	**95.0**	**0.54**	**91.5**	**0.69**

Table 4: Upstream evaluation and comparison results under **Case-1** with SimCLR. The **bolded** results denote the best defense results among the evaluated defenses.

Detection performance against different attacks in SSL. Now we study the efficacy in detecting unlabeled poisons under the SSL adaptation cases. Table 4 and Table 5 list out the upstream and downstream evaluation results, respectively, on CIFAR-10 using ResNet-18 trained via SimCLR-based SSL for 600 epochs with linear adaptation for 100 epochs. We find that the ASRs of C-brd and C-Squ are below 20% so these attacks cannot lead to a successful attack on average. We still keep their results but show the number of successfully attacked samples (denoted with ASR^*) as done in [20]. Even though these attacks do not result in as high ASR as the attacks in SL or as the CTRL attack, they can still result in an increase of samples with triggers being classified as the target class.

	C-brd (0.5%)		C-Squ (0.5%)		CTRL (1%)	
	ASR* ↓	ACC ↑	ASR* ↓	ACC ↑	ASR ↓	ACC ↑
No Def.	404	85.2	435	84.6	81.4	85.3
Spectral	405	84.1	478	84.2	81.3	85.2
Spectre	405	84.1	445	84.2	81.4	85.3
Beatrix	402	84.2	444	84.2	16.8	85.0
AC	513	73.26	376	73.2	36.5	78.6
ABL	380	84.6	399	84.4	46.6	85.3
Ours	100	85.1	87.0	84.9	2.47	85.9

Table 5: Downstream evaluation and comparison results under **Case-1** with SimCLR. We highlight the ASR below 20% in **blue** as a success defense, the ASR above 20% in **red** as a failed defense case. ASR* is the number of successfully attacked samples. We use ASR* instead for the C-brd and the C-Squ attack, referring to the original work [20], as their ASRs are naturally low to SSL paradigms.

As shown in Table 4, among all the evaluated attacks, our method obtains the highest TRP while remaining the lowest FPR among all detection methods. Noting the absence of CT under the SSL. Recall that in the SL setting, CT can achieve compatible results as our method on most attack settings; yet, it is inapplicable to SSL as its core technique—confusion training—relies on label information [23]. In particular, as C-brd and C-Squ do not result in a high ASR as shown in Table 5, the model's response to clean and backdoor samples is not sufficiently different, thereby making detection very difficult. In fact, none of the baselines provides reliable detection of these two attacks. For the CTRL attack, which achieves an ASR of over 80%, we start to see that some of the baseline defenses take effect, e.g., the Beatrix. But still, our method achieves the best upstream detection performance (Table 4) and gives rise to the highest ACC and lowest ASR downstream (Table 5).

Further evaluation with more SSL training algorithms. We further evaluate our defense under other popular SSL training algorithms and different model structures and datasets, e.g., ResNet-18 and ViT-Small/16 trained using SimCLR, MoCO V3, BYOL, MAE over CIFAR-10 or the ImageNet (Appendix 6.3). The upstream and downstream evaluation results on the CIFAR-10 are shown in TBALE 6 and Table 7, respectively. Across all the evaluated settings, our method provides reliable upstream detection results with TPRs over 90% for all the cases and low FPRs. Thanks to the upstream efficacy, our detection method can give rise to the downstream model with a low ASR and an ACC close to or better than the settings without removing any training point. Overall, our results demonstrate that our method can reliably sift out the poisoned samples across different settings of SSL adaptation.

Impact of # logits w.r.t. SSL downstream task. Note that for SSL evaluation, the pre-trained model requires a fixed number of logits, each corresponding to a different output category. In our evaluation, we use the actual classes contained (e.g., 10 for the CIFAR-10 and 100 for the ImageNet 100-subset). Such a setting is applicable when the defender knows the exact downstream classification task. Now we consider a much more strict case where one tries to conduct detection over unlabeled datasets without any prior knowledge about the number of categories in downstream tasks. As shown in Table 8, we find our method is robust to the change in the number of logits and can maintain a TPR higher than 90%.

	C-brd (0.5%)		C-Squ (0.5%)		CTRL (1%)	
	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
SimCLR	91.5	0.67	96.0	0.25	97.4	0.69
MoCo V3	91.3	0.49	96.9	0.20	98.2	0.32
BYOL	95.9	0.22	95.8	0.35	94.6	0.57
MAE	97.2	0.67	98.2	0.50	97.2	0.73

Table 6: Further upstream evaluation of ASSET under **Case-1** with four SSL training algorithms, CIFAR-10.

5.4 Case-2: Transfer Learning

Detection performance against different attacks in TL. We consider two of the most popular TL schemes for evaluation: FT-all and FT-last with models pre-trained on the ImageNet. All the existing backdoor defenses can be easily generalized to TL. However, none of them has empirically evaluated the backdoor detection efficacy under the TL settings in the prior literature, which leaves a gap to fill.

The upstream and downstream results are listed in Table 9. Existing methods' detection results on FT-all seem more consistent than the results on FT-last. This observation might be due to that FT-all is a setting much closer to the end-to-end SL. While many defenses can achieve satisfying results on some specific attacks in SL, none can achieve a TPR above 90% for all attack settings in TL, except CT on BadNets. We now take a closer look at the reason why existing detection methods fall short in TL. We depict the feature space t-SNE results comparing the attacks in **Case-0** and **Case-2** in Figure 6. Since in TL, the model parameters have been initialized with additional knowledge obtained from pre-training, clean and poisoned samples are harder to be separated in the embedding space, thus resulting in a worse detection result compared to SL. As shown in Figure 6, for both BadNets and the Blended attack, the clean and poisoned samples have a larger overlapping in the TL case than in SL. These results emphasize the importance of introducing active measures to increase separability.

