Proceedings of Machine Learning Research 299:1–32, 2025 Conference on Applied Machine Learning for Information Security

PD-AutoR: Towards Automatic Restoration of Poisoned Examples in Machine Learning

Haoyang Chen HAOYANGC@MTU.EDU
Xinyun Liu XINYUNL@MTU.EDU
Xu Zhou XZHOU4@MTU.EDU
Ziao Jiao EXPLAIN.QUANSHI@GMAIL.COM
Xinyu Lei* XINYULEI@MTU.EDU
Department of Computer Science, Michigan Technological University

Editor: Edward Raff and Ethan M. Rudd

Abstract

Machine learning (ML)-based systems are increasingly being deployed in real-world applications with high societal impacts. A pivotal factor that contributes to the success of ML techniques is the availability of high-quality training datasets. However, there are many attack vectors (exploitable by attackers) to launch various data poisoning (DP) attacks against ML systems since training datasets are often collected from untrusted data sources. One direct negative consequence of DP attacks is that the data quality of the poisoned dataset can be significantly deteriorated compared with the original clean dataset. To mitigate the low-data-quality issue, we design a neural network (NN)-based Poisoned Data Automatic Restoration (PD-AutoR) engine to automatically detect and restore PD prior to ML model training. Our high-level methodology is to develop a transductive learning-supported pipeline, which allows the target PD (that needs to be restored) to be used in PD-AutoR training, so PD-AutoR can achieve very high restoration accuracy. In addition, we design transformer-based networks (with a self-attention mechanism) to enable PD-AutoR to precisely and automatically pay attention to the areas that need to be restored, enabling PD-AutoR to restore the PD even if the attacker's poisoning strategy is agnostic. Our theoretical analysis and preliminary experimental results show that PD-AutoR can simultaneously fulfill the three design goals including high PD detection accuracy, high PD restoration accuracy, and strong fault tolerance.

Keywords: machine learning security, data poisoning attack, automatic restoration, transformer

1. Introduction

Background and Motivation. Data is recognized as one of the most vital and valuable resources in the 21st century. A recent survey shows that about 96% of machine learning (ML)-driven businesses suffer from the issue of "not enough data" (i.e., insufficient-data issue) and the issue of "bias or errors in data" (i.e., low-data-quality issue) (Cos). However, acquiring a large amount of high-quality training data for data-hungry ML tasks is typically a costly endeavor. According to Dimensional Research (Cos), it would cost around $70,000 to collect a dataset with 100,000 samples using a crowdsourcing platform (e.g., Amazon Mechanical Turk). Moreover, to get 100,000 data samples properly labeled, the cost is

* Corresponding Author

between $8,000 and $80,000 depending on the complexity of the annotation. With the rise of ML techniques, a training dataset (with large quantity and high quality) will be increasingly precious and valuable.

In the data collection or management phase, there are many attack vectors (exploitable by attackers) to launch various **D**ata **P**oisoning (DP) attacks against ML systems because the required training datasets are often collected from untrusted data sources. Some examples are given as follows. 1) The attacker can publish **P**oisoned **D**ata (PD) online and simply wait for victims to scrape data from the web. 2) The attacker may sell poisoned datasets via various data trading markets (e.g., online data trading platforms). 3) The attacker can serve as a data contributor and upload PD to the crowdsourcing platform that collects training data (Miao et al., 2018; Tahmasebian et al., 2020). 4) The victim may outsource the model training task to an untrusted third-party (e.g., cloud server), the third-party (i.e., attacker) may poison the training dataset (Ma et al., 2022; Wang et al., 2023). Nowadays, DP attacks are ubiquitous across a wide range of real-world ML applications. For example, the DP attack can corrupt Google's image search engine (Jew), Microsoft's AI chatbot (Mic), and the face recognition models (Radiya-Dixit et al., 2021). According to a recent survey (Kumar et al., 2020), industry practitioners rank DP attacks as the most worrisome threat to ML systems.

Limitations of Prior Art. To address the low-data-quality issue originating from DP attacks, previous solutions can be roughly divided into the following two types. ① The detection-and-discard solutions (e.g., (Chen et al., 2018; Tran et al., 2018; Hayase and Kong, 2021)) work by first detecting the PD in the training dataset and then directly discarding them. The major limitation of these solutions is that they may exacerbate the insufficient-data issue encountered by many data-driven businesses. If the poisoning rate (i.e., the fraction of PD in the training dataset) is high, discarding a significant portion of the valuable training dataset implies a big loss for the data owner. For example, the model hijacking (MH) attack (Salem et al., 2022) needs to poison about 33% of the training dataset. Moreover, the model unavailability attacks (Wen et al., 2023; Feng et al., 2019; Tao et al., 2022) needs to add a small noise/perturbation on all training data (i.e., the poisoning rate is 100%) in order to achieve a good attack performance. ② The image inpainting solutions (e.g., (Pathak et al., 2016; Shin et al., 2020; Wang et al., 2019, 2018; Doan et al., 2020)) aim to train a neural network (NN) to repair the damaged images. These image inpainting solutions suffer from two drawbacks when applied in DP attack scenarios. First, in the traditional image inpainting tasks, the properly labeled damaged dataset and clean dataset are available (or can be synthesized) for the supervised inpainting network training. In contrast, in DP attack scenarios, the PD and clean data are mixed in a single dataset in which there are no labels to indicate whether data is poisoned or clean, so the traditional image inpainting solutions cannot be directly adopted. Second, in the traditional image inpainting tasks, image areas that need to be inpainted are typically clearly segmented. In contrast, in DP attack scenarios, the information about how an image is modified by attackers is completely agnostic, so it is impossible to directly use the traditional image inpainting solutions to restore PD.

