TRAP: Mitigating Poisoning-based Backdoor Attacks by Treating Poison with Poison

Chun Li, Zhong Li, Minxue Pan, Xuandong Li

***Abstract*—The backdoor attack poses a significant threat to deep neural networks. Existing works on poison suppression defense mainly focus on differentiating between poisoned and benign samples based on various metrics and removing the backdoor using Unlearning. However, these metrics can be bypassed by certain attacks, and Unlearning often leads to sub-optimal model performance. Through examination of the model attack process, we discovered that poisoned samples always form clusters distant from benign samples in the early stages of training and when the model is fully trained, there is a unique pathway within the classifier connecting backdoor features to the target label. Leveraging these observations, in this paper, we propose a novel training method to detect poisoned samples during the early stages of training and remove the backdoor by retraining the classifier part of the model on relabeled poisoned samples. We evaluated our method against twelve attacks on four datasets, and the results showed that our method significantly outperforms existing state-of-the-art defenses. We reduced the average attack success rate to 0.07% while only decreasing the average accuracy by 0.33%. Our code is available at https://anonymous.4open.science/r/TRAP-2672.**

***Index Terms*—Backdoor Attack, Trustworthy ML, AI Security**

I. INTRODUCTION

DEEP Neural Networks (DNNs) have been extensively used in various applications, including critical domains where trustworthiness is a concern, e.g., self-driving cars [1], healthcare [2], and finance [3]. The training of DNN models in general requires a large amount of training data. However, obtaining massive high-quality training data is a challenging task in practice. Thus, some individuals or companies may resort to using datasets from untrustworthy third-parties [4].

While untrustworthy sources of data provide a convenient way for developers to build their DNN models, they also pose significant security threats. One of the typical threats is the poisoning-based backdoor attack [5], wherein the attackers inject triggers into a small portion of benign samples and relabel these samples as the attacker-specific target label. Then, by training a DNN model on the poisoned training dataset, the mapping between the backdoor trigger and the target label, that is, the backdoor, is implanted into the model. In the inference time, the backdoored model predicts normally on benign samples but predicts any inputs with the backdoor trigger as the attacker-specific target label.

To counter the threats of backdoor attacks, researchers have developed numerous empirical defense methods. Among these, *Poison suppression-based defense* that has recently achieved promising results in the field [6]. Poison suppression-based defenses try to prevent backdoor creation by suppressing the effects of poisoned samples during the training process [7]–[14]. For example, ABL [7] differentiates between poisoned and benign samples by analyzing their losses and eliminates backdoors by conducting machine unlearning on these detected poisoned samples; DBD [10] defends against backdoor attacks by decoupling the training of the feature extractor and the classifier; NONE [11] enforces non-linear decision regions during training to mitigate backdoors.

Although the effectiveness of existing poison suppression-based defenses has been demonstrated, they suffer from several limitations. Particularly, the current criteria used in existing defenses may be ineffective in detecting poisoned samples under certain attacks. For instance, DBD [10] employs the SCE loss, which is originally designed for learning with noise, to filter poisoned samples. Consequently, DBD may fail to defend against clean-label attacks [15], [16] where the poisoned and benign samples exhibit similar SCE losses. Failures in the detection process would further affect backdoor removal. For example, treating benign samples as poisoned samples not only fails to eliminate backdoors completely but also impacts the model's performance. In addition, the current poison suppression-based defenses mainly remove backdoors in the model through machine unlearning [17], [18] by maximizing the error between model prediction and poisoned sample labels. Although such removal methods can remove backdoors, they often result in sub-optimal classification accuracy and can even render the model completely unusable [18]. Therefore, to address these limitations, we propose a novel defense method called TRAP which achieves state-of-the-art performance against poisoning-based backdoor attacks while maintaining high accuracy in learned models.

Observations & Insights. To start with, we conduct an in-depth study of the underlying mechanisms of poisoning-based backdoor attacks, which involves a thorough analysis of attack behaviors during model training within the hidden space. Our analysis reveals several key insights. **First**, the backdoor trigger is a strong feature of the target label. As a result, it

This research is supported by the National Natural Science Foundation of China (62372227 and 62402214), the Natural Science Foundation of Jiangsu Province (BK20241194), the China Postdoctoral Science Foundation (2025T180420), the Postdoctoral Fellowship Program of CPSF (GZB20250386), the Jiangsu Funding Program for Excellent Postdoctoral Talent (2025ZB317), and the Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX25_0368).

Chun Li, Zhong Li, and Minxue Pan are with the State Key Laboratory for Novel Software Technology and the Software Institute of Nanjing University, China (Email: chunli@smail.nju.edu.cn, lizhong@nju.edu.cn, mxp@nju.edu.cn).

Xuandong Li is with the State Key Laboratory for Novel Software Technology and the Department of Computer Science and Technology of Nanjing University, China (Email: lxd@nju.edu.cn).

is easier to learn the poisoned samples than the benign ones. This leads to the formation of a distinct and suspect cluster of poisoned samples that can be differentiated from the benign samples in the initial stages of training. **Second**, the poisoned samples have unique features compared to the benign ones. Transitively, there is a unique path for associating the backdoor features with the target labels, which differs significantly from the paths between the benign features and the labels of benign samples. More can be found in Section III-B.

TRAP overview. Inspired by the insights mentioned, we propose the "treating poison with poison (TRAP)" approach, utilizing the inherent mechanisms of poisoning-based backdoor attacks against themselves to mitigate their effects and ultimately "trap" attackers. Specifically, TRAP begins by training a DNN model through standard supervised learning. During this process, its *poisoned sample isolation* module inspects a few initial training stages to identify poisoned samples within the training dataset. It clusters the features of training examples during these stages and searches for suspicious clusters that may contain poisoned samples. Upon completing the model training, the *backdoor hiding* module fine-tunes the classifier part of the model to remove the backdoors contained within. It first relabels the detected poisoned samples with a new virtual class to construct a new training dataset, and then fine-tunes the classifier part of the model on the new dataset using standard cross-entropy loss. As such, the backdoors in the model are removed by creating a new virtual path that replaces the original mapping between the backdoor features and target labels.

We assess TRAP's defense performance by comparing it with nine state-of-the-art poison suppression-based defenses across four benchmark datasets and against twelve poisoning-based backdoor attacks. Our experimental results demonstrate that TRAP achieves both effective and efficient defense simultaneously. Specifically, we reduce the average attack success rate to 0.07% at an extremely low cost of decreasing the model accuracy by an average of 0.33%, significantly outperforming other defenses. Our main contributions can be summarized as follows:

- We perform an in-depth analysis of poisoning-based backdoor attack behaviors during model training and uncover insights against a wide range of attacks, revealing distinct differences in distributions and label-connecting pathways for poisoned and benign samples within the hidden space.
- We present TRAP, a novel poison suppression-based defense approach that effectively removes backdoors in attacks while maintaining the performance of learned models.
- We assess TRAP's efficacy through extensive experiments with a diverse range of twelve attack methods across four benchmark datasets. The results demonstrate that TRAP's defense performance significantly outperforms nine state-of-the-art poison suppression-based defenses.
- We have made TRAP publicly available at https://anonymous.4open.science/r/TRAP-2672 to facilitate future research.

II. BACKGROUND

A. Deep Neural Network

In this work, we target Deep Neural Network (DNN) models that perform image classification tasks (e.g., classifying dog and cat images). Given an image domain $\mathcal{X} \in \mathbb{R}^{ch \times W \times H}$ where W is the image width, H is the image height, ch is the number of image channels and a set of classes $\mathcal{Y} = \{1, 2, ..., C\}$ where C is the number of image classes indexed by c, an image classification DNN model refer to a mapping function $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$. That is, the function $\mathcal{F}$ takes an image $\boldsymbol{x} \in \mathcal{X}$ as input and outputs a label $y \in \mathcal{Y}$ for $\boldsymbol{x}$. We view the function $\mathcal{F}$ as $\mathcal{F} = g \circ f$, where f is the feature extractor, and g is the last linear prediction layer that transforms a latent representation into the final prediction label. To derive such a function $\mathcal{F}$, one in general needs to collect a high-quality training dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, consisting of pairs of images and their corresponding labels. With the training dataset D, the function $\mathcal{F}$ is then learned via minimizing the empirical risk $\mathcal{R}_{\mathcal{L}}(\mathcal{F})$ under a loss function $\mathcal{L}$:

$$\mathcal{R}_{\mathcal{L}}(\mathcal{F}) = \frac{1}{|D|} \sum_{(\boldsymbol{x}_i, y_i) \in D} \mathcal{L}(\mathcal{F}(\boldsymbol{x}_i), y_i) \tag{1}$$

where $\mathcal{L}$ is usually the cross-entropy (CE) loss in image classification tasks.

B. Existing Poisoning-based Backdoor Attacks

In this work, we mainly consider the poisoning-based backdoor attacks against image classification, where the attacker-chosen malicious behavior is to misclassify the input (i.e., an image) with the backdoor trigger as the target label (e.g., dog).

To launch the poisoning-based backdoor attacks, the adversary typically poisons a few training samples by injecting the backdoor trigger into them and relabeling these samples as the attacker-specific target label. The existing poisoning-based backdoor attacks can be divided into three main categories [7]: *dirty-label attacks*, *clean-label attacks* and *feature space attacks*.

Dirty-label Attack is a backdoor attack in which the target label is different from the ground-truth label of poisoned samples. BadNets [5] uses a square as a trigger and stamps the trigger onto the bottom right corner of benign images to create poisoned samples. BadNets specifies the target labels in two ways: all2one, which selects one class as the target label, and all2all, which changes the labels of poisoned samples to the label of the next class in the ordering of the label space. Following BadNets, Trojan [19] inverses the neuron network to generate a general *trojan trigger* and then adds the trigger to images to complete the attack. Blend [20] completes trigger addition by blending random noise or HelloKitty images with the original image and makes the trigger invisible by adjusting the blend ratio. Adap-Blend [21] is an adaptive attack that refrains from altering the labels of all poisoned samples while employing a weak trigger during training and a strong one during inference to achieve significant stealthiness.

Clean-label Attack is a backdoor attack in which the target label is consistent with the ground-truth label of poisoned samples. SIG [15] injects triggers by adding a slowly increasing

ramp to the benign samples from the target label. CL [16] manipulates a few benign samples from the target label by using adversarial perturbations [22] or generative models [23] to generate poisoned samples.

Feature space Attack is a backdoor attack that the trigger is input dependent, i.e., different poisoned samples will come with different trigger [24]. Instagram Filter [25] changes the style of the image to insert the trigger. Pixel-level mutations induced by these filters vary from one image to another. IAB [26] trains an input-aware trigger generator to generate triggers that vary from input to input. WaNet [27] discovered that machines exhibit a higher sensitivity to warping in images as compared to humans. Accordingly, they employed a warping-based trigger to subtly distort the images. SSBA [28], inspired by the DNN-based image steganography [29], proposes to use a pre-trained encoder-decoder network to generate sample-specific triggers.