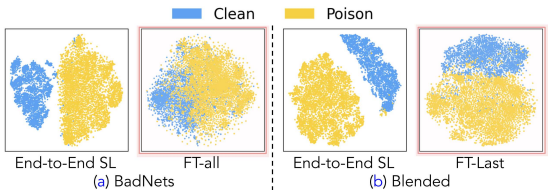

Figure 6: In-class features space t-SNE results with the model trained with CIFAR-10 using end-to-end SL or TL: **(a)** BadNets 20%, **(b)** Blended 20%.

On the other hand, for all the evaluated settings on the two datasets (CIFAR-10 and STL-10, Appendix 6.3), our method consistently achieves the best TPR, FPR, ASR, and ACC.

	No Attack			C-brd (5%)				C-Squ (5%)				CTRL (1%)			
	ASR ↓	ASR* ↓	ACC ↑	ASR^*_0	ASR* ↓	ACC_0	ACC ↑	ASR^*_0	ASR* ↓	ACC_0	ACC ↑	ASR_0	ASR ↓	ACC_0	ACC ↑
SimCLR	1.78	79	85.4	403	100	84.7	84.8	434	87	84.6	85.0	61.4	2.47	85.3	85.9
MoCo V3	1.88	83	87.2	411	95	87.0	87.1	374	83	87.2	87.13	56.3	3.70	86.5	87.9
BYOL	1.13	50	85.6	455	79	85.5	85.3	446	56	85.2	85.4	39.7	4.36	85.5	85.5
MAE	1.58	70	89.2	83	74	88.4	88.4	104	70	88.65	88.93	15.9	3.42	87.2	89.9

Table 7: Downstream evaluation results of our method under **Case-1**, CIFAR-10. ASR* is the number of successfully attacked samples. ASR^*_0 and ACC_0 with subscripts are the results without defense (i.e., the "No Defense" baseline in other tables). We use ASR* instead of ASR for the C-brd and the C-Squ attack, referring to the original work [20], as their ASRs are naturally low.

	5		10		100		1000	
	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
CTRL (1%)	93.2	0.02	97.4	0.07	95.2	0.34	92.6	2.81

Table 8: # logits used and the detection effects over unlabeled CTRL poisons (**Case-1**, CIFAR-10, SimCLR, ResNet-18).

Remarkably, the averaging performance on both upstream and downstream of ASSET is of magnitude better than the seven baselines. The results highlight that actively introducing different model behaviors can help a detection method to be of better robustness to the DL paradigm shift.

	FT-all				FT-last				Average		Worst-Case	
	BadNets (20%)		SAA (5%)		Blended (20%)		HTBA (5%)					
(a) Upstream Evaluation												
	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
Spectral	82.3	16.9	39.2	5.83	11.6	34.6	53.6	5.07	46.7	15.6	11.6	34.6
Spectre	85.1	16.2	53.6	4.54	68.5	20.4	74.4	3.98	70.4	11.3	53.6	20.4
Beatrix	64.4	19.1	66.8	3.96	13.1	31.4	89.6	3.50	58.5	14.5	13.1	31.4
AC	21.6	46.3	57.2	32.5	0.60	46.9	41.6	34.4	30.3	40.0	0.60	46.9
ABL	59.8	22.6	48.4	5.35	49.3	25.2	61.2	4.67	54.7	14.5	48.4	25.2
STRIP	92.3	10.6	25.6	8.23	67.1	16.8	35.6	8.70	55.2	11.1	35.6	16.8
CT	94.6	10.4	78.0	7.24	0.00	0.00	82.4	3.49	63.8	5.33	0.00	10.4
Ours	98.7	1.03	95.2	0.51	99.2	0.10	95.6	0.34	**97.2**	**0.50**	**95.2**	**1.03**
(b) Downstream Evaluation												
	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
No Def.	97.5	91.3	98.7	92.3	93.9	71.4	56.4	72.8	86.6	82.0	98.7	71.4
Spectral	97.4	91.5	80.2	91.8	91.4	68.7	16.9	72.1	71.5	81.0	97.4	68.7
Spectre	95.8	91.8	75.9	91.9	92.5	69.8	10.9	72.3	68.8	81.5	95.8	69.8
Beatrix	96.0	91.7	68.9	92.0	92.7	67.6	5.50	72.6	65.8	81.0	96.0	67.6
AC	97.4	86.7	73.2	88.7	93.3	65.4	21.4	66.1	71.3	76.7	97.4	65.4
ABL	96.4	91.7	80.1	92.0	93.7	68.3	14.2	72.2	71.1	81.1	96.4	68.3
Strip	94.4	91.8	87.0	91.9	92.9	70.8	24.3	71.3	74.7	81.5	94.4	70.8
CT	93.2	91.8	18.6	91.9	93.9	71.4	8.60	72.5	53.6	81.9	93.9	71.4
Ours	10.2	92.9	8.40	92.3	16.2	74.8	3.40	72.8	**9.55**	**83.2**	**16.2**	**72.8**

Table 9: (a) Upstream and (b) Downstream Evaluation and comparison results under **Case-2** with CIFAR-10: The first row denotes the TL strategy. The **bolded** results denote the best defense results among all defenses. We highlight the ASR below 20% in **blue** as a success defense, the ASR above 20% in **red** as a failed defense case.