Proposed PD-AutoR Engine. In this project, we propose to design a **P**oisoned **D**ata **Auto**matic **R**estoration (PD-AutoR) engine to automatically detect and restore PD. If PD-AutoR can accurately restore all PD to their original state prior to ML model training,

then the low-data-quality issue can be alleviated. PD-AutoR leverages an NN-based data content restorator to restore the PD. Given a new dataset as input, PD-AutoR processes it and then outputs a sanitized dataset.

Technical Challenges and Proposed Solutions. There are four major technical challenges in PD-AutoR design.

1) There are no labels to indicate whether a data sample is poisoned or clean, so it is challenging to train PD-AutoR without properly labeled poisoned dataset and clean dataset. To address this challenge, we design a high-performance PD detector by using the proposed over-parameterization strategy and training loss bifurcation strategy. Given a newly collected dataset, the PD detector can divide it into a flagged poisoned dataset and a flagged clean dataset, which can be used for PD-AutoR training later.
2) It is challenging for PD-AutoR to achieve high content restoration accuracy. To address this challenge, our high-level methodology is to develop a PD-AutoR pipeline that can support transductive learning (Joachims, 2003; Rossi et al., 2018). Different from the traditional supervised learning (i.e., inductive learning (Mao et al., 2021; Zeng et al., 2019)), transductive learning allows the test data (i.e., the PD to be restored) to be used in PD-AutoR training. By incorporating the test dataset in training, the training loss of the test data can be directly optimized to an extremely small value. Consequently, the developed transductive learning-supportive pipeline (used in PD-AutoR) can achieve significantly higher restoration accuracy than the traditional supervised learning-based image inpainting solutions.
3) Different DP attacks may use different types of triggers/perturbations, so it is challenging to design a universal PD-AutoR engine that can handle any type of triggers/perturbations. To tackle this issue, we design a vision transformer (ViT)-aided (Dosovitskiy et al., 2020) data content restorator (denoted as R) with a self-attention mechanism to precisely and automatically pay attention to the areas that are stamped with triggers/perturbations. The self-attention mechanism (along with several other techniques) enables the restorator to automatically fix the PD even if the trigger/perturbation patterns are agnostic.
4) Since the designed PD detector may lead to false positives (FPs), it is not an easy task to design a content restorator R with strong FP fault tolerance. To handle the FPs, we design a content-preserving loss used in R training. Once R is well-trained, when taking clean data (i.e., an FP) as input for data restoration, R is nearly an identity mapping, so the quality of the R-output data is barely deteriorated. Hence, PD-AutoR can achieve strong FP tolerance.

Broader Impacts. PD-AutoR brings to the following three broader impacts, which further necessitate PD-AutoR design. ① PD-AutoR can be used to restore PD not only in newly collected datasets but also in all legacy datasets that contain PD. ② Since an accumulated dataset usually can be used for training many ML models, the restoration of PD by PD-AutoR can permanently benefit all consecutive ML models trained on the accumulated dataset. From this perspective, the benefit of PD-AutoR is far-reaching and keeps augmenting in the long run. ③ We note that PD-AutoR can be developed in cloud servers, providing the business model of restoration as a service (RaaS). Clients can call the cloud-host PD-AutoR API in a pay-per-use manner. Therefore, resource-limited devices

(e.g., IoT devices, smartphones) can also enjoy PD-AutoR with the help from a powerful cloud server.

Contributions. The main contributions of this paper are summarized as follows.

- To the best of our knowledge, we are the first to develop an automatic pipeline to achieve restoration of poisoned examples in a trigger/perturbation-agnostic and fault-tolerant manner.
- We develop multiple innovative techniques (including the over-parameterization strategy and the training loss bifurcation strategy) to facilitate precise PD detection.
- We leverage multiple innovative techniques (including the transductive learning technique, the self-attention restorator, and the multi-granularity discriminator) to achieve high PD restoration accuracy.
- We are the first to propose the business model of restoration as a service (RaaS), in which clients can call the cloud-host PD-AutoR API in a pay-per-use manner. Therefore, this project could extend the border of cloud computing services and bring in new business growth possibilities for cloud computing service providers.

2. Problem Formulation

Data & Task. In this work, our focused data type is the image data. The considered ML task is the most widely used DNN-based classification task.

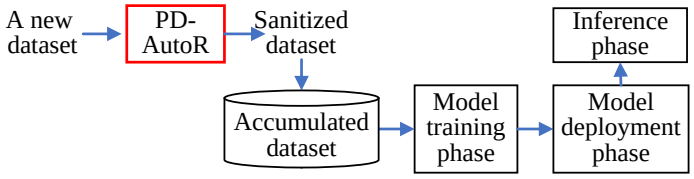

Figure 1: The role of PD-AutoR engine in the ML pipeline.

PD-AutoR role & Modes. Fig. 1 shows the role of PD-AutoR engine in the ML pipeline, in which PD-AutoR works prior to the ML model training. PD-AutoR operates in the following two modes. 1) One-Time Mode: in this mode, a dataset is fed to PD-AutoR for one-time data sanitization service. 2) Accumulation Mode: in this mode, PD-AutoR works in a continuous manner. Once a newly collected dataset reaches a certain size, it is fed to PD-AutoR, then PD-AutoR restores the detected PD. Next, the sanitized dataset is added to the accumulated dataset that will be used for ML model training.

Threat Model. We consider that the attacker can poison a portion of the training dataset used by the victim for ML model training. The attacker can launch any kind of DP attack (known or unknown) using any type of trigger/perturbation pattern. Our solution PD-AutoR should work well in a trigger/perturbation-agnostic manner: we do not assume that PD-AutoR has any knowledge of the trigger/perturbation patterns including types, shapes, positions, sizes, etc.