In addition to injecting backdoors by poisoning the dataset, previous work [11], [25], [30] has found that models trained on benign datasets can also contain backdoors. Specifically, the model learns to associate unintended but learnable features with the target class. Through reverse engineering, attackers can extract these features and inject them into samples from other classes, causing the model to misclassify those samples as the target class. In this paper, we refer to this as a natural trigger.

C. Existing Defenses and Analysis

Considerable effort has been made to defend against poisoning-based backdoor attacks. We direct readers to Li et al. [6] for an extensive overview of existing defenses and focus our discussion on *Poison suppression-based defenses*, which are most relevant to our proposed TRAP.

• **ABL** [7] discovers that poisoned sample losses decrease more rapidly than those of benign samples. It leverages loss values during training to distinguish poisoned samples from benign ones and eliminates backdoor by maximizing the loss function between poisoned samples and the target class (referred to as Unlearning).

• **DBD** [10] argues that self-supervised learning [31] can prevent a model from learning the correlation between triggers and target classes. Therefore, DBD proposes learning a purified feature extractor via self-supervised learning, then training a classifier using standard supervised learning with symmetric cross entropy (SCE) [32]. DBD detects benign samples during classifier training by comparing sample losses, with benign samples having smaller SCE losses. Based on isolated benign samples, DBD fine-tunes the entire model in a semi-supervised manner, treating isolated benign samples as labeled and the remaining samples as unlabeled.

• **ASD** [14] also employs SCE to split the poisoned dataset. Additionally, it utilizes the loss value to distinguish benign hard samples from poisoned ones. The poisoned dataset was dynamically divided into labeled and unlabeled data pools. Like DBD, ASD also leverages semi-supervised learning (SSL) on two data pools to defend against backdoors.

• **EBD** [12] observes a substantial discrepancy in feature representations between poisoned images before and after transformations (e.g., rotation). Utilizing this discrepancy, EBD partitions the dataset into poisoned, benign, and uncertain subsets. EBD proposes two training schemes, *D-BR* and *D-ST*, to learn a clean model. The processes of D-BR and D-ST are similar to those of ABL and DBD, respectively, but with different improvements. D-BR applies iterative unlearning and relearning techniques to poisoned and benign subsets. D-ST trains classifier with mixed cross-entropy, which includes learning on benign samples and unlearning on poisoned sample.

• **NONE** [11] reveals that backdoor-related neurons form a hyperplane as the classification surface across input domains of all affected labels. By utilizing this hyperplane, NONE filters out poisoned data, preventing the training process from generating such a hyperplane and subsequently learning a DNN model without the backdoor.

• **CBD** [13] leverages a backdoored model to encapsulate the causal effects caused by backdoor attacks. In the process of training the clean model, CBD encourages the independence of the clean model's representation from that of the backdoored model, thereby making the model focus on the correct causal effects and preventing backdoor implantation. Unlike previous works, CBD does not involve the detection of poisoned samples.

Limitations of existing defenses. Despite promising results achieved by existing Poison suppression-based defenses, they have several key limitations.

First, the existing defenses may fail to detect poisoned samples under some attacks. As discussed earlier, to mitigate the negative effects of poisoned samples, the existing defenses typically require first distinguishing between benign and poisoned samples. Although they have demonstrated effectiveness in isolating poisoned samples, we observe that there is a possibility that they can be bypassed by some attacks. More specifically, ABL uses the loss reduction rate as a metric to identify poisoned samples. However, in the case of the BadNets-all2all attack, the loss reduction rates of poisoned samples can slow down due to the multi-target attack, making ABL fails to defend against this type of attack. Like ABL, DBD and ASD employ the SCE loss to filter poisoned samples. However, since the SCE loss is originally designed for learning with noisy labels, these two defenses may be ineffective against clean-label attacks [15], [16]. EBD relies on feature consistency under transformations as a metric for detecting poisoned examples. Unfortunately, CL [16] has shown that the features of poisoned examples can be invariant to image transformations by adding appropriately flipped triggers to the samples. NONE identifies poisoned samples by establishing a reference distribution for a particular neuron and flagging inputs whose activation values deviate from the distribution. However, NONE makes an assumption that there is only one backdoor trigger, making it unable to defend against attacks like SSBA [28] which employs different triggers for different samples. In contrast, TRAP detects poisoned samples based on the intrinsic features of poisoned samples (see Section III-B for detailed discussion). Therefore, TRAP is capable of detecting various types of poisoned samples compared to the existing defenses, as we will show in Section VI.

Second, existing defenses may lead to sub-optimal performance of learned DNN models. The existing defenses employ carefully designed training schemes to alleviate the negative effects of backdoors. While these training schemes significantly remove the backdoor, they also pose a threat to the performance of learned DNNs. In particular, both ABL and EBD leverage machine unlearning techniques [17], [18] on the detected poisoned examples to remove backdoor triggers in models. However, prior works [18] have demonstrated that machine unlearning techniques can lead to a decrease in model performance due to *exploding loss* and *catastrophic forgetting*. DBD and ASD employ self/semi-supervised learning to counter backdoor injection. However, this approach comes at the cost of discarding all or some sample labels, which are crucial pieces of information in the dataset, and thus may potentially degrade the performance of learned DNNs. NONE removes backdoor triggers from DNN models by resetting backdoor-related neurons. This resetting process may impair the convergence of models and weaken their learning ability, thus resulting in sub-optimal performance [33]. CBD prevents backdoor injection by promoting clean and backdoored model representation independence. However, the backdoored models may also contain the correct causal effects. Hence, the independence between the clean and backdoored models could potentially limit the capability of the clean model, resulting in sub-optimal performance of the learned model [13]. In contrast to the existing defenses, TRAP does not alter the training process of DNN models but rather mitigates the side-effects of poisoned samples by relabeling them (see Section IV-B). As a result, TRAP can effectively remove backdoor triggers while simultaneously maintaining the performance of the learned models. More details can be found in Section VI.

III. OVERVIEW

A. Threat Model

In this work, we adopt the threat model of poisoning-based backdoor attacks that has been widely considered in prior work [7], [10]–[14].

Attacker's Goal. An attacker aims to inject poisoned samples into a training dataset such that the model learned on the dataset achieves two goals: *effectiveness goal* and *utility goal*. The effectiveness goal requires the model to predict any inputs containing an attacker-chosen trigger as an attacker-specific target label. The utility goal demands that the model maintain high accuracy on inputs without an attacker-chosen trigger.

Attacker's Capabilities. We assume that the attackers are *only* allowed to control a small portion (typically between $1\% \sim 10\%$) of the training datasets while having neither the knowledge nor the ability to control the training process (e.g., training loss, model structure). Formally, let $D_c = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ denotes clean training dataset, $\mathcal{G} : \mathcal{X} \mapsto \mathcal{X}$ denotes the attacker-specified trigger planting function, and $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{Y}$ denotes the attacker-specified poison label flipping function. The attackers can design a trigger planting function $\mathcal{G}$ and a label flipping function $\mathcal{L}$ to maliciously manipulate the controlled training samples. More specifically, the attacks will maliciously manipulate a subset of samples $D_m \in D_c$ to $D'_m = \{(\mathcal{G}(\boldsymbol{x}_i), \mathcal{L}(\boldsymbol{x}_i, y_i))|(\boldsymbol{x}_i, y_i) \in D_m\}$ and release poisoned training set $D_p = D'_m \cup (D_c/D_m)$. When users train a model $\mathcal{F}$ on the poisoned training dataset D_p the backdoor will be injected into the model. That is, the model will predict any inputs containing an attacker-specific trigger as an attacker-specific target label while keeping high accuracy on benign inputs.

Defender's Goal. The goal of defenders is to develop a security model from a given training dataset. This security model can prevent classifying inputs that contain a backdoor trigger as the attacker-chosen class. Additionally, it is capable of maintaining high accuracy on benign samples that do not have a backdoor trigger.

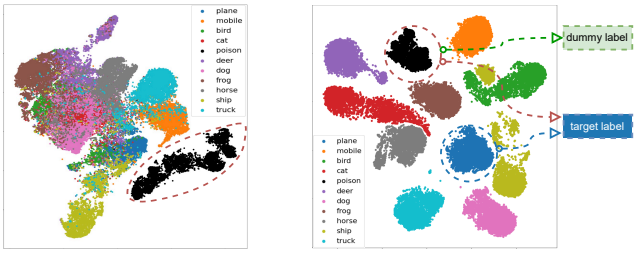
Defender's Capabilities. We assume that the defender has complete control over the training procedure, which includes model structures and training losses. However, the defender is unaware of the distribution or the proportion of poisoned examples within the given training dataset.

B. Key Observations

In this section, we present the design intuition behind TRAP. We first revisit the inherent mechanism of poisoning-based backdoor attacks, and then conduct an in-depth analysis of the behaviors of such attacks from the hidden space in the model training process.

The goal of backdoor attacks is to inject backdoor triggers into a model such that the model misclassifies inputs containing the triggers to a target label. In other words, the adversary aims to embed the correlation between backdoor trigger(s) and target label(s) into the model. To achieve so, the poisoning-based backdoor attacks maliciously add the backdoor trigger into several training images and relabel these images to the target label. This results in the poisoned samples having two characteristics: **(a)** The backdoor trigger is a strong feature of the target label. To achieve a high attack success rate, it is essential for the model to learn a stable mapping between the backdoor trigger and the target label. Hence, the backdoor trigger necessitates being a prominent attribute inherent to the target label, facilitating the model in easily learning such a stable mapping. **(b)** The poisoned samples have unique features compared to the benign ones. Considering the strong correlation between the backdoor trigger and the target label, the injected backdoor trigger is highly likely to generate remarkable discriminative features within the poisoned samples. Additionally, to successfully classify the backdoored examples as the target label, the backdoor features should suppress the other features in those examples. Therefore, the poisoned samples would have different features from the benign ones.

Following these two characteristics of poisoned samples, we next further explore the behavior of poisoned samples in the model training process by inspecting the hidden features of these samples. In particular, we perform a BadNets attack on CIFAR-10 with a poisoning rate of 10%, and the target label is set to 0. After that, we train a ResNet-18 model on the poisoned dataset and visualize the features of training examples extracted from the penultimate layer of the model in Figure 1a and Figure 1b. From the Figures, we can make



(a) Visualization of features in Epoch 10.

(b) Visualization of features after the completion of training.

Fig. 1: Visualization of training examples in the hidden space. TSNE is used for dimensionality reduction and visualization. Poisoned samples are marked in black, while others are benign samples.

the following observation. Note that similar observations can also be observed in other attacks. More results can be found in Appendix A.

Observation I: Poisoned samples form distinct clusters in the early stages of training. As depicted in Figure 1a, although the model fails to differentiate benign examples (which are mixed together in the hidden space), it is already able to distinguish between poisoned and benign examples (with poisoned samples forming a distinct cluster far from benign samples), in a very early stage of training. This occurs because the backdoor trigger is a strong feature of the target label, causing the model to learn the trigger feature rapidly. Since all poisoned samples share this similar trigger feature, they cluster together in the feature space, separate from the benign samples. Even in more concealed attacks (such as clean label attacks and feature space attacks), we have similarly observed such clear separation. In light of the observation, we define a *Suspicious Cluster*, which contains poisoned samples, as: *In the early stages of training, the clusters other than the largest cluster are considered suspicious in feature space.* We exclude the largest cluster as it primarily consists of benign samples. This arises from the difficulty the model experiences in learning highly discriminative features for benign samples during early training stages, leading to the aggregation of similar features of benign samples. Based on this, we design a detection mechanism that identifies poisoned samples by searching for suspicious clusters in the hidden space during the early stage of model training.