5.5 Adaptive Attack Analysis

From the above, we find ASSET is the most reliable detection method across different attacks, datasets, poison ratios, and training paradigms. Now we study adaptive attacks, where we want to understand how an attacker's knowledge about defense implementation impacts defense performance.

Attacker goal & settings. The attacker aims to craft poisoned samples resulting in a low TPR while maintaining a low FPR for upstream detection , and resulting in a high ASR while maintaining a high ACC for the downstream poisoned model. A successful adaptive attack should achieve satisfying results based on these metrics simultaneously. We consider two models of attack knowledge: *White-box* attack and *Gray-box* attack. (1) *White-box* Settings. The attacker has full access to the details of ASSET, namely, the workflow of ASSET; the architecture of the detector model, and the architecture of the feature extractor; the architecture of the weighting network will be used for poison concentration; the original poisoned dataset, D_{poi}; and the clean base set D_{b}. Although such disclosure of the defense details is rare in practice, an investigation of this setting gives insights into the worst-case performance of ASSET. (2) *Gray-box* Settings. We also consider a more realistic attack scenario where the attacker is aware of the ASSET pipeline and the respective datasets but not aware of the specific model architectures used by the defender for conducting the detection and performing downstream tasks. In both *White-box* attack and *Gray-box*, the attacker updates the original poisoned samples in D_{poi} and then supplies the updated dataset to the defender.

Attack design. For both *White-box* and *Gray-box* attack, we investigate optimization-based techniques to design poisoned samples to evade ASSET. The attacker can use D_{poi} and D_{b} to obtain trained detector parameters, θ_I and then resolve the following optimization to obtain an additive noise for each poisoned sample x_{poi} in D_{poi} to evade the detection

$$\delta^* = \arg\min_{\delta} \mathcal{L}_{\max}\left(f(x_{\text{poi}}+\delta|\theta_I)\right), \tag{6}$$

where $\mathcal{L}_{\max}$ is inherited from Eqn. (2). Recall that ASSET optimizes θ so that poisoned samples are assigned with large loss values while clean samples are assigned with small loss values. The above formulation manipulates one poisoned sample, x_{poi}, such that the trained detector will assign low loss values to $x_{\text{poi}}+\delta^*$, which helps disguise the poison. To resolve the proposed adaptive attack formulation in Eqn (6), we conduct gradient descent 100 steps for each example. Visual examples of the adaptive attack manipulated poisons for the attacks considered in **Case-0** are depicted in Figure 7, Appendix 6.3. After the update of D_{poi}, we obtain new model parameters (i.e., feature extractor $\tilde{\theta}^*_{\text{poi}}$, detector $\tilde{\theta}_I$, and weighting network $\tilde{M}$) on the updated D_{poi} and evaluate the attack

	Dirty-Label Backdoor Attacks								*Clean-Label Backdoor Attacks*					
	BadNets (5%)		**Blended (5%)**		**WaNet (10%)**		**ISSBA (1%)**		**LC (1%)**		**SAA (1%)**		**Narci. (0.05%)**	
(a) Upstream Evaluation														
	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
White-box	60.6	1.47	98.1	0.49	65.3	5.41	80.6	0.03	41.4	37.3	85.4	0.14	36.0	17.6
Gray-box	99.7	0.22	99.6	0.18	83.4	4.18	90.6	0.08	98.6	0.57	96.4	47.1	100	0.03
(b) Downstream Evaluation														
	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
White-box Adaptive Attack														
No Defense	93.1	93.5	83.7	93.9	41.2	92.9	84.1	93.8	95.6	94.2	34.2	93.7	25.3	94.7
Ours	58.3	94.5	8.41	94.1	11.4	93.3	22.5	94.4	20.3	93.9	2.35	94.4	5.49	94.9
Ours + Unlearn	3.21	89.4	0.46	91.2	2.45	88.7	0.87	93.6	0.53	71.2	0.63	92.3	1.21	92.0
Gray-box Adaptive Attack														
No Defense	91.3	90.5	88.5	91.5	83.6	89.8	26.2	90.3	64.6	91.0	15.2	91.1	22.4	91.1
Ours	6.23	90.9	4.35	90.6	8.64	89.0	1.23	90.3	8.57	91.1	1.06	91.1	1.34	91.1

Table 10: (a) Upstream and (b) Downstream evaluation results for the adaptive attacks. We consider the same attacks from **Case-0**, CIFAR-10, and implement the white-box adaptive attack to disguise the original poisoned samples.

performance following the aforementioned attack settings. For the *White-box* attack setting, we evaluate the downstream with the same model structures as used by the attacker for synthesizing δ^*. For the *Gray-box* attack setting, we use different model structures.

Results and insights. The results of ASSET against the adaptive attacks are summarized in Table 10. From the upstream evaluation, for *White-box* attack, we find the adaptive attack's effect varies from trigger to trigger. The performance of the *White-box* attack on disguising Blended triggers is limited, while on BadNets, LC, and Narci., the TPR is largely decreased. Interestingly, the model mismatch introduced in the *Gray-box* largely impacts the attack efficacy and ASSET is able to maintain high defense performance across all the *Gray-box* attacks. While moving on to the downstream evaluation, we find both *White-box* and *Gray-box* adaptive attack introduced additional noise that impedes some of the backdoor triggers from taking effect, i.e., lower ASR at the end, even without any additional defensive measure. For *White-box* attack, we find only the adaptive BadNets attack can achieve an ASR greater than 50% after the model converges over the subset removing the detected samples using ASSET. By following the standard procedure in many detection-based defenses [11,23], we use the detected samples to provide revered gradients for the downstream model (e.g., minimize negative CE loss) or known as Unlearning, denoted by "Ours+Unlearn". We find this simple adaptation of ASSET can successfully diminish the effect of all the evaluated *White-box* adaptive attacks. On the other hand, the ASSET on the *Gray-box* adaptive attacks with detector model mismatch (attacker uses ResNet-18 to obtain θ_I, defender uses VGG-16 to obtain $\tilde{\theta}_I$) are almost the same on the vanilla attacks without adaptation.