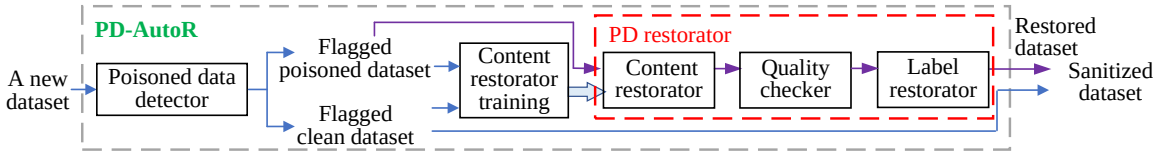

Figure 2: PD-AutoR system model.

System Model. Fig. 2 depicts the proposed PD-AutoR system model, which consists of three modules: the PD detector module, the content restorator training module, the PD restorator module. For the PD restorator module, it can be further divided into three components: the content restorator, the quality checker, and the label restorator. PD-AutoR works as follows. First, a new dataset is fed to the PD detector. The detector aims to detect the PD and then dividing the dataset into two datasets: the flagged poisoned dataset and the flagged clean dataset. Then, both datasets are used to train an NN-based content restorator. Next, the flagged poisoned dataset is fed to the content restorator and then to the quality checker. Only the restored data with sufficiently high restoration quality is forwarded to the label restorator, which aims to give a label to the restored data. Finally, by combining the flagged clean dataset with the restored dataset, a sanitized dataset is generated.

Design Goals. There are three design goals in PD-AutoR.

- High Detection Accuracy. PD-AutoR should achieve high PD detection accuracy. This goal can be further divided into the following two sub-goals. 1) *High Precision.* In the detected poisoned dataset, the fraction of the correctly detected PD among all the detected data instances should be as high as possible. 2) *High Recall.* The fraction of the detected PD (in the detected poisoned dataset) among all exiting PD instances (in the input dataset) should be as high as possible.
- High Restoration Accuracy. PD-AutoR should achieve high restoration accuracy. This goal can be further divided into the following two sub-goals. 1) *High Content Restoration Accuracy.* The content of the restored data should be close to their original content as much as possible. 2) *High Label Restoration Accuracy.* The label of the restored data should be their original label with a probability as high as possible.
- Strong Fault Tolerance. PD-AutoR should achieve strong fault tolerance. This goal can be further divided into the following two sub-goals. 1) *Strong False Positive (FP) Tolerance.* An FP occurs if a clean data is detected as a PD. PD-AutoR should achieve high restoration accuracy in the presence of FPs. 2) *Strong False Negative (FN) Tolerance.* An FN occurs if a PD is not detected. PD-AutoR should achieve high restoration accuracy in the presence of FNs.

3. PD-AutoR Design

As shown in Fig. 2, PD-AutoR consists of three modules: PD detector module, content restorator training module, and PD restorator module. The three modules are elaborated as follows.

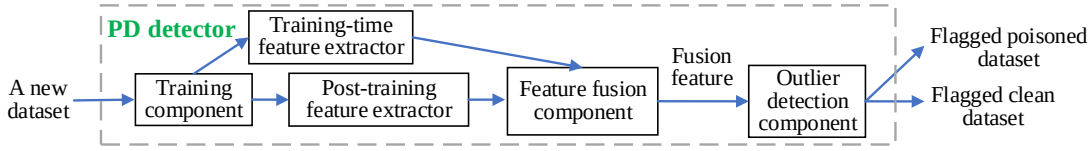

Figure 3: The PD detector pipeline.

3.1. PD Detector Module

3.1.1. PD Detector Overview

The proposed PD detector pipeline is exhibited in Fig. 3. It is composed of five components: 1) the training component, 2) the post-training feature extractor, 3) the training-time feature extractor, 4) the feature fusion component, and 5) the outlier detection component. The PD detector works as follows. First, a new dataset is fed to the training component, which is responsible for training the ML model using the new dataset. When the training process is proceeding, the training-time feature extractor collects the training-time information. After training, the post-training information is collected by the post-training feature extractor. Next, each feature extractor produces a feature vector. Afterward, the two feature vectors are concatenated to generate a fused feature, which is fed to the outlier detection component. Lastly, the outlier detection component invokes an outlier detection algorithm to divide the input dataset into two parts: the flagged poisoned dataset, and the flagged clean dataset.

3.1.2. Design Rationale

The state-of-the-art PD detection solutions (e.g., (Chen et al., 2018; Hayase and Kong, 2021)) show that the post-training feature representation can be used to detect PD. To improve the PD detection accuracy, we employ two strategies as follows. First, we employ the over-parameterization strategy to generate a more precise post-training feature representation that can facilitate the PD detection. Second, we leverage the training loss bifurcation strategy to generate an additional training-time feature representation. By combining both the post-training feature and training-time feature, the PD detection accuracy is higher than the prior solutions. The two strategies are further illustrated in details as follows.

3.1.3. Technical Challenges and Proposed Solutions

There are two major technical challenges.

1) It is challenging to design a fine-grained post-training feature extractor. To address this challenge, we employ the *over-parameterization strategy*, which is developed based on the following crucial observation.

Observation 1

For the post-training feature extractor, the feature extracted from the well-trained over-parameterized DNN model achieves a higher PD detection accuracy. The higher degree of the over-parameterization, the higher PD detection accuracy. Note that over-parameterization means having more model parameters than necessary.

The intuitive explanation for this observation is that the over-parameterized strategy leads to an overfitted ML model after training. The overfitted ML model does not generalize well on an unknown test dataset, but it has a higher performance on the known training dataset (i.e., the poisoned dataset). This indicates that the overfitted over-parameterized DNN model can capture more fine-grained information of the poisoned dataset to better distinguish between clean data and PD therein. Note that the PD detection solely works on the poisoned dataset itself, so leveraging the over-parameterized DNN model can lead to higher PD detection accuracy.

2) It is challenging to extract the useful training-time feature to effectively amplify the distinctive characteristics (between a PD and a clean data) to facilitate PD detection. To address this challenge, we develop the *training loss bifurcation strategy* on the basis of the following critical observation.