Note that our observation significantly differs from previous latent separation-based defenses (such as SS [34] and AC [35]), which only detect poisoned samples from the feature space after training. These methods also overlook the early training phase where benign samples are not easily classified by the model, leading to the formation of the largest clusters in feature space. Moreover, our observation is also distinctly different from ABL [7]. While both are based on the assumption that the model learns poisoned samples faster than benign ones, our observation from a representation perspective, whereas ABL is based on the rate of loss reduction—poisoned samples lose their predictive loss faster than benign samples. Our experiment demonstrates that our approach is more effective and robust than ABL.

Observation II: The classification path of poisoned samples is different from that of benign samples. As demonstrated in Figure 1b, since the remarkable distinct features introduced by the backdoor triggers, the poisoned samples are separated from the benign ones within the hidden space of the model that is fully trained. We can view the model as consisting of a feature extractor and a classifier. Considering such a hidden space, the feature extractor effectively extracts the trigger-related features, and a classification pathway exists within the classifier that maps these backdoor features to the target class. The pathway from the backdoor features to the target label may exhibit substantial deviation in comparison to the paths between the benign features and the labels of benign samples. This aligns with the attack objective of backdoors not affecting the model's classification of benign samples. Inspired by this, we can hide the backdoor behaviors while maintaining the performance of benign samples by simply changing the destination of the backdoor path to a new dummy label that does not exist in the original dataset in the classifier part of the model (i.e., turning the red path into the green path in Figure 1b). In this way, when a new poisoned sample is input to the model, the feature extractor first extracts the trigger features. Subsequently, because we have modified the endpoint of the backdoor path, these trigger features are ultimately recognized and classified by the classifier into our designated dummy label, thereby mitigating the backdoor.

C. Our Technique

We propose TRAP, a novel, effective, and practical training method that is capable of defending against a variety of existing poisoning-based backdoor attacks. The **key idea** of TRAP is to detect poisoned samples within the training dataset and then hide the mapping between the backdoor triggers and the target labels. As such, TRAP can eliminate the correlation between the backdoor triggers and the target labels, thus preventing the learned model from predicting the backdoored inputs as an attacker-chosen label. We illustrate how TRAP realizes this idea as follows.

Poisoned Sample Isolation. To differentiate between poisoned and benign samples within a training dataset, we propose to detect poisoned samples by examining their features during the initial stages of training. The insight is that *the poisoned samples would form a distinct suspicious cluster that is distinguishable from the benign samples during the early stages of training* (c.f., Observation I). Based on this, TRAP extracts the features of all training examples at the end of each initial training epoch. Then, we cluster the extracted features to identify suspicious clusters. Finally, TRAP inspects the samples present in the suspicious clusters to identify the poisoned samples, resulting in the original training dataset being split into clean and poisoned sets. We will present the details about how we identify poisoned examples in Section IV-A.

Backdoor Hiding. To remedy the negative impacts of the poisoned samples, we relabel the detected poisoned samples as a new dummy class and retrain the model's classifier to inject this updated trigger-label mapping. In other words,

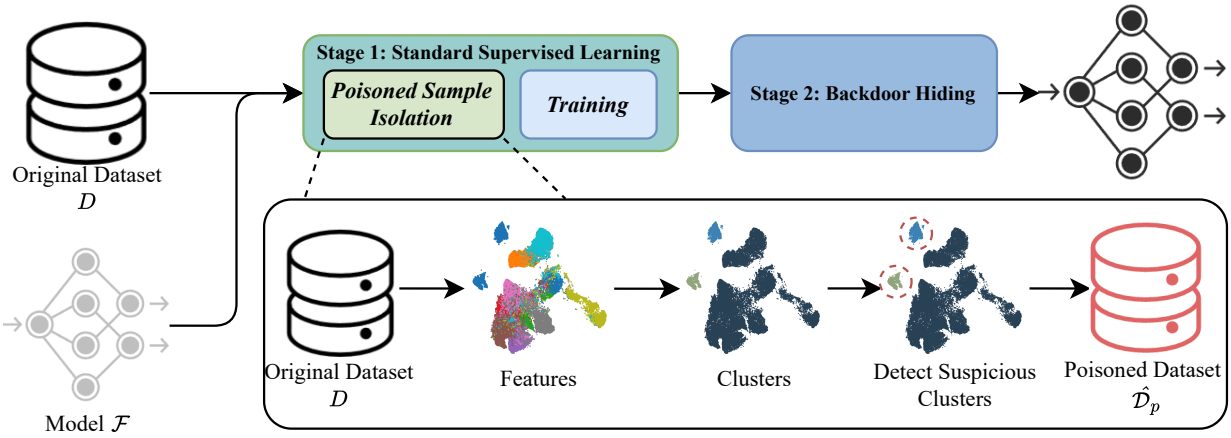


Fig. 2: Overview of TRAP. Early in the model training process, we first perform poisoned sample isolation. This step involves using the model for feature extraction, clustering, identifying the suspicious cluster, and obtaining the poisoned samples. After we obtain the poisoned samples, we continue to train the model. Once training is complete, we proceed to backdoor hiding, eliminating the backdoor by freezing the feature extractor, relabeling the poisoned samples, and retraining the classifier.

we introduce a novel dummy class to further poison the poisoned samples. The intuition is that *there is a unique distinct path for associating the backdoor features with the target labels in the classifier part of a model* (c.f., Observation II). As such, by re-training the classifier part of the model using the newly labeled poisoned samples, we rewrite the mapping of the backdoor features to the new dummy class, thereby alleviating the original backdoor path. It is important to note that the feature extractor part of the model is able to discriminate well between poisoned and benign features (as shown in Observation II). Therefore, it is sufficient to hide the backdoor behaviors by only retraining the model's classifier part. Additionally, associating the backdoor features with a new dummy class does not impact the classification of benign samples, because the backdoor features differ significantly from the benign features. We will discuss the implementations of the backdoor hiding mechanism in Section IV-B.

IV. DESIGN

Figure 2 illustrates the overview of TRAP. It consists of two stages: (1) standard supervised learning and (2) backdoor hiding. In the first stage, TRAP trains the model $\mathcal{F}$ on the original training dataset D using the standard supervised learning. During the standard supervised learning, we adopt poisoned sample isolation to inspect a few initial training stages to detect the poisoned samples within the training dataset D (Explained in Section IV-A). In the second stage, TRAP conducts backdoor hiding to mitigate the backdoors in the model learned in the first stage. In particular, we relabel the poisoned dataset $\hat{D}_p$ produced by the poisoned sample isolation and then retrain the classifier part of the model on the relabeled poisoned dataset to hide the backdoors (Explained in Section IV-B). We will discuss more details of TRAP in the following.

A. Poisoned Sample Isolation

The goal of poisoned sample isolation is to detect the poisoned samples within a given training dataset D. Our intuition is that *the poisoned samples would form a distinct suspicious cluster that is distinguishable from benign samples during the early stages of training*. Therefore, we propose to examine the features of training examples at the first E_d training epochs to discriminate the poisoned samples from the benign ones. The workflow of our poisoned sample isolation method, as shown in Figure 2, begins by extracting hidden features for each training example $\boldsymbol{x}$ in the training dataset D. These features are then clustered into different groups. Next, we examine each feature cluster and search for the suspicious cluster that may contain the poisoned examples. Finally, the poisoned samples are determined based on these suspicious clusters. In the following, we elaborate on each step in detail.

1) Feature Extraction: We start by extracting features for training examples in the given training dataset D. In this work, we follow the prior work [10], [12] to employ the penultimate layer representations as the features of training examples. That is, we run each example $\boldsymbol{x}_i \in D$ through the model $\mathcal{F}$ and extract the output of the penultimate layer as the feature $\boldsymbol{v}_i$ of $\boldsymbol{x}_i$. More specifically, let us view the model $\mathcal{F}$ as $\mathcal{F} = g \circ f$, where f is the feature extractor of $\mathcal{F}$ which is the function of the layers before the last layer in the model $\mathcal{F}$ and g is the classifier part of $\mathcal{F}$ which is the last fully-connected layer in the model $\mathcal{F}$. Then, the feature $\boldsymbol{v}_i$ of $\boldsymbol{x}_i \in D$ can be formally represented as $\boldsymbol{v}_i = f(\boldsymbol{x}_i)$.

2) Clustering: After acquiring hidden features $\boldsymbol{v}$ of all training examples, we adopt the DBScan algorithm (Density-Based Spatial Clustering of Applications) [36] to cluster features of all training examples into different groups. The reasons why choosing DBScan are twofold: (a) Specifying the number of clusters in the feature space is a non-trivial task, which renders clustering algorithms that require pre-definition of the number of clusters, such as the K-means algorithm [37], unsuitable. DBScan is a density-based clustering algorithm, which groups together data points in high-density regions that are spatially close together, and thus, it does not need to pre-define the number of clusters. (b) DBScan has been demonstrated to be effective and efficient and has been

widely used in many domains [38], [39]. DBScan requires two hyper-parameters for clustering: *eps* and *min-samples*. The *eps* parameter determines the maximum distance around data points that determines the neighboring boundary. The *min-samples* parameter is the number of neighboring data points required to form a cluster. We discuss how we determine the values of these two parameters in Appendix VI-F.

However, applying the DBScan algorithm directly on the features $\boldsymbol{v}$ from the penultimate layer of the model $\mathcal{F}$ could be less efficient because the features $\boldsymbol{v}$ are high-dimensional. Therefore, we explore the dimensionality reduction algorithm TSNE [40] to reduce the dimensionality of the features $\boldsymbol{v}$, We provide the visualization results of DBScan clustering after dimensionality reduction in Appendix B. We also experimented with other dimensionality reduction algorithms such as PCA [41] and SVD [42], but found TSNE to be most robust and effective in our setting. Please see Appendix F for a more detailed analysis.

3) Suspicious Cluster Detection: Based on the feature clusters, we further identify suspicious clusters that may contain poisoned samples. Our criterion is that *a cluster is considered suspicious if it is not the largest cluster.* Following this, we examine each cluster produced by the DBScan algorithm one by one and determine whether or not the cluster is suspicious by quantifying both the number of classes it encompasses and its density. If a cluster contains only one class or all classes and meanwhile is not the largest cluster, then we classify the cluster as a suspicious cluster.