To conclude, the above study shows that ASSET is robust to the evaluated *White-box* attack with the standard unlearning procedure using the detected samples and robust to the evaluated *Gray-box* attack. The results highlight that disclosing the knowledge of our defense workflow and models can expose ASSET to the risk of adaptive attacks. Not releasing the model architecture can mitigate the risk of adaptive attacks to a large extent. Also, using the detected samples for unlearning can be a simple yet effective post-processing method that can be used in tandem with our detection to safeguard ML applications against adaptive attacks to our defense. One thing worth highlighting is that the unlearning process requires the detection method to obtain a better precision upstream. Otherwise, if the FPR of the upstream is high (more clean samples are wrongly flagged), the downstream unlearning would result in an unfavorable impact on the ACC (e.g. the results on the *White-box* LC results).

6 Conclusion

This work is motivated by the glaring gap between the focused evaluation of the end-to-end SL settings in prior backdoor detection literature and the fast adaption of other more data- and computation-efficient learning paradigms, including SSL adaptation and TL. We find that existing detection methods cannot be applied or suffer limited performance for SSL and TL; even for the widely studied end-to-end SL setting, there is still large room to improve detection in terms of their robustness to variations in poison ratio. This work proposes a novel idea for actively enforcing different model behaviors on clean and poisoned samples through a two-level nested offset loop. Our approach provides the first backdoor defense that operates across different learning paradigms, different attack techniques, and poison ratios.

Our work opens up many directions for future work. (1) *Theoretical Understanding of Offset*: Despite the empirical success, an in-depth understanding of convergence behaviors and sample complexity of ASSET is still lacking. In addition, we have shown multiple offset objectives, but how to explain why a loss design is better than the other is still an open question. (2) *Alternative Offset Goal Designs*: Our work provides a general algorithmic framework for active backdoor data detection by optimizing opposite goals. Are there other optimization objectives beyond what we proposed in this paper that can lead to better detection performance? (3) *Extension to Broader Data Types*: Evaluating ASSET on domains beyond images and texts is of practical importance.

Acknowledgement

RJ and the ReDS lab appreciate the support of the Amazon - Virginia Tech Initiative for Efficient and Robust Machine Learning and the Cisco Award. YZ is supported by the Amazon Fellowship. XL gratefully acknowledges the support of National Science Foundation Award No. CNS-1929300.

References

[1] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[2] Y. Zeng, M. Pan, H. A. Just, L. Lyu, M. Qiu, and R. Jia, "Narcissus: A practical clean-label backdoor attack with limited information," *ACM CCS*, 2023.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[4] Y. Li, B. Wu, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *arXiv:2007.08745*, 2020.

[5] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of {DNNs} for robust backdoor contamination detection," in *USENIX Security*, 2021, pp. 1541–1558.

[6] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *ACM ACSAC*, 2019.

[7] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NeurIPS*, 2018, pp. 8000–8010.

[8] J. Hayase, W. Kong, R. Somani, and S. Oh, "Spectre: defending against backdoor attacks using robust statistics," in *ICML*, 2021.

[9] W. Ma, D. Wang, R. Sun, M. Xue, S. Wen, and Y. Xiang, "The" beatrix"resurrections: Robust backdoor detection via gram matrices," in *NDSS Symposium*, 2022.

[10] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv:1811.03728*, 2018.

[11] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: Training clean models on poisoned data," in *NeurIPS*, vol. 34, 2021.

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020, pp. 1597–1607.

[13] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *CVPR*, 2022, pp. 16 000–16 009.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[15] J. Z. HaoChen, C. Wei, A. Kumar, and T. Ma, "Beyond separability: Analyzing the linear transferability of contrastive representations to related subpopulations," *arXiv:2204.02683*, 2022.

[16] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," in *NeruIPS*, 2020.

[17] X. Chen, S. Xie, and K. He, "An empirical study of training self-supervised vision transformers," in *CVPR*, 2021.

[18] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," in *NeurIPS*, vol. 33, 2020, pp. 21 271–21 284.

[19] N. Carlini and A. Terzis, "Poisoning and backdooring contrastive learning," in *ICLR*, 2022.

[20] A. Saha, A. Tejankar, S. A. Koohpayegani, and H. Pirsiavash, "Backdoor attacks on self-supervised learning," in *CVPR*, 2022, pp. 13 337–13 346.

[21] C. Li, R. Pang, Z. Xi, T. Du, S. Ji, Y. Yao, and T. Wang, "Demystifying self-supervised trojan attacks," *arXiv:2210.07346*, 2022.

[22] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer." *J. Mach. Learn. Res.*, 2020.

[23] X. Qi, T. Xie, J. T. Wang, T. Wu, S. Mahloujifar, and P. Mittal, "Towards a proactive ml approach for detecting backdoor poison samples," 2023.

[24] Y. Zeng, M. Pan, H. Jahagirdar, M. Jin, L. Lyu, and R. Jia, "Meta-sift: How to sift out a clean subset in the presence of data poisoning?" 2023.

[25] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012.

[26] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.