Observation 2

When a DNN model is training on a poisoned dataset, the average training losses of a poisoned sample and a clean sample exhibit different decreasing rates. The earlier the training epoch, the larger the difference in their decreasing rates.

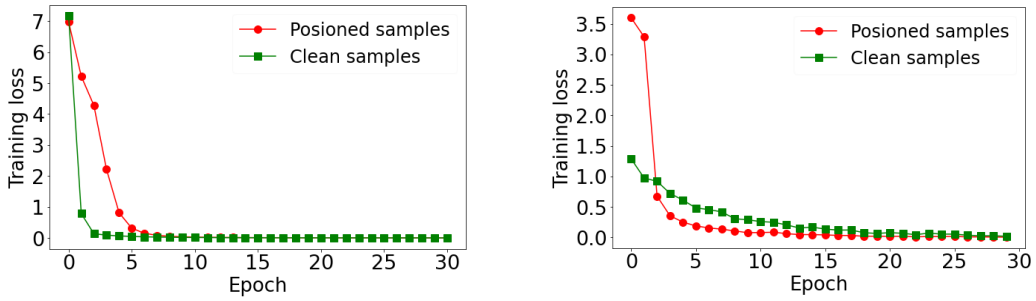

Figure 4: The average training loss v.s. epoch number on a) MINST and b) CIFAR-10.

We have Observation 2 in the following experiment. In the experiment, two DNNs are trained on MNIST and CIFAR-10, respectively. For both datasets, we randomly poison 1% of the training data using the BadNets (Gu et al., 2017) attack. As shown in Fig. 4, the average training loss of the poisoned samples is decreasing slower than that of the clean samples on MNIST, whereas the decreasing rate is opposite on CIFAR-10. The intuitive reason for this observation is that a DP attack imposes a correlation between the PD content (where there is trigger/perturbation stamped) and its label. Such a correlation is

abbreviated as "PD-label correlation". Likewise, the correlation between the clean data (CD) content and its label is referred as "CD-label correlation". It holds that the inherent strengths of the PD-label correlation and the CD-label correlation are usually different. The higher strengths of correlation, the easier and faster to be learned by DNN models. As shown in Fig. 4, compared to the CD-label correlation, the PD-label correlation is slower to be learned on MNIST, but faster to be learned on CIFAR-10, thereby resulting in Observation 2. Accordingly, our solution is to employ the average training loss gaps (in the first several epochs of training) as the extracted training-time feature vector. It is desirable that the distinctive feature patterns (among poisoned samples and clean samples) are helpful to effectively distinguish between them. The mathematical details of the extracted feature vector are presented in the next paragraph.

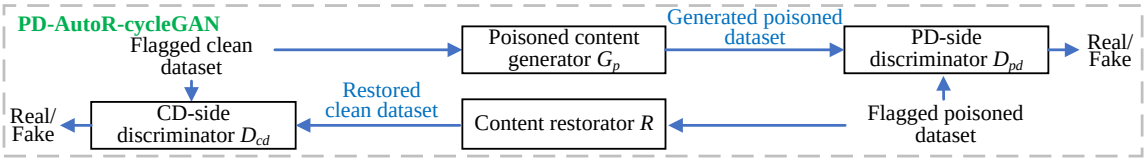

Figure 5: The PD-AutoR-cycleGAN system for R training.

3.1.4. PD Detector Details

As shown in Fig. 3, PD detector consists of 1) training component, 2) post-training feature extractor, 3) training-time feature extractor, 4) feature fusion component, and 5) outlier detection component. Each component is elaborated below. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be the input dataset, where y_i is the label of the i-th data $\mathbf{x}_i$ and $N = |\mathcal{D}|$ is the size of the dataset.

1) **Training Component.** This component is responsible for training the ML model (denoted as $\mathcal{M}$) and forwarding the requested information to the downstream post-training feature extractor and the training-time feature extractor.

2) **Post-Training Feature Extractor.** The state-of-the-art PD detection solution (Hayase and Kong, 2021) shows that the post-training feature representation can be used for PD detection. Our PD detector also leverages the same feature representation in PD detection. Mathematically, on receipt of $\mathcal{M}$, the extracted feature of $\mathbf{x}_i$ is represented as $\mathbf{E}_1(\mathbf{x}_i) = \mathbf{H}_k(\mathbf{x}_i)$, where $\mathbf{H}_k(\mathbf{x}_i)$ is the activations of k-th hidden layer of $\mathcal{M}(\mathbf{x}_i)$ flattened into a single 1D vector.

3) **Training-Time Feature Extractor.** Other than the post-training feature representation, our PD detector also make use of the training-time feature, which is extracted based on Observation 2. During $\mathcal{M}$ training, let $\mathcal{L}_j(\mathbf{x}_i)$ denote the average training loss of $\mathbf{x}_i$ after the training epoch j. The extracted feature of $\mathbf{x}_i$ is represented as $\mathbf{E_2}(\mathbf{x}_i)$, which is designed as

$$\begin{aligned}\mathbf{E}_2(\mathbf{x}_i) = [&\mathcal{L}_2(\mathbf{x}_i) - \mathcal{L}_1(\mathbf{x}_i), \mathcal{L}_3(\mathbf{x}_i) - \mathcal{L}_2(\mathbf{x}_i), \cdots, \\ &\mathcal{L}_j(\mathbf{x}_i) - \mathcal{L}_{j-1}(\mathbf{x}_i)] \ (j = 2, \cdots, d_2),\end{aligned} \tag{1}$$

where d_2 is the number of training epochs. The dimension for $\mathbf{E}_2(\mathbf{x}_i)$ is $d_2 - 1$. Equation (1) means that the average training loss differences between two consecutive epochs are

adopted as the training-time feature vector of $\mathbf{x}_i$. As shown in Fig. 4, the feature vector constructed via Equation (1) can be used to distinguish between PD and CD.