Algorithm 1: Poisoned Sample Isolation

Input: Dataset D, Model $\mathcal{F}$, Detection Epochs E_d
Output: Poisoned Dataset $\hat{D}_p$, Clean Dataset $\hat{D}_c$, Model $\mathcal{F}$

1 allSuspiciousClusters $\leftarrow \varnothing$;
2 **while** *Epoch* $e \leq E_d$ **do**
3 Train $\mathcal{F}$ using Standard Supervised Learning ;
4 features $\leftarrow$ `ExtractFeature` ($\mathcal{F}$, D) ;
5 features $\leftarrow$ `TSNEReduction` (features) ;
6 clusters $\leftarrow$ `DBScanClustering` (features, D) ;
7 sc $\leftarrow$ `DetectSuspiciousClusters` (clusters) ;
8 allSuspiciousClusters.`Extend` (sc) ;
9 **end**
10 $\hat{D}_p \leftarrow$ `GetSampleInClusters` (allSuspiciousClusters) ;
11 $\hat{D}_c \leftarrow D \setminus \hat{D}_p$;
12 **return** $\hat{D}_p$, $\hat{D}_c$, $\mathcal{F}$

4) Poisoned Samples Detection: With the detected suspicious clusters, we finally discerned the poisoned samples within the training dataset D from these clusters. Algorithm 1 outlines the process of identifying poisoned samples. In the algorithm, D denotes the original training dataset which may contain poisoned examples; $\mathcal{F}$ denotes the model that needs to be trained; E_d denotes how many training epochs we consider to identify the poisoned samples. Next, we provide the details of each step in Algorithm 1.

Initially, we train the model $\mathcal{F}$ a few E_d epochs using the standard supervised learning and search the suspicious clusters at the end of every epoch (The loop in lines 2-9). In particular, in each training epoch e, we first train the model $\mathcal{F}$ with the standard supervised learning (Line 3). Next, we extract

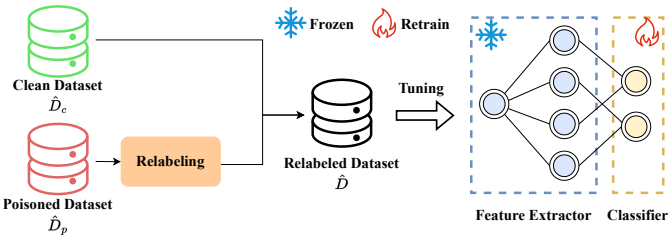


Fig. 3: Backdoor Hiding

the features of all samples in the training dataset and then reduce the dimensionality of the features (Lines 4-5). Then, we cluster the features into different groups (Line 6). Finally, Line 7 identifies the suspicious clusters using the criterion stated in Section IV-A3 and Line 8 saves these clusters to a global list "allSuspiciousClusters". Based on our criterion, all clusters except for the largest one (the cluster containing the most samples) will be considered suspicious. Finally, we treat all samples within the suspicious clusters as poisoned samples to form the poisoned dataset $\hat{D}_p$, and divide the entire dataset D into clean and poisoned datasets (Lines 10-11).

Remark. In Algorithm 1, we detect the poisoned samples within the training dataset D by considering total E_d epochs instead of a single epoch. This is because the model training would be unstable during the very early phases of training, which is the focus of our poisoned sample isolation method. Thus, detecting poisoned samples using a single epoch has sub-optimal results due to the randomness in the learning process. By considering all E_d epochs, we can obtain stable training dynamics, thus achieving better results of poisoned samples isolation.

B. Backdoor Hiding

We herein introduce our backdoor hiding method, which aims to remedy the negative impacts of the poisoned samples after the standard supervised learning. The key insight of our backdoor hiding method is to *change the mapping between the backdoor features and the attacker-chosen target labels by re-training the classifier part g of the model $\mathcal{F}$*. Figure 3 illustrates the basic idea of the backdoor hiding approach.

Given the clean dataset $\hat{D}_c$ and the poisoned dataset $\hat{D}_p$ produced by poisoned sample isolation, we first relabel the samples in the poisoned dataset $\hat{D}_p$ with a new dummy class to change the correlation between the poisoned samples and the attacker-chose target labels. That is, we poison the poisoned dataset $\hat{D}_p$ with a new dummy class. Assuming the original training dataset D contains C classes, then we index the new dummy class by $C+1$. Accordingly, we construct the relabeled poisoned dataset $\hat{D}_p^r$ as $\hat{D}_p^r = \{(\boldsymbol{x}, C+1) | \boldsymbol{x} \in \hat{D}_p\}$. Transitively, a newly training dataset $\hat{D}$ is obtained by combining the relabeled poisoned dataset $\hat{D}_p^r$ and the clean dataset $\hat{D}_c$, i.e., $\hat{D} = \hat{D}_c \cup \hat{D}_p^r$.

We then inject the new mapping between the poisoned samples and the dummy class $C+1$ into the model $\mathcal{F}$. To achieve this, we retrain the classifier part g of the model $\mathcal{F}$ for E_r epochs from scratch on the new training dataset $\hat{D}$ using the standard cross-entropy loss. Note that we froze the

feature extractor f of the model $\mathcal{F}$ throughout the retraining process because after the feature extractor f is capable of distinguishing between the backdoor and benign features and we only need to rewrite the pathway between the backdoor features and the attacker-chose target labels (discussed in Observation II). Furthermore, by fixing the feature extractor f, the classifier g could better utilize the discriminative features already learned by f. Therefore, our backdoor hiding method effectively associates the backdoor features with the new dummy class $C+1$, while maintaining the high accuracy of benign samples. Note that it is necessary to preserve the backdoor hiding mechanism rather than simply retrain the model. We included a detailed discussion in Appendix M.

V. Experimental Methodology

Dataset and Model. We extensively evaluate our proposed TRAP on four classical benchmark datasets, GTSRB [43], CIFAR-10 [44], CIFAR-100 [44], and an ImageNet [45] subset. These datasets are selected because they are the most widely used datasets in previous work [7], [10]–[14]. More details about the subject datasets can be found in Appendix C. As for model architectures, we adopt the ResNet-18 [46] for all three benchmarks, which is consistent with prior work [7], [10]–[12] to ensure a fair comparison. Furthermore, following the prior works [7], [8], [13], we also adopt the WideResNet-16 to evaluate the generalizability of TRAP.

Attack Configuration. We consider 12 poisoning-based backdoor attacks based on the taxonomy outlined in Section II-B, including 6 dirty-label attacks: BadNets (all2one and all2all) [5], Trojan [19], Blend-Strip [20], Blend-Kitty [20], and Adap-Blend [21], 2 clean-label attacks: CL [16] and SIG [15], and 4 feature-space attacks: Instagram Filter [25], IAB [26], WaNet [27], and SSBA [28]. Besides, we also consider natural trigger [11], [25] on CIFAR-10. We utilize the open-source codes accompanying their respective papers to configure these attacks. To keep their original parameter settings, we only run SSBA on the ImageNet subset. Furthermore, we also omit some attacks (e.g., the clean-label attacks and Trojan) on CIFAR-100 and the ImageNet subset because these attacks fail to scale to the two datasets. More detailed settings of the 12 attacks can be found in Appendix D and our code repository.

Compared Approaches. We compare TRAP with 7 SOTA poison suppression-based defenses discussed in Section II-C: ABL [7], DBD [10], D-BR [12], D-ST [12], NONE [11], CBD [13] and ASD [14], and 2 baseline backdoor defenses: DP-SGD [9] and NAD [8]. For all the 9 defenses, we adopt their official codes and default configurations specified in their original papers. Furthermore, we also provide results of DNNs trained without any defenses, i.e., No Defense.

Parameters Settings. Unless otherwise mentioned, we use the following parameter settings for our TRAP: We follow prior work [7], [10]–[12] to set the poisoning rate γ to 10%. The epoch E_t of the standard supervised learning stage is set to $E_t = 100$. For the poisoned sample isolation module, we inspect the initial $E_d = 10$ epochs. We adopt *eps* = 3 and *min-samples* = 50 for the clustering algorithm DBScan. For the backdoor hiding module, we re-training the classifier part of a model for $E_r = 10$ epochs. Due to space constraints, we have placed the parameter experiments in Appendix VI-F.

Evaluation Metrics. Following previous work [7], [8], [10]–[12], we employ Accuracy (ACC) and Attack Success Rate (ASR) as metrics for evaluating defense performance. ACC is the accuracy of the model on the clean test set, while ASR is the accuracy of predicting poisoned samples as the target label in poisoned test set. Specifically, ACC and ASR are respectively defined as $ACC \triangleq \Pr_{(\boldsymbol{x},y)\in D_c}\{\mathcal{F}(\boldsymbol{x}) = y\}$ and $ASR \triangleq \Pr_{(\bar{\boldsymbol{x}},y_t)\in D_p}\{\mathcal{F}(\bar{\boldsymbol{x}}) = y_t\}$. Note that, a larger ACC means a higher classification performance of the DNN model and a smaller ASR means a higher robustness of the DNN model. Therefore, a larger $ACC - ASR$ indicates a better method. Furthermore, we also include precision and recall as evaluation metrics to evaluate the performance in detecting poisoned samples. Precision indicates the proportion of isolated samples that are poisoned, while recall indicates the proportion of overall poisoned samples that are isolated.

Experiment Environment. All experiments are conducted on a workstation with Intel Core i9-10900X, 64GB memory, and one RTX 3080Ti GPU, running Ubuntu 22.04.

VI. Results

A. Overall Defense Performance

This section presents the experimental results to demonstrate the effectiveness of our proposed TRAP. Table I presents the comparison results among TRAP and the nine compared approaches in terms of the ACC and the ASR for the ResNet18 on the three subject datasets (i.e., CIFAR-10, CIFAR-100, and the ImageNet subset), respectively. Note that the results for the GTSRB and WideResNet-16 follow the same trend, and hence, we put those into Appendixes G and H. In Table I, each row represents the defense performance of different methods under the same dataset and attack method, and each column represents the defense performance of the same method on different attack methods and datasets. Next, we detail the comparison results on each dataset.

Results on CIFAR-10. From the comparison results on CIFAR-10, we have the following observations. First, TRAP can effectively protect models from being attacked. Specifically, compared with the model without defense, TRAP achieves a substantial reduction of the average ASR of models, from 97.2% to 0.01%, and meanwhile, only decreases 0.21% ACC on average. This demonstrates that TRAP can effectively mitigate backdoor threats while preserving the high clean accuracy of learned models. Second, TRAP significantly outperforms all nine defenses in terms of both the ASR and the ACC. Specifically, when contrasted with the average ASRs of the nine defenses — which range from 15.8% in NAD to 63% in DP-SGD — TRAP attains a markedly lower average ASR of 0.01%. Regarding ACC, TRAP exceeds the performance of all other defenses. Notably, compared to the defenses which achieve ACC exceeds 90%, such as NONE, DBD, D-ST, CBD, and ASD, TRAP still surpasses the ACC of them by average margins of 1.39%, 3.28% 4.19%, 3.51%, and 1.51% respectively. These results demonstrate the apparent superiority of

TABLE I: Comparisons of TRAP with two baseline and seven state-of-the-art defense method (%). The results of D-ST on ImageNet are omitted because SupCon [47] used in D-ST does not provide open-source code on ImageNet. The bold numbers denote the best defense performance across different methods.