[27] Z. Xie, Y. Lin, Z. Yao, Z. Zhang, Q. Dai, Y. Cao, and H. Hu, "Self-supervised learning with swin transformers," *arXiv:2105.04553*, 2021.

[28] J. Gui, T. Chen, Q. Cao, Z. Sun, H. Luo, and D. Tao, "A survey of self-supervised learning from multiple perspectives: Algorithms, theory, applications and future trends," *arXiv preprint arXiv:2301.05712*, 2023.

[29] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," in *arXiv:1712.05526*, 2017.

[30] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *NDSS*, 2018.

[31] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in *USENIX Security*, 2021, pp. 1505–1521.

[32] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *ICCV*, 2021.

[33] S. Li, M. Xue, B. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE TDSC*, 2020.

[34] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *ECCV, 2020*. Springer, 2020, pp. 182–199.

[35] T. A. Nguyen and A. T. Tran, "Wanet-imperceptible warping-based backdoor attack," in *ICLR*, 2020.

[36] Y. Zeng, W. Park, Z. M. Mao, and R. Jia, "Rethinking the backdoor attacks' triggers: A frequency perspective," in *ICCV*, 2021.

[37] H. A. A. K. Hammoud and B. Ghanem, "Check your other door! establishing backdoor attacks in the frequency domain," *arXiv:2109.05507*, 2021.

[38] T. Wang, Y. Yao, F. Xu, S. An, H. Tong, and T. Wang, "An invisible black-box backdoor attack through frequency domain," in *ECCV*, 2022.

[39] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," *arXiv:1912.02771*, 2019.

[40] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *AAAI*, 2020.

[41] H. Souri, M. Goldblum, L. Fowl, R. Chellappa, and T. Goldstein, "Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch," *arXiv:2106.08970*, 2021.

[42] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020, pp. 9729–9738.

[43] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv:2003.04297*, 2020.

[44] N. Peri, N. Gupta, W. R. Huang, L. Fowl, C. Zhu, S. Feizi, T. Goldstein, and J. P. Dickerson, "Deep k-nn defense against clean-label data poisoning attacks," in *ECCV*, 2020.

[45] E. Soremekun, S. Udeshi, and S. Chattopadhyay, "Exposing backdoors in robust machine learning models," *arXiv:2003.00865*, 2020.

[46] A. Chan and Y.-S. Ong, "Poison as a cure: Detecting & neutralizing variable-sized backdoor attacks in deep neural networks," *arXiv:1911.08040*, 2019.

[47] E. Chou, F. Tramer, and G. Pellegrino, "Sentinet: Detecting localized universal attacks against deep learning systems," in *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020, pp. 48–54.

[48] T. Wang, Y. Zeng, M. Jin, and R. Jia, "A unified framework for task-driven data quality management," *arXiv:2106.05484*, 2021.

[49] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *ICML*, 2017.

[50] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021.

[51] M. J. Wainwright, *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*, ser. Cambridge Series in Statistical and Probabilistic Mathematics, 2019.

[52] G. Brys, M. Hubert, and P. Rousseeuw, "A robustification of independent component analysis," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 19, no. 5-7, pp. 364–375, 2005.

[53] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[54] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR, 2011.

[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[57] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[58] Y. Zeng, S. Chen, W. Park, Z. Mao, M. Jin, and R. Jia, "Adversarial unlearning of backdoors via implicit hypergradient," in *ICLR*, 2022.

Appendix

6.1 Detailed Defense Settings

In the evaluation section, we provide a thorough comparison of existing backdoor detection techniques. These methods can be classified into several categories, including Spectral [7], Spectre [8], and Beatrix [9], which utilize analysis of activation patterns; AC [10], which leverages clustering of feature information; ABL [11], which detects the lowest loss from poisoned datasets; Strip [6], which focuses on logits of sample outputs; and CT [23], which employs confusion training in end-to-end supervised learning settings.

Note that the above baseline defenses were only evaluated under the settings of end-to-end SL (**Case-0**) in their original papers. They can also be directly generalized to **Case-2**. We will incorporate the above seven baseline defenses in **Case-0** and **Case-2** with the suggested hyperparameters proposed in these original works for comparison. As for **Case-1**, some of the methods are not applicable, whereas others can be adapted to operate without label information. In particular, Strip [6] and CT [23] are label-information-dependent methods, which are excluded from evaluation in **Case-1**. The vanilla design of Spectral [7] and Spectre [8] used a feature extractor trained with label information. In our **Case-1** experiment, we replace the feature extractor trained with labels with one trained using the SSL paradigm. The original implementation processes samples class-wisely for the Beatrix [9] and AC [10]. However, since there is no label information in **Case-1**, we process all training samples together. For ABL [11], we replace the original implementation's Cross-Entropy loss with the respective training loss function used in the respective SSL algorithm (e.g., the InfoNCE loss for the MoCo V3 [17]).