4) **Feature Fusion Component.** In our PD detector, both the post-training feature representation and the training-time feature representation are fused to be used for PD detection. It is expected that using fused feature achieves a better PD detection accuracy than using only the post-training feature. Mathematically, on receipt of both $\mathbf{E}_1(\mathbf{x}_i)$ and $\mathbf{E}_2(\mathbf{x}_i)$, the fused feature is defined as $\mathbf{E}_f(\mathbf{x}_i) = \mathbf{E}_1(\mathbf{x}_i)||\lambda \cdot \mathbf{E}_2(\mathbf{x}_i)$, where λ is a hyperparameter and $||$ means the vector concatenation.

5) **Outlier Detection Component.** On receipt of $\mathbf{E}_f(\mathbf{x}_i)$ $(i = 1, \cdots, N)$, the outlier detection component employs the Singular Value Decomposition (SVD)-based outlier detection algorithm (Tran et al., 2018) to detect the PD. The SVD-based detection algorithm works as follows. First, the feature vectors form a matrix, which is used to perform SVD. Then, the feature vectors with high singular values are flagged as PD. After detection, the dataset $\mathcal{D}$ can be divided into a flagged poisoned dataset (denoted as $\mathcal{D}_p$) and a flagged clean dataset (denoted as $\mathcal{D}_c$). We have $\mathcal{D} = \mathcal{D}_p \bigcup \mathcal{D}_c$ and $\mathcal{D}_p \cap \mathcal{D}_c = \emptyset$.

3.2. Content Restorator Training Module

In PD-AutoR, the content restorator takes an image in D_p as input and outputs a restored image. Fig. 5 shows the designed PD-AutoR-cycleGAN system for the content restorator training. It consists of four NNs: 1) the poisoned content generator (denoted as G_p), 2) the content restorator R, 3) the PD-side discriminator (denoted as D_{pd}), and 4) the clean data (CD)-side discriminator (denoted as D_{cd}).

Loss Functions. The data distribution is denoted as $\mathbf{x} \sim \mathcal{D}_p$ and $\overline{\mathbf{x}} \sim \mathcal{D}_c$. The following three types of loss functions are employed in R training.

- Adversarial Loss. The adversarial loss includes i) the adversarial loss between G_p and D_{pd}; ii) the adversarial loss between R and D_{cd}. For the adversarial loss between G_p and D_{pd}, the objective is expressed as

$$\begin{aligned}\mathcal{L}_{adv}(G_p, D_{pd}, \overline{\mathbf{x}}, \mathbf{x}) =& \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_p}\left[\log D_{pd}(\mathbf{x})\right] + \\ & \mathbb{E}_{\overline{\mathbf{x}}\sim\mathcal{D}_c}\left[\log\left(1 - D_{pd}(G_p(\overline{\mathbf{x}}))\right)\right].\end{aligned} \tag{2}$$

  Similarly, for the adversarial loss between R and D_{cd}, we have

$$\begin{aligned}\mathcal{L}_{adv}(R, D_{cd}, \mathbf{x}, \overline{\mathbf{x}}) =& \mathbb{E}_{\overline{\mathbf{x}}\sim\mathcal{D}_c}\left[\log D_{cd}(\overline{\mathbf{x}})\right] + \\ & \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_p}\left[\log\left(1 - D_{cd}(R(\mathbf{x}))\right)\right].\end{aligned} \tag{3}$$

- Cycle-Consistency Loss. The cycle-consistency loss includes i) the forward cycle-consistency loss and ii) the backward cycle-consistency loss. For the forward cycle-consistency loss, the objective is expressed as

$$\mathcal{L}_{cyc}(G_p, R, \overline{\mathbf{x}}) = \mathbb{E}_{\overline{\mathbf{x}}\sim\mathcal{D}_c}\left[\|R(G_p(\overline{\mathbf{x}})) - \overline{\mathbf{x}}\|_1\right]. \tag{4}$$

  Similarly, the backward cycle-consistency loss is

$$\mathcal{L}_{cyc}(R, G_p, \mathbf{x}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_p}\left[\|G_p(R(\mathbf{x})) - \mathbf{x}\|_1\right]. \tag{5}$$

  A graphical illustration of the forward and backward cycle-consistency loss is depicted in Fig. 6.

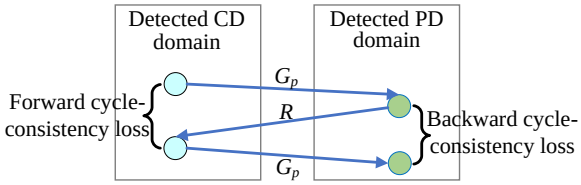

Figure 6: The illustration of the forward and backward cycle-consistency losses.

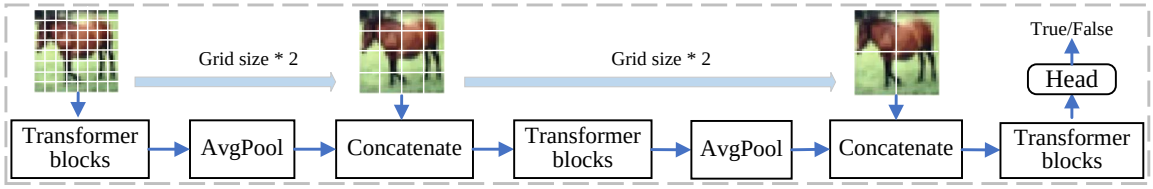

Figure 7: The sketched network architecture of the multi-granularity transformer used by D_{pd} and D_{cd}. Three levels of granularities are depicted in the figure as an example.

- Content-Preserving Loss. The content-preserving loss includes i) the forward content-preserving loss and ii) the backward content-preserving loss. For the forward content-preserving loss, the objective is expressed as

$$\mathcal{L}_{cont}(G_p, \overline{\mathbf{x}}) = \mathbb{E}_{\overline{\mathbf{x}} \sim \mathcal{D}_c} \left[\| G_p(\overline{\mathbf{x}}) - \overline{\mathbf{x}} \|_1 \right]. \tag{6}$$

  Similarly, the backward content-preserving loss is

$$\mathcal{L}_{cont}(R, \mathbf{x}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_p} \left[\| R(\mathbf{x}) - \mathbf{x} \|_1 \right]. \tag{7}$$

Network Architecture.