Dataset	Defense→ / Attack↓	No defense		DP-SGD		NAD		ABL		NONE		DBD		D-BR		D-ST		CBD		ASD		TRAP	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
CIFAR10	None	95.1	0	60.66	0	83.02	0	81.18	0	92.49	0	90.17	0	85.13	0	90.73	0	91.53	0	52.43	0	**95.06**	0
	BadNets-all2one	94.8	100.0	51.4	100.0	83.6	7.2	85.2	0.1	93.4	1.7	92.2	0.9	90.4	0.5	90.2	1.4	90.5	1.9	91.6	2.5	**94.8**	**0.0**
	BadNets-all2all	94.6	90.9	52.9	4.5	80.7	2.3	78.9	80.5	93.3	20.9	91.8	3.7	83.2	78.4	87.0	11.2	90.5	87.6	92.8	11.0	**93.4**	**0.0**
	Trojan	94.7	100.0	53.4	95.5	83.9	3.7	88.1	0.2	94.2	1.5	92.3	2.8	91.6	0.9	94.1	0.0	91.4	1.9	93.8	1.9	**94.3**	**0.0**
	Blend-Strip	94.6	100.0	50.3	100.0	83.9	56.5	84.5	3.2	90.6	40.2	92.2	2.5	91.5	70.5	93.9	0.0	91.7	5.1	92.1	100.0	**94.7**	**0.01**
	Blend-Kitty	94.6	100.0	51.8	100.0	84.1	0.5	84.5	0.0	93.4	2.3	92.7	1.3	91.0	63.0	93.3	0.1	91.1	48.0	93.7	0.5	**94.6**	**0.0**
	SIG	94.9	99.5	53.8	77.5	86.1	47.5	69.7	0.3	94.2	99.0	91.5	22.1	80.4	0.0	89.9	97.6	91.9	5.5	93.8	99.0	**94.9**	**0.03**
	CL	94.8	89.7	54.1	14.9	84.8	21.3	76.2	0.1	93.7	2.1	90.8	16.2	73.1	46.2	89.7	16.4	90.3	63.0	93.6	0.8	**94.6**	**0.0**
	WaNet	94.4	98.9	54.7	11.2	85.2	6.0	77.4	5.1	92.5	20.1	91.4	2.6	88.3	80.2	88.9	20.2	91.0	12.9	91.8	1.8	**93.7**	**0.1**
	Instagram Filter	94.6	99.7	47.9	77.7	85.6	7.1	73.2	4.4	92.8	81.3	89.0	27.6	91.2	98.8	91.1	99.1	92.0	6.2	92.9	1.3	**94.7**	**0.03**
	IAB	94.6	100.0	53.3	95.9	79.2	1.4	91.8	0.1	92.7	14.6	91.2	87.3	89.8	90.3	88.3	79.8	91.0	6.4	93.5	1.8	**94.5**	**0.0**
	Adap-Blend	94.0	90.9	51.8	15.9	85.2	20.0	61.9	90.9	93.1	67.0	90.2	75.2	87.4	64.8	90.7	66.6	91.7	77.0	93.0	27.0	**94.4**	**0.0**
	Average	94.6	97.2	52.3	63.0	83.9	15.8	79.2	16.8	93.1	31.9	91.4	22.0	87.1	54.0	90.6	35.7	91.2	28.7	93.0	22.5	**94.4**	**0.01**
CIFAR100	None	75.51	0	43.59	0	60.85	0	68.19	0	69.72	0	69.53	0	70.96	0	73.4	0	71.79	0	67.59	0	**74.78**	0
	BadNets-all2one	73.1	100.0	39.1	35.7	65.0	0.2	67.9	0.0	69.1	8.6	69.5	0.0	60.9	60.9	67.6	71.9	67.6	5.5	67.7	0.5	**73.0**	**0.0**
	Trojan	74.3	100.0	37.6	55.7	63.0	0.4	67.1	0.0	73.4	0.3	70.8	0.1	73.7	0.1	71.8	0.0	70.4	0.7	67.1	2.9	**74.3**	**0.0**
	Blend-Strip	74.4	100.0	44.6	98.7	62.3	0.4	64.9	0.0	70.5	64.3	71.4	5.5	72.6	0.2	70.5	0.0	71.8	8.4	67.2	99.8	**74.2**	**0.0**
	Blend-Kitty	74.8	100.0	43.1	96.5	66.2	6.8	67.8	0.0	73.2	39.7	70.5	22.7	70.3	0.0	70.9	0.0	70.8	22.6	68.1	99.4	**74.5**	**0.0**
	WaNet	73.8	89.8	40.0	35.8	67.4	0.8	64.3	0.0	68.2	77.3	70.2	7.8	71.5	77.4	71.5	41.9	69.2	80.7	64.9	7.8	**72.5**	**0.0**
	Instagram Filter	74.0	100.0	40.7	89.0	62.6	0.7	65.6	0.0	70.9	76.5	71.5	83.5	72.3	99.6	72.0	97.7	70.2	52.8	66.2	64.6	**73.9**	**0.01**
	Average	74.1	98.3	40.9	68.6	64.4	1.5	66.3	0.0	70.9	44.4	70.6	19.9	70.2	39.7	70.7	35.3	70.0	28.4	66.8	45.8	**73.7**	**0.002**
ImageNet	None	88.66	0	45.38	0	87.01	0	83.86	0	85.28	0	76.41	0	84.22	0	\		86.88	0	75.89	0	**88.2**	0
	BadNets-all2one	87.9	96.3	43.1	29.5	88.1	51.5	79.2	0.1	85.9	0.7	76.9	2.9	86.6	0.1			82.3	91.2	75.5	0.2	**88.0**	**0.02**
	Blend-Strip	87.6	96.1	49.3	82.6	88.3	48.7	79.9	0.2	85.5	99.6	74.9	0.2	86.3	5.2			83.4	92.1	73.6	0.4	**87.1**	**0.0**
	Blend-Kitty	88.4	99.5	46.3	99.8	88.5	98.7	76.7	0.0	85.6	99.5	75.3	7.3	86.7	13.8	\		86.1	67.0	74.3	1.2	**87.2**	**0.0**
	SSBA	88.1	99.3	42.8	85.7	88.7	64.9	79.8	0.0	85.4	98.1	77.6	14.2	84.1	98.7			86.1	74.0	75.6	1.4	**87.9**	**0.0**
	WaNet	86.4	99.2	49.1	51.1	87.7	53.5	80.6	11.7	84.7	54.4	75.9	17.4	86.1	19.7			86.2	15.2	74.1	0.6	**88.3**	**0.28**
	Average	87.7	98.1	46.1	69.7	88.3	63.5	79.2	2.4	85.4	70.4	76.1	8.4	86.0	27.5			84.8	67.9	74.6	0.8	**87.7**	**0.06**

TRAP compared with the existing defense methods. Third, TRAP is capable of defending against all studied poisoning-based backdoor attacks, while the current defenses may fail in certain types of attacks. For example, although ABL reduces ASR to below 1% on most attacks, it only reaches an 80.5% ASR on the BadNets-all2all attack and 90.9% on the Adap-Blend attack. In contrast, TRAP obtains an ASR of less than 0.1% on all attacks, indicating a high generality of TRAP. This suggests that TRAP, based on the inherent mechanisms of backdoor attacks, has the ability to effectively prevent a wide range of poisoning-based backdoor attacks.

Results on CIFAR-100. We also evaluate TRAP on CIFAR-100. From Table I, we find that TRAP still protects the model most effectively and has the lowest accuracy loss among all defense methods. In particular, TRAP decreases the average ASR of models from 98.3% to 0.002% which is significantly better than other defense methods (NONE, D-BR, D-ST, DBD can only reduce the average ASR to 44.4%, 39.7%, 35.3%, and 19.9% respectively). We notice that NAD and ABL can also achieve lower ASR of models. However, NAD and ABL significantly reduce the ACC of models with a reduction of 13.09% and 10.52%, respectively, while TRAP only reduces the ACC with a minor reduction of 0.53%. These results again confirm the effectiveness of TRAP.

Results on ImageNet. We additionally evaluate TRAP on ImageNet, to demonstrate that TRAP is scalable with high-resolution datasets. According to Table I, TRAP significantly and consistently outperforms all existing defense methods, in terms of both the ACC and the ASR, on ImageNet. Specifically, TRAP significantly decreases the ASR of models from 98.1% to 0.06%, while ABL, NONE, DBD, and ASD can only reduce the ASR to 2.4%, 70.4%, 8.4%, and 0.8%, respectively. In addition, TRAP obtains on average 10.73%, 2.69%, 15.24%, and 17.56% larger ACC than ABL, NONE, DBD, and ASD, respectively. These results demonstrate that TRAP is still able to achieve state-of-the-art defense performance on high-resolution datasets.

Results on No Attack. We also conducted experiments in settings without attacks. From Table I, we can see that when the attack is set to None, TRAP still achieves the best ACC on three datasets. Compared to the no defense, the ACC on CIFAR-10, CIFAR-100, and ImageNet only drops by 0.04%, 0.96%, and 0.5%. This performance surpasses the best comparative methods by 2.70%, 1.88%, and 1.51%, respectively.

Results on Natural Trigger. In addition to evaluating the inserted triggers, we further evaluate the defense effectiveness of TRAP against natural triggers on CIFAR-10. Specifically, we treat the suspicious clusters identified by TRAP as strong features of benign samples when there are no poisoned samples. For backdoor hiding, we relabel these strong feature samples to the virtual class to eliminate natural backdoors. We compare TRAP against both baselines and state-of-the-art approaches. Table II presents the comparison results on natural triggers. We observe that TRAP obtains 92.34% ACC and 28.11% ASR, achieving the highest performance compared to state-of-the-art techniques. Specifically, compared with the model without defense, TRAP significantly lowers the mean ASR from 76.81% to 28.11%, whilst only marginally lessening the ACC by an average of 2.48%. Meanwhile, TRAP increases the ACC by 1.62% and decreases the ASR by 11.60% compared

TABLE II: Comparisons on Natural Trigger.

No Defense		DPSGD		NAD		ABL		NONE		DBD		D-BR		D-ST		CBD		ASD		TRAP	
ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
94.8	76.8	69.6	34.8	88.5	65.1	75.7	35.2	85.4	40.3	77.2	37.9	84.4	39.7	90.8	31.8	60.1	20.3	50.9	68.7	**92.34**	**28.11**

TABLE III: Comparison results of different poisoned sample isolation algorithms.

Methods→	SS		AC		ABL		NONE		ST		TRAP	
Attack↓	P	R	P	R	P	R	P	R	P	R	P	R
Badnets-all2one	19.1	62.8	21.8	99.6	100	10	99.8	100	100	55.6	**100**	**100**
Badnets-all2all	11.7	34.2	34.2	81.9	1.6	0.16	23.6	85.1	0.8	0.4	**100**	**96.8**
Trojan	16.4	54.2	9.9	42.9	100	10	99.9	99.7	100	55.5	**100**	**98.2**
Blend-Strip	19.0	63.2	23	100	95.6	9.6	3.5	17.5	100	55.4	**100**	**100**
Blend-Kitty	18.9	63.2	22.3	100	96	9.6	18.6	99.9	99.9	55.6	**100**	**100**
SIG	1.0	30.2	1.4	58.9	36.2	36	0.1	2.7	3.8	19.4	**100**	**95.7**
CL	0.2	8.3	0.2	10.2	68	69.5	5.7	100	3.9	20.2	**100**	**99.5**
WaNet	4.28	11.6	25.5	96.5	31.1	2.9	56.1	8.4	5.8	2.7	**98.5**	**96.8**
Instagram Filter	18.1	60.5	21.1	**99.9**	99.4	9.9	25.5	8.1	8.8	4.8	**99.7**	96.3
IAB	13.9	46.1	23.1	**100**	100	10	99.6	8.9	7.57	3.17	**100**	99.1
Adap-Blend	18.1	65.5	60.4	83.2	0	0	5.6	1.7	3.9	6.5	**98.0**	**85.1**

to the best-performing defense approach D-ST. These results demonstrate the effectiveness of TRAP in defending against natural triggers.