6.2 Detailed Attack Settings

In this work, we examine several representative attacks for each category of attack design. For **Case-0**, which is the end-to-end supervised learning setting mentioned in Section 5.2, we thoroughly investigate existing Dirty-label attacks and Clean-label backdoor attacks. Dirty-label attacks create a backdoor by altering the label of the poisoned samples to the target class. We selected some representative attacks for experiments. For example, BadNets [1] and Blended [29] are used as triggers by simply superimposing special patterns; there are also affine transformations that are difficult to find on pictures, such as WaNet [35]; as well as training an encoder to create distinct backdoor trigger for each sample like ISSBA [33]. On the other hand, Clean-label backdoor attacks maintain the original label of the poisoned samples. Examples include LCciteturner2019label, which makes models learn simple triggers by patching adversarial noise on the remaining part of sample; SAA [41], which produces effects through model feature collisions; and the state-of-the-art attack Narcissus [2], which obtains the backdoor trigger by optimizing the distribution within the class and the connection of the target label. For these three Clean-label backdoor attacks, we set $l_\infty = 16/255$ to ensure the consistency of the attack. For **Case-1**, we consider the backdoor attack in the SSL setting (detailed in Section 5.3). Since the training does not require labels and always contains strong augmentations, traditional attacks against SSL are not effective. However, with the development of this training paradigm, attacks against it have started to emerge. There are attacks by superimposing specific design patterns [19, 20] and attacks by adding specific frequency noise to the YCbCr color space [21]. The C-brd and C-Squ adopt a fixed in-class poison ratio w.r.t. only the samples from the targeted category (50% in-class), following [20]. CTRL adopts a fixed poison ratio w.r.t. the whole dataset (1% of all the samples), following [21]. For **Case-2**, we investigate the attacks in the context of transfer learning, as described in section 5.4. Our evaluation revealed that adding backdoor attack samples to the fine-tuned dataset leads to a successful attack. Basic backdoor attacks, such as BadNets and Blended, can easily be generalized and result in an effective attack. Furthermore, attacks based on the collision of the model's feature space, such as SSA or HTBA [40] can also work in this scenario. All the attacks use the default settings in the original paper to ensure consistency with the original work.

6.3 Additional results

In addition to the results presented in the main text, we also evaluate the performance of the baseline defenses in different attack settings and dataset settings.

Additional Results with Multiple Attacks. For **Case-0**, we test the scenario where multiple backdoor attacks appear simultaneously in a training set. We deploy 4 different dirty label attacks that have appeared in the main text into 4 different classes of the CIFAR-10 dataset, and the poison ratio is consistent with the main text. At the same time, the ASR of all attacks is above 90% to ensure the effectiveness of the attack. The results are listed in Table 14. When multiple attacks are present, all the baseline defense methods except CT can maintain a reliable detection, as at least one set of poisoned samples ends up with a TPR lower than 50%. Our method achieves the highest average TPR among all defenses and demonstrates a better and more consistent detection performance with all the TPR above 85% under this setting.

Additional Results with SSL. For **Case-1**, we evaluate the re-

	No Attack			C-brd (5%)				C-Squ (5%)				CTRL (1%)			
	ASR ↓	ASR* ↓	ACC ↑	ASR^*_0	ASR* ↓	ACC_0	ACC ↑	ASR^*_0	ASR* ↓	ACC_0	ACC ↑	ASR_0	ASR ↓	ACC_0	ACC ↑
SimCLR	0.34	6	67.2	168	8	65.9	66.9	141	12	65.6	66.8	25.0	1.36	66.1	66.8
MoCo V3	0.32	6	68.4	67	8	68.0	68.2	119	12	67.8	68.2	23.6	2.12	68.2	68.3
BYOL	0.36	7	67.1	290	11	66.6	67.1	263	19	66.8	66.9	40.9	1.44	66.5	66.8
MAE	0.28	5	70.2	28	6	68.9	70.1	68	8	69.1	69.9	30.7	1.66	68.7	69.3

Table 11: Downstream evaluation results of our method under **Case-1** in ImageNet-100.

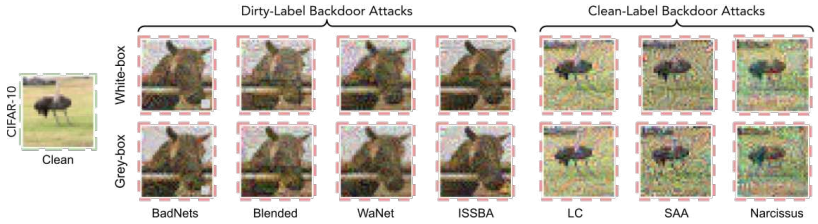

Figure 7: Visual examples of the backdoor poisoned samples disguised by adaptive attacks (**Case-0**, CIFAR-10).

sults on the ImageNet-100 dataset. ImageNet-100 is a subset of ImageNet-1K, consisting of 100 randomly selected classes (about 128,000 samples), which is currently the most popular benchmark dataset for self-supervised learning. All images are resized to 224x224 pixels to fit the model input. Here we use self-supervised learning methods consistent with those in Section 4.3, including the contrastive learning method SimCLR, MoCO V3, BYOL, and the masked-model training method MAE. Here all backbone models are ViT-Small/16 to obtain a satisfactory ACC. The upstream and downstream results can be found in Table 15, and Table 11, respectively. As the dataset becomes more complex compared to CIFAR-10, detection also becomes more difficult. Nevertheless, our method provides a TPR greater than 88% in all cases. All FPRs are below 0.5%, providing as clean samples as possible for subsequent downstream tasks and minimizing the impact on ACC. In the downstream task, our method succeeded in reducing the ASR with no significant improvement over the baseline without poison, indicating that our method was successful in removing the poison. At the same time, thanks to the extremely low FPR, the ACC of the model has seen a certain increase compared to the poisoned model.

Additional Results with TL. Finally, in **Case-2**, we present the upstream and downstream results of STL-10 in Table 16, where all images were scaled to 224x224 pixels to align with the ImageNet-1K [55] pre-trained ViT-Tiny/16 [14] model. Our method consistently achieves a TPR of over 90%, while keeping the FPR below 0.6%. Compared to other defense methods, our method achieves the best average TPR and FPR. In the downstream tasks, which benefited from the high TPR and low FPR, our method successfully keeps all ASRs below 20%, ensuring attacks will not effectively occur. Our method obtains the highest average value for ACC as well as ASR.