1) ViT-Aided U-Net. We design the Vision Transformer (ViT)-aided U-Net architecture used by G_p and R. In ViT-Aided U-Net, ViT is placed between the encoder and decoder of the U-Net network. The detailed network architecture of the ViT can be found in (Dosovitskiy et al., 2020). The decoder and encoder use a similar network architecture to the classical U-Net (Ronneberger et al., 2015).

2) Multi-Granularity Transformer. We design the multi-granularity transformer used by D_{pd} and D_{cd}. As shown in Fig. 7, the key idea is to divide the input data into multiple patches according to a certain grid size. The grid size gradually increases from a small size to a large size. For each granularity, the input data patches are first fed to transformer blocks and then have the average pooling operation. The transformer blocks use a similar structure to the classic ViT. Next, the output data patches are concatenated with the data patches from the next granularity. This process repeats until the last transformer blocks (see Fig. 7). Finally, a classification head outputs true or false.

3.3. PD Restorator Module

The PD restorator module consists of the content restorator, the quality checker, and the label restorator.

3.3.1. Content Restorator

Once R is well-trained via a min-max game following Equation (3), it is forced to generate the data that has the same distribution as $\mathcal{D}_c$. Mathematically, for a data $\mathbf{x}_i \in \mathcal{D}_p$, after feeding into R, it is restored into its clean version $\overline{\mathbf{x}}_i$.

3.3.2. Quality Checker and Label Restorator

After the content restorator, $\overline{\mathbf{x}}_i$ is forwarded to the label restorator, which is responsible for generating a restored label. In the label restorator, the restored label of $\overline{\mathbf{x}}_i$ is determined according to its restored content. The key idea used in the label restorator is: the label restorator sets the label of $\mathbf{x}$ to be the label with the maximum classification confidence (i.e., maximum likelihood), which is produced by a classifier $\mathcal{C}$ trained on $\mathcal{D}_c$. In other words, the label of $\overline{\mathbf{x}}_i$ is determined by the soft label (with the largest value) of $\mathcal{C}$ when taking $\overline{\mathbf{x}}_i$ as input. After the quality checking, only those data above a pre-specific threshold logic value would be left for downstream tasks.

4. PD-AutoR Analysis

Content-Preserving Generative-Model-Based Technique. PD-AutoR employs a content-preserving generative model (GM)-based technique. On one hand, PD-AutoR tries to have a minimum scale of modification on a PD by using the content-preserving loss. On the other hand, PD-AutoR tries to remove the triggers/perturbations (and generate the missing content if needed) to make the restored image distribute identically with $\mathcal{D}_c$. Thus, PD-AutoR-restored data tries to preserve the content information of the original data in $\mathcal{D}_p$ in the best possible manner.

Worst-Case Performance Guarantee. Suppose that an attacker masks all the content of a original clean data and inserts a pure white image into the dataset. Taking the white image as input, R will output a restored image that distributes identically to the detected clean dataset, so the R-output data will not be harmful to the performance of downstream ML tasks. In this worst-case, R is reduced to perform the generative-model-based data augmentation task. Thus, PD-AutoR provides a worst-case performance guarantee to the downstream ML tasks.

Verifiable Performance Improvement. PD-AutoR can be used in a trustworthy and benefit-verifiable manner. Given a new dataset, users can train the downstream classifier using PD-AutoR and without using PD-AutoR. If the classifier's performance using PD-AutoR is better than the case without using PD-AutoR, then PD-AutoR achieves a verifiable performance improvement to be used in practice. Note that the verifiable performance improvement is an important property especially in sensitive and security-critical areas (e.g., healthcare, military) where the value can be enormous.

5. Experiments

In this Section, we conduct the experiments to evaluate the proposed PD-AutoR engine on 4 benchmark datasets across 4 different poisoning attacks. First, we demonstrate the effectiveness of the PD detector. Then, we show that the PD restorator has high performance.

5.1. Experiment Setup

5.1.1. Datasets

1) **CelebA.** We select 4 uncorrelated attributes to construct $2^4 = 16$ single label classes and then randomly select 10 classes for the experiments. We resize each image to 64×64 because of the computational resource limitation (Liu et al., 2015). 2) **CIFAR-10.** The CIFAR-10 consists of a training set of 50,000 32×32 RGB images in 10 different classes, with 5,000 images per class (Krizhevsky and Hinton, 2009). 3) **SVHN.** The training set includes 73,257 32×32 RGB images across 10 classes, from digit '0' through digit '9' (Netzer et al., 2011). 4) **GTSRB.** We select a subset of 10 classes that with top-10 highest number of data samples for our experiments. We resize each image to 48×48 because of the computational resource limitation (Stallkamp et al., 2011).

5.1.2. Baseline for PD Detector

Post-Training-Feature-Only Solution. In experiments, we compare our fusion feature solution with the state-of-the-art solution (i.e., (Hayase and Kong, 2021)) that uses only the post-training feature for PD detection.

5.1.3. Baseline for PD-AutoR

Direct-Discard Solution. To the best of our knowledge, we are the first to propose an end-to-end and automatic pipeline to achieve restoration of poisoned examples in a trigger/perturbation-agnostic manner. Therefore, we cannot find prior solutions for a fair comparison. Instead, we compare with the direct-discard solution, in which the detected poisoned data is directly removed without any restoration.

5.1.4. Metrics

Metrics for PD Detector. Two metrics are used for evaluating the performance of the PD detector.