B. Analysis of Poisoned Sample Isolation

We commence by conducting a comparison between our poisoned samples isolation component and other currently available methods. Following the previous work [10], [12], we also include two detection-based defenses as baselines, Spectral Signatures (SS) [34] and Activation Clustering (AC) [35]. We conduct experiments on CIFAR-10 with the eleven attacks. Table III shows the isolation performance of each studied approach. 'P' presents the precision and 'R' presents the recall.

As shown in Table III, it is evident that TRAP outperforms the two baseline methods, SS and AC, by achieving a precision of over 98% and a recall rate of over 95% except Adap-Blend. In particular, TRAP exhibits a significant improvement in precision compared to AC, ranging from 37.6% (from 60.4% to 98.0% in Adap-Blend) to 99.8% (from 0.2% to 100% in CL). Despite our recall rate being marginally less than that of AC during both the Instagram Filter and IAB attacks, with a decrease of 3.6% and 0.9% respectively, we witness a substantial improvement in precision relative to AC. Specifically, the precision rate increased from 21.1% to 99.7% and from 23.1% to 100%, respectively, signifying a significant enhancement in precision. Due to our ability to capture the characteristics of benign samples converging into large clusters during early training, we rarely screen out any benign samples. Although SS and AC demonstrate effectiveness in handling dirty label attacks and feature space attacks, their defense against clean label attacks is inadequate. In contrast, TRAP exhibits superior performance in both dirty and clean label attacks. In SIG attacks, TRAP achieves a precision rate of 100% and improves recall from 30.2% and 58.9% to 95.7% compared to SS and AC, respectively. All the above results demonstrate the effectiveness and scalability of TRAP.

TRAP stands out from other defenses as it can simultaneously achieve high precision and recall. While ABL and ST can achieve nearly 100% precision, their recall is limited to a maximum of 69.5% and 55.6%, respectively. However, TRAP can achieve over 95% recall while maintaining nearly 100% precision on almost all attacks. While NONE performs well in most dirty label attacks, it fails to provide satisfactory isolation of clean label attacks and feature space attacks. For example, although NONE detects all poisoned samples in the CL attack (with a recall of 100%), it achieves only 5.7% precision, indicating that a number of benign samples are mistakenly filtered out. This indicates a scalability issue with current methods. In contrast, TRAP can adapt to all attacks and effectively filter out poisoned examples.

C. Analysis of Backdoor Hiding

In the evaluation of Backdoor Hiding effectiveness, we compared our method to three backdoor erasure methods: Unlearning, Unlearning + Relearning, and Mixed Cross-Entropy, which were used by ABL, D-BR, and D-ST, respectively. We excluded NONE, CBD, and ASD from the comparison object because it is incompatible with our method. We compare the defense performance of different erasure methods combined with our isolation methods to evaluate their effectiveness. Figure 4, 5 show the removal performance of TRAP with other compared methods.

From Figure 4 and Figure 5, we observe that while the Unlearning method reduces ASR to 0% in all attacks, it also results in a significant decline in ACC. In contrast, Backdoor Hiding can achieve a higher ACC, with an improvement range by 11.6% (from 83.17% to 94.77% in BadNets-all2one) to 46.31% (from 48.63% to 94.94% in SIG) compared to unlearning, while achieving the same level of backdoor removal. This indicates that Backdoor Hiding can erase backdoors with almost no change in model performance compared to unlearning. Furthermore, Unlearning may lead to model collapse, as shown in Figure 4, Unlearning and Unlearning + Relearning only achieve ACC of 48.63% and 54.07% in SIG attack, respectively. However, Backdoor Hiding achieves an ACC of 94.94% in SIG attacks. Although the Mixed Cross-Entropy method achieves a good balance between maintaining ACC and reducing ASR, our method still outperforms it with a 2.01% higher average ACC and reduces average ASR from 0.93% to 0.01%. The comparative experimental results demonstrate that our method effectively maintains model accuracy and reduces ASR, excelling at both tasks.

D. Time Efficiency

In this section, we empirically investigate the time efficiency of TRAP. Specifically, we measure the time cost spent by No Defense, 9 backdoor defenses and TRAP to defend against BadNets-all2one attacks on CIFAR-10. The results in Figure 6 show that TRAP incurs slightly more time than training without defenses (36.45 mins vs. 27.56 mins) because it includes additional modules for poisoned sample isolation and backdoor hiding, but this overhead is significantly lower than that

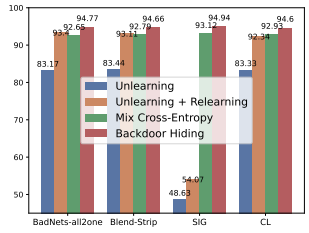


Fig. 4: ACC under four different backdoor removal modules.

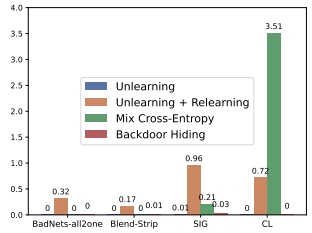


Fig. 5: ASR under four different backdoor removal modules.

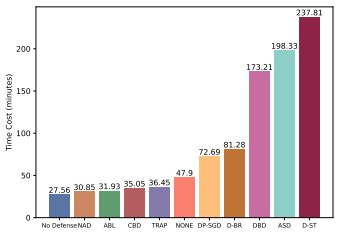


Fig. 6: Time consumed by different defense methods.

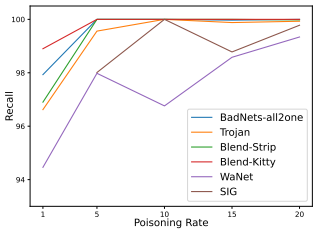


Fig. 7: Recall of poisoned sample isolation under different poisoning rates.

TABLE IV: Performance of TRAP on CIFAR-10 poisoned under different poisoning rates.

γ	0.5%				5%				20%			
Attack↓	No Defense		TRAP		No Defense		TRAP		No Defense		TRAP	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
BadNets-all2one	95.02	99.97	94.89	0	94.56	100	94.55	0	92.53	100	94.21	0
BadNets-all2all	94.74	82.26	94.85	0.01	94.42	91.01	93.87	0.03	94.27	91.34	92.80	0.04
Trojan	95.18	98.22	95.12	0.05	95.00	99.95	94.73	0.02	94.51	100	94.18	0.01
Blend-Strip	94.94	98.74	94.47	0.01	94.88	100	94.89	0	94.26	100	94.17	0
Blend-Kitty	94.65	99.95	94.65	0.01	94.83	100	94.75	0	94.21	100	94.26	0
AVG	94.90	95.82	94.79	0.016	94.73	98.19	94.55	0.01	93.95	98.26	93.92	0.01

TABLE V: Recall and precision of the poisoned sample isolation component under different hyper-parameters of DBScan.

eps	0.5	1	1.5	2	2.5	3	3.5
Recall	24.64	66.63	66.65	83.15	99.82	99.82	95.75
Preicison	63.58	66.66	66.54	83.145	99.90	100	100
min-samples	10	20	30	40	50	60	70
Recall	99.82	99.82	99.82	99.82	99.82	99.82	99.82
Precision	100	100	100	100	100	100	100

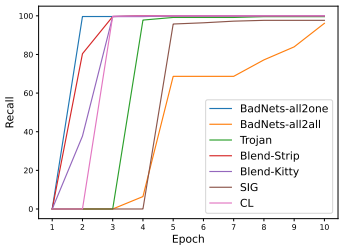


Fig. 8: Recall of each epoch under different attacks.

of NONE, DP-SGD, D-BR, DBD, ASD, and D-ST, and is comparable to that of NAD, ABL, and CBD. Furthermore, TRAP achieves significantly better defense performance than the other methods as demonstrated in Section VI-A. This demonstrates the clear efficiency of TRAP compared to the prior work. Moreover, since the training process is generally conducted offline, the training time of TARP, although slightly longer than the standard training, is practicable. In the poisoned sample isolation process, the time consumed by t-SNE is the dominant factor. To mitigate the impact of t-SNE on the scalability of TRAP, we used GPU-accelerated t-SNE [48] in our implementation of TRAP. This method leverages a GPU to significantly accelerate the t-SNE dimensionality reduction process. For example, on the full ImageNet dataset (1.2 million images), this method requires only 8 minutes to complete a dimensionality reduction pass on the entire dataset. Therefore, we believe that TRAP can scale to much larger datasets.

E. Analysis under different poisoning rate

The effectiveness of TRAP may be impacted by different poisoning rates γ. In line with prior research [12], we configure the γ to 0.5%, 5% and 20% correspondingly to counteract various attacks. Table IV displays the performance of TRAP under different poisoning rates. As shown in Table IV, TRAP consistently maintains high ACC while reducing ASR to nearly zero across various poisoning rates. In particular, TRAP is capable of reducing the average ASR from 95.82% to 0.016% with a slight decrease in ACC (0.11% in average) when the γ is 0.5%. Similarly, when the γ is 5%, our approach can decrease ASR from 98.19% to 0.01% with hardly decrease in ACC (0.19% on average). These findings suggest the effectiveness of TRAP under different poisoning rates.

In addition, we also conducted experiments on poisoned sample isolation under different poisoning rates. Specifically, we attacked CIFAR-10 at poisoning rates of 1%, 5%, 10%, 15%, and 20%, respectively. The attack method used is consistent with the above. We omit the result of SIG at a 1% poisoning rate, as only 50 images were poisoned at this time, which is not enough to successfully attack the dataset. Figure 7 shows the result under different poisoning rates. According to Figure 7, we find that our poisoned sample isolation module can isolate over 94% of poisoned samples under all poisoning rates, indicating that our detection method can effectively identify poisoned samples across various poisoning rates.

F. Study of Parameters on TRAP

Hyper-parameters of DBScan. To evaluate the hyper-parameters of DBScan, we conducted experiments on CIFAR-10 against BadNets-all2one, Trojan, Blend (Strip and Kitty), SIG and CL attacks under different *eps* and *min-samples*. The poisoning rate γ is 10%. We fixed *min-samples* to our default setting when evaluating *eps*, and vice versa. Table V shows the recall and precision of our poisoned sample isolation under different *eps* and *min-samples*. From Table V, it can be observed that when the *eps* is 3, the optimal balance between recall and precision can be achieved. Therefore, in practice, we choose 3 as the default setting for *eps*. On the other hand, we found that our poisoned sample isolation component is robust to the *min-samples* parameter as our results did not change from 10 to 70.