Visual Results of Adaptive Attacks. Figure 7 depicts the visual results of the adaptive attacks discussed in Section 5.5.

	Upstream Evaluation		Downstream Evaluation	
	TPR ↑	FPR ↓	ASR ↓	ACC ↑
AC	79.8	18.1	68.4	90.3
Ours	100	3.77	10.3	91.6

Table 12: Textual backdoor detection, BadNets, SST-2 dataset.

Additional Results on Other Modality. We provide additional results on exploring the applicability of the ASSET on detecting backdoor samples in the Natural Language Processing domain. We implemented the BadNets attack[6] on the SST-2 dataset with BERT [57] as the target model. We set the poisoning rate to be 10%, with the trigger as "cf mn bb tq." We observe that ASSET can achieve good detection results. Compared to the AC evaluated under the same settings, we find our method provides more effective detection results. One possible explanation for the AC's limited effectiveness is that the BERT model relies on pre-trained features, which limits the separability based on feature space clustering.

Computation Overhead. Table 13 compares the computation overhead of ASSET and other baseline methods in **Case-0**.

	Spectral	Spectre	Beatrix	AC	ABL	Strip	CT	Ours
CIFAR-10	1800+63	1800+137	1800+782	1800+123	847	1800+374	6300	1800+1800

Table 13: Computational overhead (GTX 2080 Ti GPU seconds) under (**Case-0**). Defense methods rely on a pre-trained poisoned model incur additional 1800s for training.

6.4 Ablation Study

Table 17 shows that solely adopting the outer offset loop will experience limitations in low poison ratio cases. In the case of a low poison ratio, since the poison samples account for a relatively small proportion in each mini-batch, the model will tend to optimize its output for clean samples, thus ignoring its output for poison samples, finally leading to limited performance. However, this limitation can be effectively overcome by embedding an inner loop to perform poison concentration. In a relatively high poison ratio setting (e.g., 20%) where the outer loop alone can already achieve good detection performance, inserting an inner loop is still useful and can further boost the detection efficacy. It can be seen that the design of the inner loop is the key to our successful defense in spite of the very low poison ratio in Table 3.

We ablate on the size of the base set used in our detection, and the result is provided in Table 18. We find that the detection performance slightly decreases as the base set size is smaller; nevertheless, ASSET can achieve strong performance even with 10 samples—one sample per class on CIFAR-10. Our experiment confirms our conclusion in Section 4.1 that the base set and the clean portion of the poisoned

[6] https://github.com/thunlp/OpenBackdoor

	FPR	BadNets (5%) Class 0	WaNet (10%) Class 4	ISSBA (1%) Class 6	Blended (5%) Class 9	Average	Worst-Case
		TPR	TPR	TPR	TPR	TPR	TPR
Spectral	27.2	88.6	0.00	98.4	92.6	69.9	0.00
Spectre	26.5	94.6	0.16	84.2	98.9	69.5	0.16
Beatrix	2.61	92.8	50.9	42.6	75.9	65.5	42.6
AC	42.4	58.9	79.2	4.60	53.3	49.0	4.60
ABL	33.7	89.9	0.64	0.00	6.72	24.3	0.00
Strip	11.2	83.6	6.82	45.4	51.7	46.9	6.82
CT	1.22	99.6	96.5	81.8	96.8	93.7	81.8
Ours	0.36	99.7	86.8	94.2	100	**95.2**	**86.8**

Table 14: Defense results on multi-trigger-multi-target attack under **Case-0**, FPR refers to the overall FPR in the training dataset. The **bolded** results denote the best defense results among all the evaluated defenses w.r.t. each attack.

	C-brd (0.5%)		C-Squ (0.5%)		CTRL (1%)	
	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
SimCLR	95.7	0.17	92.9	0.20	92.9	0.34
MoCo V3	92.5	0.21	90.3	0.23	91.4	0.10
BYOL	90.5	0.15	88.6	0.18	97.4	0.21
MAE	97.8	0.17	94.2	0.27	98.8	0.17

Table 15: Further upstream evaluation of our method under **Case-1** with four training algorithms under ImageNet.