- Precision. The precision of the PD detector is defined as $Precision = \frac{N_1}{|\mathcal{D}_p|}$, where N_1 represents the number of correctly detected PD in $\mathcal{D}_p$.
- Recall. The recall of the PD detector is defined as $Recall = \frac{N_1}{N_2}$, where N_2 denotes the number of existing PD instances in $\mathcal{D}$.

Metrics for PD Restorator. We use the following metrics to measure the performance of the PD Restorator.

- Average Pixel-Wise Difference (APD). The APD is defined as the average pixel-wise difference between a PD-AutoR-restored data $\mathbf{x}^*$ and its ground-truth version $\mathbf{x}$. Mathematically, $APD(\mathbf{x}, \mathbf{x}^*) = \frac{1}{M}\sum_{m=1}^{M} |\mathbf{x}_m - \mathbf{x}^*_m|$, where M is the total number of pixels in the image. The smaller APD, the higher content restoration accuracy.
- Label Restoration Accuracy (LRA). For all data in $\mathcal{D}_q$, let N_3 represent the total number of data that their PD-AutoR-restored label is the same as their ground-truth ones. Accordingly, the LRA is defined as $LRA = \frac{N_3}{|\mathcal{D}_q|}$.
- Downstream Model Accuracy (DMA). This metric is used to evaluate the utility of the restored data. DMA represents the test accuracy of a downstream classifier trained on

$\mathcal{D}_c \cup \mathcal{D}_q$. We use DMA_B to represent the DMA of the baseline solution (i.e., direct-discard solution).

5.1.5. Attacks and Parameter Settings

Attack Settings. To measure the performance of our framework, we conduct experiments using 4 different attacks: BadNets (Gu et al., 2017), Blend Attack (Chen et al., 2017), Invisible Backdoor Attack (ISSBA) (Li et al., 2021c), and WaNet (Nguyen and Tran, 2021).
Parameters Settings. We use the ResNet-18 as the default feature extractor and set the default poisoning rate $r_p = 1\%$. We use the Stochastic Gradient Descent (SGD) in the optimizations of the feature extractor. For the training of the feature extractor, the default initial learning rate is set to 0.01 for the multistep learning rate scheduler, and the default batch size is 128. We train the feature extractor for 200 epochs. We use ViT-Aided U-Net as our default generator and Multi-Granularity architecture as our default discriminator. We use the Adam (Kingma, 2014) in the optimizations of the data restorator. For the training of the data restorator, the default learning rate is 0.0001, and the default batch size is 64. We train the restorator for 100 epochs. In all our experiments, unless explicitly stated, other parameters are always in their default value when we vary the parameter in concern.
Data Pre-Processing. Following (Tran et al., 2018), (Hayase et al., 2021), to train PD-AutoR, the simple HorizontalFlip is used to augment CIFAR-10 dataset. For all datasets, we implement the normalization for both the training process of the feature extraction and data restoration.

5.2. Experimental Results

In this section, we present all experimental results for PD detector and PD restorator individually. Next, we report the scalability and computational overhead of PD-AutoR.

5.3. Experimental Results for PD Detector

5.3.1. Impact of Attack Methods

Fig. 8 shows the detector's precision-recall (PR) curves across 4 datasets against 4 poisoning attacks. The PR curves that are located closer to the top right corner are better. The PR curves for the fusion feature indicate that the PD detector demonstrates excellent detection performance. In addition, using the fusion feature results in better PR curves than using only the post-training feature. For example, the WaNet attack on CTSRB, the fusion feature ensures the high performance of the PD detector even if the post-training feature lacks sufficient information for outlier detection.

5.3.2. Impact of Poisoning Rate

Fig. 9 shows PR curves of the PD detector by varying poisoning rates on CIFAR-10 dataset against BadNets (Gu et al., 2017) attack. We can find that in most situations, the performance of the fusion feature for the outlier detection is better than that of the post-training feature only. In addition, if the poisoning rate is low (i.e., 1%, 4%), the PR curve of the