TABLE VI: ACC and ASR before and after defense against five cases attacks on CIFAR-10.

Cases	No Defense		TRAP	
	ACC	ASR	ACC	ASR
One trigger, two target classes	94.51	50.3	94.19	0.12
One trigger, sub2sub attack	94.91	92.2	94.76	0
Two triggers, one target class	94.88	99.48	94.60	0.05
Two triggers, two target classes	94.76	98.78	93.81	0.26
Two triggers, all2all attack	94.99	91.95	94.25	0

Study of detection epoch E_d. We conducted experiments on CIFAR-10 to collect the recall of each epoch within the initial ten epochs for different attacks. The poisoning rate γ is 10%. Figure 8 depicts the obtained results. We observed that, except for the BadNets-all2all attack, a significant portion of the poisoned samples were detected within the first five epochs. However, the recall of BadNets-all2all showed a significant improvement in the last five epochs. Considering that the poisoned sample isolation module increases the time complexity, a smaller detection epoch is preferred. As the model gets better at classifying benign samples with training, the likelihood of the component mistakenly filtering out benign samples also increases. Thus, we set E_d to 10 as a trade-off between time complexity and effectiveness.

G. Adaptive Attack

In our work, we adopt the classical defense setting in which adversaries have no information about the defense. However, the adversaries may design adaptive attacks if they possess knowledge of our TRAP. Therefore, we further study two potential adaptive attacks that are aware of TRAP and try to bypass it in this section.

Multi-trigger and Multi-target Attacks. Firstly, we maintain the current threat model but assume that attackers are aware of our defense methods. To isolate poisoned samples, TRAP identifies suspicious clusters if the clusters are not the largest cluster. However, previously, we considered only all2one or all2all attacks with a single type of trigger. When employing adaptive multiple triggers or targeting only a subset of class labels (a variant of the all-to-all attack), the feature distribution of poisoned samples may be disturbed or become more complex, potentially circumventing our criterion for isolating them. Meanwhile, a low poisoning rate might prevent suspicious clusters from detaching from the largest cluster, ultimately leading to the failure of poisoned isolation. Accordingly, we extended the attack modes to the following five cases: (1) One trigger, two target classes; (2) One trigger maps class A to class B, and class C to class D. We refer to this as a sub2sub attack; (3) Two triggers, one target class; (4) Two triggers, two target classes, one of the triggers will only be associated with one target class; (5) Two triggers, all2all attack. In all cases, the poisoning rate was set at 1% (0.5% for each trigger if two triggers were present), triggers were selected from BadNets-all2one and Trojan, and the target classes were set to 0 and 1. Extremely low poisoning rates can make the attack more stealthy, thereby preventing the formation of a suspicious cluster [21]. In the sub2sub attack, we map class 0 to class 1, and class 2 to class 3. Following the prior works [10], [11], we conduct these five cases on CIFAR-10. Table VI presents the values of the ACC and ASR before and after defense. From the table, we can observe that in all five cases, TRAP can still reduce the ASR to below 1% while almost having no impact on ACC (reduction within 1%). This suggests that TRAP can effectively defend against various attack cases, encompassing both single and multiple trigger and target cases that are meticulously designed to bypass the poisoned sample isolation, with poisoning rates as low as 1% or 0.5% respectively.

Loss-based Attacks. Secondly, we broaden our threat model and enable attackers to modify the loss function while TRAP isolates poisoned samples. Under this setting, we design an adaptive loss that minimizes the feature distance between the poisoned samples and the corresponding benign samples. Through this method, the model, during its optimization process, is guided to minimize the distance between poisoned samples and their corresponding benign samples, thereby preventing the formation of a suspicious cluster to achieve an adaptive attack. Accordingly, the new training loss is defined as:

$$\mathcal{L} = \mathcal{L}_{ce}(\mathcal{F}(\boldsymbol{x}), y) + \mathcal{L}_{ce}(\mathcal{F}(\bar{\boldsymbol{x}}), y_t) + \alpha \mathcal{L}_{MSE}(f(\boldsymbol{x}), f(\bar{\boldsymbol{x}})) \quad (2)$$

where $\mathcal{F}$ is the model we train, f is the feature extractor of $\mathcal{F}$, α is a hyper-parameter that controls the influence of the third loss item, $\boldsymbol{x}$ is benign samples, and $\bar{\boldsymbol{x}}$ is the corresponding poisoned sample. Note that adaptive attacks based on modifying loss may have a certain impact on the parameters of the model, resulting in relatively low accuracy. Thus, we empirically optimize the value of α and find that this attack achieves a good trade-off between the ACC and the ASR when $\alpha = 5$. Following the prior works [10], [11], we conduct this adaptive attack on CIFAR-10. We take Equation 2 as the loss function of the poisoned sample isolation module. The loss-based adaptive attack can achieve 93.35% ACC and 100% ASR without any defense. However, this attack can obtain 92.71% ACC and 37.27% ASR under our TRAP, which illustrates our defense can resist the adaptive attack.

In addition to adaptive attacks studied above, we also considered Adap-Blend [21] attacks at extremely low poisoning rates (a poisoning rate of 0.3% and a cover rate of 0.3%), which are detailed in Appendix I respectively. Since TRAP relies on clustering results to identify poisoned samples, its effectiveness diminishes against backdoor attacks where the attack strength has been intentionally weakened, even though it remains effective against the majority of attacks. This is because a weaker attack reduces the efficacy of TRAP's clustering mechanism. In such scenarios, nearly all backdoor defense methods are affected to some extent; however, TRAP's effectiveness remains superior in comparison. Nevertheless, the reduced attack strength also means that the applicability and success conditions for this type of backdoor attack are more stringent than those for standard attacks.

VII. RELATED WORK

Backdoor Attack. Backdoor attacks are emerging security threats to deep neural networks. In this paper, we focus on defending against poisoning-based backdoor attacks, where the attacker inject backdoors by adding a small portion of

poisoned samples to the training datasets. Besides those discussed in Section II, there are also related attacks in other tasks such as natural language processing [49], time series data [50], and physical world patterns [51], and with different attacker's capacities [52]–[55]. Discussion on this part of the work is out of the scope of this paper, and it goes beyond the scope of our threat model. Beyond injecting backdoors via maliciously manipulating the training dataset, the adversaries can also launch backdoor attacks when users use third-party platforms [55] or third-party models [56], [57]. In the case of using a third-party training platform, attackers can modify the training process (e.g., loss function) to inject a backdoor. For instance, Shafahi et al. [55] proposed an optimization-based method for crafting poisoned data via feature collisions by modifying the loss function. In the case of using third-party models, attackers can maliciously modify the model parameters to inject backdoor. For example, Garg et al. [57] proposed injecting a benign model with a backdoor by adding adversarial perturbations to the model parameters. The defense, however, against these attacks is not the focus of this paper.

Backdoor Defense. The defense against backdoor attacks can be mainly divided into two categories: empirical backdoor defense and certified backdoor defense. Among them, empirical backdoor defense accounts for the vast majority. Empirical backdoor defenses can be generally divided into the following five categories [6]. Preprocessing-based defenses [58], [59] preprocess the dataset before training the model with poisoned data to disrupt the connection between the trigger and the target class. Model reconstruction-based defenses [8], [60] involve directly modifying the suspicious model, such as manipulating its neurons, to remove the backdoors in the model. Trigger synthesis-based defenses [17], [25], [61] first synthesize trigger patterns and then remove the hidden backdoors by suppressing the triggers. Detection-based defenses [30], [34], [35], [62] typically inspect the neurons of the model or perform statistical analysis of the feature representation of the dataset to filter out data after model training. Different from those defenses, TRAP analyzes the feature representation of the dataset during the early stage of training. Poison suppression-based defenses [7], [9]–[14] suppress the expression of triggers through security or robustness training, thereby achieving the effect of removing backdoors. We have had a detailed discussion on this part in Section II-C. SEEP [63] is designed for NLP tasks that identify poisoned samples during the training process via clustering. However, TRAP directly identifies poisoned samples via clustering early in training. In contrast, SEEP first needs to identify typical seeding samples during the training process (i.e., leverages training dynamics) and then uses label propagation to identify the remaining poisoned samples. While TRAP is simpler and more efficient than SEEP at isolating poisoned samples, the latter can utilize information from the entire training process. This might allow SEEP to achieve better effectiveness in certain complex scenarios. We plan to enhance TRAP in this aspect in the future. SHERPA [64] utilizes Shapley Additive Explanations (SHAP) for feature attribution and the HDBSCAN clustering technique to identify potential poisoners and provide a basis for their exclusion. The primary difference is that SHERPA focuses mainly on backdoor detection, whereas TRAP includes both detection and removal steps. Furthermore, SHERPA is specifically designed for the distributed training setting of federated learning. QUEEN [65] defends against model extraction attacks by utilizing the distance between cluster centers in the feature space, whereas TRAP employs the boundary distance between different clusters as a metric. SMP [66] is designed to mitigate membership inference and model inversion attacks, adopting a strategy similar to TRAP by proposing fake targets. Similarly, Chen et al. [67] attack image processing networks by forging watermarks.

VIII. CONCLUSION

In this paper, we revisit the inherent mechanism of poisoning-based backdoor attacks and conduct an in-depth analysis of the behavior of such attacks in the hidden space during the model training process. We make two key observations: first, poisoned samples form a distinct cluster distant from benign samples in the early training stages. Second, when the model is fully trained, the classifier part of the model contains a unique pathway from the backdoor features to the target label. Based on the two observations, we propose a novel defense method, TRAP, which differentiates between poisoned and benign samples and removes the backdoor. Our evaluation results indicate that TRAP has significantly better defense performance compared to existing poison suppression-based defense methods across all datasets and attacks.

REFERENCES

[1] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. F. R. Jesus, R. F. Berriel, T. M. Paixão, F. W. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. D. Souza, "Self-driving cars: A survey," *Expert Syst. Appl.*, vol. 165, p. 113816, 2021.

[2] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, 2019.

[3] A. M. Özbayoglu, M. U. Gudelek, and O. B. Sezer, "Deep learning for financial applications : A survey," *Appl. Soft Comput.*, vol. 93, p. 106384, 2020.

[4] Z. Wang, J. Ma, X. Wang, J. Hu, Z. Qin, and K. Ren, "Threats to training: A survey of poisoning attacks and defenses on machine learning systems," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 134:1–134:36, 2023.

[5] 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.

[6] Y. Li, B. Wu, Y. Jiang, Z. Li, and S. Xia, "Backdoor learning: A survey," *CoRR*, vol. abs/2007.08745, 2020.

[7] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: Training clean models on poisoned data," in *NeurIPS*, 2021, pp. 14 900–14 912.

[8] ——, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," in *ICLR*. OpenReview.net, 2021.

[9] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitras, and N. Papernot, "On the effectiveness of mitigating data poisoning attacks with gradient shaping," *CoRR*, vol. abs/2002.11497, 2020.

[10] K. Huang, Y. Li, B. Wu, Z. Qin, and K. Ren, "Backdoor defense via decoupling the training process," in *ICLR*. OpenReview.net, 2022.