	FT-all				FT-last				Average			Worst-Case
	BadNets (20%)		SAA (5%)		Blended (20%)		HTBA (5%)					
(b) Upstream Evaluation												
	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
Spectral	54.4	23.9	11.2	7.31	16.6	33.4	25.6	4.97	27.0	17.4	11.2	33.4
Spectre	76.8	18.3	17.2	6.99	16.0	33.5	46.4	3.87	39.1	15.7	16.0	33.5
Beatrix	86.9	3.55	74.8	12.1	56.7	11.7	89.2	13.5	76.9	10.2	56.7	13.5
AC	34.6	46.2	18.0	14.5	9.80	60.3	8.40	13.3	17.7	33.6	8.40	60.3
ABL	81.2	17.2	57.2	4.88	75.3	18.7	75.6	3.91	72.3	11.2	57.2	18.7
Strip	83.2	11.2	0.00	20.7	52.9	16.3	71.2	17.6	51.8	16.5	0.00	20.7
CT	98.3	10.5	82.4	3.98	98.7	6.58	96.4	2.57	94.0	5.91	82.4	10.5
Ours	97.7	0.53	90.8	0.34	99.6	0.18	99.2	0.19	**96.8**	**0.31**	**90.8**	**0.53**
(b) Downstream Evaluation												
	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
No Def.	99.6	97.9	93.6	98.6	97.7	98.5	68.1	98.5	89.8	98.4	99.6	97.9
Spectral	99.5	97.6	91.4	98.1	97.6	98.4	49.6	98.5	84.5	98.2	99.5	97.6
Spectre	99.3	98.0	86.3	98.1	97.6	98.4	31.3	98.5	78.6	98.3	99.3	98.0
Beatrix	99.4	98.0	36.2	97.9	95.2	98.6	8.93	98.5	59.9	98.3	99.4	97.9
AC	99.6	97.3	85.6	98.0	98.4	98.2	56.3	98.5	85.0	98.0	99.6	97.3
ABL	99.6	97.1	59.6	98.2	94.2	98.5	14.3	98.5	66.9	98.1	99.6	97.1
Strip	99.5	97.7	94.0	97.8	98.4	98.4	16.8	98.4	77.2	98.1	99.5	97.7
CT	7.65	98.1	11.2	98.2	90.2	98.4	1.27	98.5	27.6	98.3	90.2	**98.1**
Ours	8.93	98.1	15.4	98.1	1.44	99.2	0.36	98.5	**6.53**	**98.5**	**15.4**	**98.1**

Table 16: Upstream and Downstream Evaluation and comparison results under **Case-2** with STL-10.

	BadNets (5%)		BadNets (20%)	
	TPR	FPR	TPR	FPR
Outer loop only	39.0	37.3	96.6	0.83
Outer + Inner	**99.5**	**5.24**	**99.9**	**0.03**

Table 17: Detection effects w/ or w/o inner loop (**Case-0**).

dataset share the same clean distribution, while the clean sample and poison sample originate from distinct distributions.

Figure 8 depicts AO's impact on mini-batches from the same poisoned training set. In particular, even though the two mini-batches are from the same distribution, the number of poisoned samples varies due to random sampling. With different sizes of poisoned samples resulting in different distributions of the loss values, it becomes harder for the inner loop to use a fixed threshold or fixed ratio to determine the most likely poisoned samples to form B_{pc}. AO helps to map the distribution adaptively so that we find a fixed threshold to consistently obtain the poison-concentrated subset.

	10		100		1000		5000	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
BadNets (5%)	98.2	1.04	98.9	1.0	99.5	0.55	99.5	0.22
Blended (5%)	100	0.18	99.9	0.01	100	0.00	100	0.00

Table 18: Ablation study in the base set size (**Case-0**).

6.4.1 Impact of Base Set Quality on Detection Efficacy

While ASSET exhibits robust performance across a range of attack settings, its effectiveness may fluctuate depending on the quality of the base set.

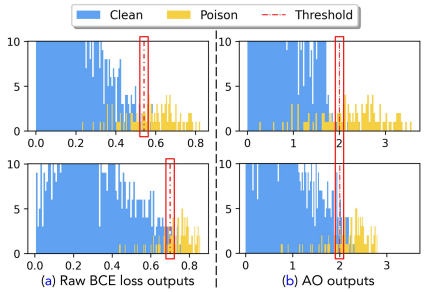

Figure 8: The original BCE loss output **(a)** and the output processed after AO **(b)** (WaNet attack, CIFAR-10, ResNet-18, **Case-0**). In particular, AO maps the original outputs to a more separable range which is easier to concentrate the poisoned samples with a fixed threshold.

	CIFAR-10		CIFAR-100		STL-10		GTSRB	
	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
BadNets (5%)	99.5	0.55	98.6	0.81	87.3	3.21	0.00	100.0

Table 19: Use non-iid dataset as the base set (**Case-0**). The CIFAR-10 column represents the iid setting.

Sampling quality of the base set. In this paper, the base set follows the widely accepted setting [23, 58] that it is drawn from the same distribution as the training set. However, it is worth noting that in practical, a distributional drifts may occur between the training and base sets. To test how ASSET fares in the face of such distributional drifts, we have outlined the detection results derived from utilizing samples taken from different datasets as base sets for poison detection on CIFAR-10 (BadNets attack, **Case-0**) in Table 19. Our observations suggest that ASSET can consistently generate acceptable detection results if the distributional drift does not drastically alter the task context, as evidenced by the results from CIFAR-100 and STL-10. However, the detection efficiency falters when an out-of-distribution dataset is used as the base set, as exemplified by the use of the traffic sign dataset, GTSRB.

	0/1000		1/1000		5/1000		10/1000	
	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓	TPR ↑	FPR ↓
BadNets (5%)	99.5	0.55	85.4	1.27	78.7	0.00	9.16	3.39

Table 20: Number of poisoned samples in the base set (**Case-0**). The 0/1000 column represents the clean base set.

Poisons in the base set. Stronger attack settings may enable attackers to tamper with the base set. Implementing this setting is challenging, and it has rarely been discussed in prior work due to the formidability of embedding the exact trigger into the carefully scrutinized base set without triggering any alerts. We evaluate the impact of different poison ratios in the base set in Table 20, and with 10 poisoned samples infiltrating the base set will cause the detection to be ineffective.

Remark. The above results on the efficacy and the base set quality are unsurprising. The detection efficacy's sensitivity to the quality of the base set is not exclusive to ASSET. This sensitivity is likewise a noted drawback of numerous defensive methods that rely on a clean in-distribution base set, as observed and discussed in [24]. The experimental results highlight the importance of obtaining high-quality base sets with the care of drift and security inspections. How to effectively acquire a high-quality base set is out of the scope of this paper.