[11] Z. Wang, H. Ding, J. Zhai, and S. Ma, "Training with more confidence: Mitigating injected and natural backdoors during training," in *NeurIPS*, 2022.

[12] W. Chen, B. Wu, and H. Wang, "Effective backdoor defense by exploiting sensitivity of poisoned samples," in *NeurIPS*, 2022.

[13] Z. Zhang, Q. Liu, Z. Wang, Z. Lu, and Q. Hu, "Backdoor defense via deconfounded representation learning," in *CVPR*, 2023, pp. 12 228–12 238.

[14] K. Gao, Y. Bai, J. Gu, Y. Yang, and S.-T. Xia, "Backdoor defense via adaptively splitting poisoned dataset," in *CVPR*, 2023, pp. 4005–4014.

[15] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in CNNS by training set corruption without label poisoning," in *ICIP*. IEEE, 2019, pp. 101–105.

[16] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," *CoRR*, vol. abs/1912.02771, 2019.

[17] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *IEEE Symposium on Security and Privacy*. IEEE, 2019, pp. 707–723.

[18] M. Du, Z. Chen, C. Liu, R. Oak, and D. Song, "Lifelong anomaly detection through unlearning," in *CCS*. ACM, 2019, pp. 1283–1297.

[19] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *NDSS*. The Internet Society, 2018.

[20] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *CoRR*, vol. abs/1712.05526, 2017.

[21] X. Qi, T. Xie, Y. Li, S. Mahloujifar, and P. Mittal, "Revisiting the assumption of latent separability for backdoor defenses," in *ICLR*. OpenReview.net, 2023.

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

[23] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[24] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in *AAAI*. AAAI Press, 2021, pp. 1148–1156.

[25] Y. Liu, W. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: scanning neural networks for back-doors by artificial brain stimulation," in *CCS*. ACM, 2019, pp. 1265–1282.

[26] T. A. Nguyen and A. T. Tran, "Input-aware dynamic backdoor attack," in *NeurIPS*, 2020.

[27] ——, "Wanet - imperceptible warping-based backdoor attack," in *ICLR*. OpenReview.net, 2021.

[28] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *ICCV*. IEEE, 2021, pp. 16 443–16 452.

[29] S. Baluja, "Hiding images in plain sight: Deep steganography," in *NIPS*, 2017, pp. 2069–2079.

[30] 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 Symposium*. USENIX Association, 2021, pp. 1541–1558.

[31] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 1597–1607.

[32] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *ICCV*. IEEE, 2019, pp. 322–330.

[33] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[34] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NeurIPS*, 2018, pp. 8011–8021.

[35] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. M. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," in *SafeAI@AAAI*, ser. CEUR Workshop Proceedings, vol. 2301. CEUR-WS.org, 2019.

[36] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*. AAAI Press, 1996, pp. 226–231.

[37] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–136, 1982.

[38] E. Schubert, J. Sander, M. Ester, H. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, 2017.

[39] K. Khan, S. ur Rehman, K. Aziz, S. Fong, S. Sarasvady, and A. Vishwa, "DBSCAN: past, present and future," in *ICADIWT*. IEEE, 2014, pp. 232–238.

[40] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, 2008.

[41] I. T. Jolliffe, *Principal Component Analysis*, ser. Springer Series in Statistics. Springer, 1986.

[42] N. Halko, P. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, 2011.

[43] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[44] A. Krizhevsky, "Learning multiple layers of features from tiny images," Jan 2009.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*. IEEE Computer Society, 2016, pp. 770–778.

[47] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *NeurIPS*, 2020.

[48] D. M. Chan, R. Rao, F. Huang, and J. F. Canny, "Gpu accelerated t-distributed stochastic neighbor embedding," *Journal of Parallel and Distributed Computing*, vol. 131, pp. 1–13, 2019.

[49] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang, "Badnl: Backdoor attacks against NLP models with semantic-preserving improvements," in *ACSAC*. ACM, 2021.

[50] Y. Jiang, X. Ma, S. M. Erfani, and J. Bailey, "Backdoor attacks on time series: A generative approach," *CoRR*, vol. abs/2211.07915, 2022.

[51] E. Wenger, J. Passananti, A. N. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, "Backdoor attacks against deep learning systems in the physical world," in *CVPR*. Computer Vision Foundation / IEEE, 2021, pp. 6206–6215.

[52] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *CCS*. ACM, 2020, pp. 113–131.

[53] I. Shumailov, Z. Shumaylov, D. Kazhdan, Y. Zhao, N. Papernot, M. A. Erdogdu, and R. J. Anderson, "Manipulating SGD with data ordering attacks," in *NeurIPS*, 2021, pp. 18 021–18 032.

[54] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," in *EuroS&P*. IEEE, 2022, pp. 703–718.

[55] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *NeurIPS*, 2018, pp. 6106–6116.

[56] A. S. Rakin, Z. He, and D. Fan, "TBT: targeted neural network attack with bit trojan," in *CVPR*. Computer Vision Foundation / IEEE, 2020, pp. 13 195–13 204.

[57] S. Garg, A. Kumar, V. Goel, and Y. Liang, "Can adversarial weight perturbations inject neural backdoors," in *CIKM*. ACM, 2020, pp. 2029–2032.

[58] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *ACSAC*. ACM, 2020, pp. 897–912.

[59] Y. Li, T. Zhai, Y. Jiang, Z. Li, and S. Xia, "Backdoor attack in the physical world," *CoRR*, vol. abs/2104.02361, 2021.

[60] A. Li, S. Zhao, X. Ma, M. Gong, J. Qi, R. Zhang, D. Tao, and R. Kotagiri, "Short-term and long-term context aggregation network for video inpainting," in *ECCV (4)*, ser. Lecture Notes in Computer Science, vol. 12349. Springer, 2020, pp. 728–743.

[61] G. Shen, Y. Liu, G. Tao, S. An, Q. Xu, S. Cheng, S. Ma, and X. Zhang, "Backdoor scanning for deep neural networks through k-arm optimization," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 9525–9536.

[62] Y. Zeng, W. Park, Z. M. Mao, and R. Jia, "Rethinking the backdoor attacks' triggers: A frequency perspective," in *ICCV*. IEEE, 2021, pp. 16 453–16 461.

[63] X. He, Q. Xu, J. Wang, B. I. P. Rubinstein, and T. Cohn, "SEEP: training dynamics grounds latent representation search for mitigating backdoor poisoning attacks," *Trans. Assoc. Comput. Linguistics*, vol. 12, pp. 996–1010, 2024.

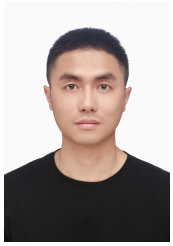
[64] C. Sandeepa, B. Siniarski, S. Wang, and M. Liyanage, "SHERPA: explainable robust algorithms for privacy-preserved federated learning in future networks to defend against data poisoning attacks," in *SP*. IEEE, 2024, pp. 4772–4790.

[65] H. Chen, T. Zhu, L. Zhang, B. Liu, D. Wang, W. Zhou, and M. Xue, "QUEEN: query unlearning against model extraction," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 2143–2156, 2025.

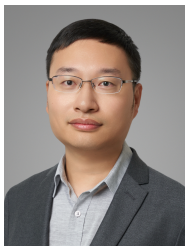
[66] H. Chen, T. Zhu, S. Ji, and W. Zhou, "Stand-in model protection: Synthetic defense for membership inference and model inversion attacks," *Knowl. Based Syst.*, vol. 316, p. 113339, 2025.

[67] H. Chen, T. Zhu, C. Liu, S. Yu, and W. Zhou, "High-frequency matters: Attack and defense for image-processing model watermarking," *IEEE Trans. Serv. Comput.*, vol. 17, no. 4, pp. 1565–1579, 2024.
[68] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *SIGMOD Conference*. ACM Press, 1999, pp. 49–60.
[69] P. S. Segura, V. Singla, L. Fowl, J. Geiping, M. Goldblum, D. Jacobs, and T. Goldstein, "Poisons that are learned faster are more effective," in *CVPR Workshops*. IEEE, 2022, pp. 197–204.
[70] B. Zhu, Y. Qin, G. Cui, Y. Chen, W. Zhao, C. Fu, Y. Deng, Z. Liu, J. Wang, W. Wu, M. Sun, and M. Gu, "Moderate-fitting as a natural backdoor defender for pre-trained language models," in *NeurIPS*, 2022.
[71] H. Liu, J. Jia, and N. Z. Gong, "Poisonedencoder: Poisoning the unlabeled pre-training data in contrastive learning," in *USENIX Security Symposium*. USENIX Association, 2022, pp. 3629–3645.
[72] E. Wallace, T. Z. Zhao, S. Feng, and S. Singh, "Concealed data poisoning attacks on NLP models," in *NAACL-HLT*. Association for Computational Linguistics, 2021, pp. 139–150.
[73] 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*. OpenReview.net, 2021.
[74] C. Li, R. Pang, Z. Xi, T. Du, S. Ji, Y. Yao, and T. Wang, "An embarrassingly simple backdoor attack on self-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 4367–4378.
[75] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 1597–1607.
[76] S. Jang, J. S. Choi, J. Jo, K. Lee, and S. J. Hwang, "Silent branding attack: Trigger-free data poisoning attack on text-to-image diffusion models," in *CVPR*. Computer Vision Foundation / IEEE, 2025, pp. 8203–8212.
[77] Y. Xu, J. Yao, M. Shu, Y. Sun, Z. Wu, N. Yu, T. Goldstein, and F. Huang, "Shadowcast: Stealthy data poisoning attacks against vision-language models," in *NeurIPS*, 2024.
[78] S. Shan, W. Ding, J. Passananti, S. Wu, H. Zheng, and B. Y. Zhao, "Nightshade: Prompt-specific poisoning attacks on text-to-image generative models," in *SP*. IEEE, 2024, pp. 807–825.

Chun Li (Member, IEEE) received the B.S. degree from Nanjing University, Nanjing, China, in 2022, where he is currently pursuing the Ph.D. degree with the State Key Laboratory for Novel Software Technology under the Supervision of Prof. Xuandong Li and Prof. Minxue Pan. His current research interests include the artificial intelligence for software engineering and security.



Zhong Li (Member, IEEE) received the Ph.D. degrees in computer science and technology from Nanjing University, Nanjing, China, in 2023. He is currently an Assistant Researcher with the State Key Laboratory for Novel Software Technology and the Software Institute, Nanjing University. His research interests include intelligent software engineering and Trustworthy AI.



Minxue Pan (Member, IEEE) received the B.S. and Ph.D. degrees in computer science and technology from Nanjing University, Nanjing, China, in 2006 and 2014, respectively. He is a Full Professor with the State Key Laboratory for Novel Software Technology and the Software Institute, Nanjing University. His research interests include software modeling and verification, software analysis and testing, cyber-physical systems, mobile computing, and intelligent software engineering.

Xuandong Li received the B.S., M.S., and Ph.D. degrees from Nanjing University, Nanjing, China, in 1985, 1991, and 1994, respectively. He is a Full Professor with the School of Computer Science, Nanjing University. His research interests include formal support for the design and analysis of reactive, distributed, real-time, hybrid, and cyber–physical systems, as well as software testing and verification.