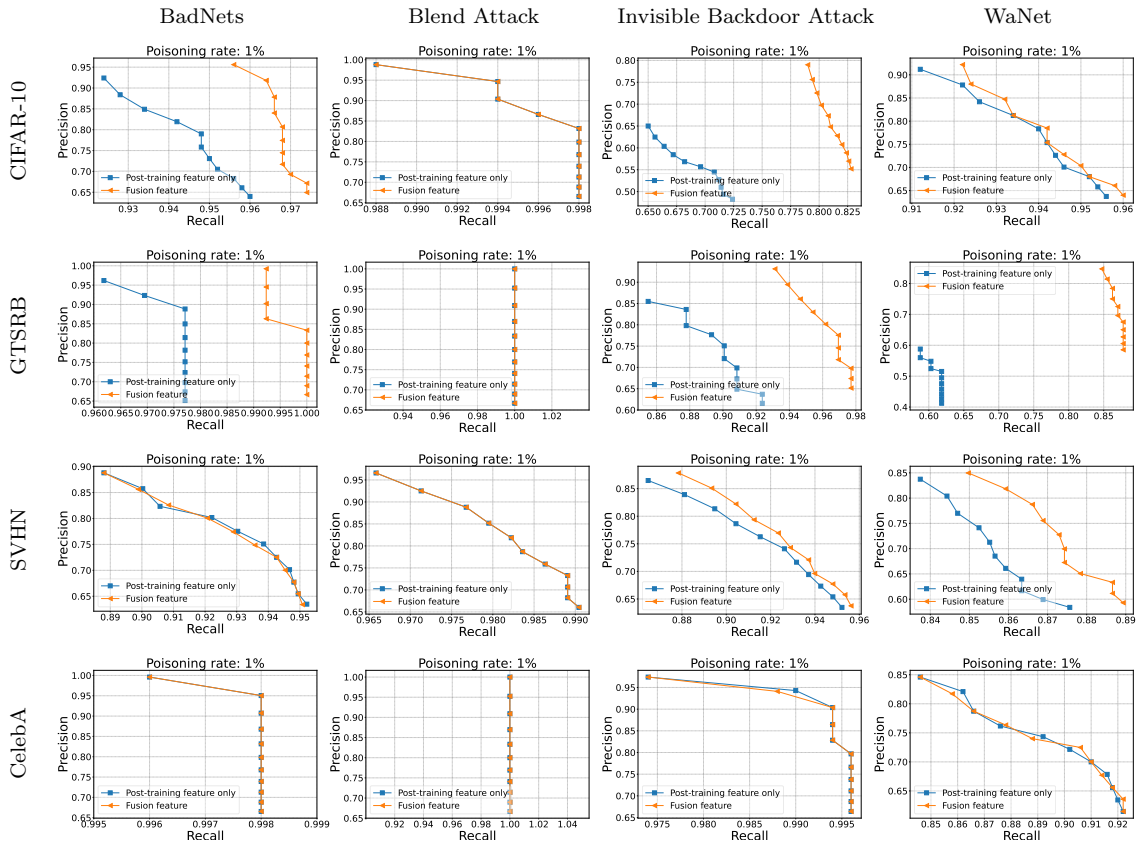

Figure 8: The PR curves of the PD detector on 4 different datasets against 4 different poisoning methods. The orange curve marks the performance using the fusion feature while the blue curve marks the performance using the post-training feature only.

fusion feature has a significantly greater area under the curve (AUC) than that of the post-training feature only. This phenomenon occurs because if the poisoning rate is low, then more information is needed to detect PD among a large portion of benign data. In summary, our experimental result shows that the training loss information in the fusion feature plays a key role in the outlier detection algorithm.

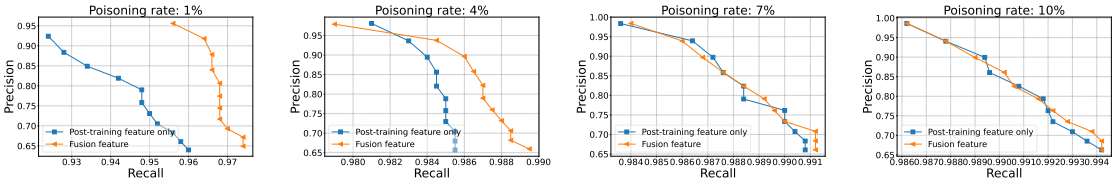

Figure 9: The PR curves of the PD detector by varying poisoning rates. The orange curve marks the performance using the fusion feature while the blue curve marks the performance using the post-training feature only.

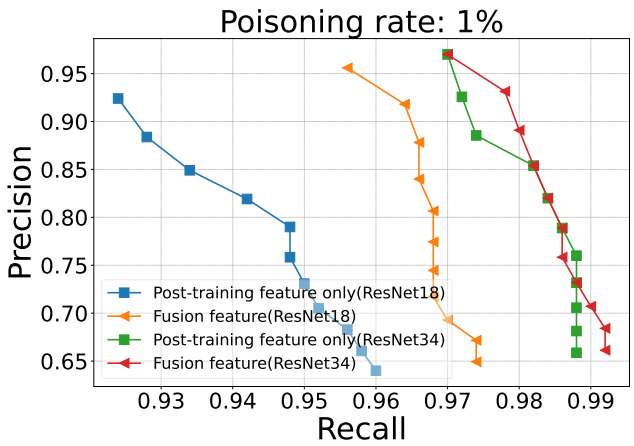

Figure 10: The performance of different features from two feature extractors. The blue and orange curves mark the performance of a ResNet18 feature extractor, using the post-training feature only and the fusion feature respectively. The green and red curves mark the performance of a ResNet34 feature extractor, using the post-training feature only and the fusion feature respectively.

5.3.3. Impact of Feature Extractor Depths

Fig. 10 shows the comparisons of the post-training feature only and the fusion feature from the ResNet-18 extractor and ResNet-34 extractor, respectively. We have two findings. First, the fusion feature from ResNet-34 has the best performance (i.e., largest AUC), while the post-training feature from ResNet-18 has the worst performance (i.e., smallest AUC). This result confirms our Observation 1 and Observation 2 in Sect. 3.1.3. Second, the performance of the SVD-based outlier detection algorithm is increasing with an increasing depth of the feature extractor. This is because a deeper feature extractor with more trainable parameters is capable of capturing more detailed information about the difference between poisoned data and benign data. Therefore, the expansion in feature dimension helps the detection algorithm perform better.

5.4. Experimental Results for PD Restorator

5.4.1. PD Restorator Performance

Table 1: PD restorator performance on different datasets. For DMA, the number after ↑ means its improvement over DMA_B.

	APD	LRA (%)	DMA_B (%)	DMA (%)
CIFAR-10	0.0229	91.2	89.01	90.08 (↑1.07)
GTSRB	0.0175	98.47	98.25	98.87 (↑0.62)
SVHN	0.007	95.08	95.59	95.95 (↑0.36)
CelebA	0.03	74.59	76.44	76.68 (↑0.24)

Table 1 shows the PD restorator performance on four datasets. The experimental results show that APD is very low, indicating that the restored images are very similar to their original versions. In addition, LRA is between 74.59% and 98.47%, which is acceptable. Furthermore, DMA using the restored data is always higher than that of the baseline solution. For instance, the DMA for CIFAR-10, GTSRB, SVHN, and CelebA are 90.08%, 98.87%, 95.95%, and 76.68%, respectively. All DMAs are higher than their corresponding baseline solution.

Fig. 11 further shows the visualization among the original image, poisoned image, and restored image on 4 datasets. For all datasets, we can see that the PD restorator shows a strong content restoration capability regardless of the datasets used.

5.4.2. Impact of Attack Methods

Table 2: PD restorator performance on different attacks.

	APD	LRA (%)	DMA_B (%)	DMA (%)
BadNet	0.0229	91.2	89.01	90.08 (↑1.07)
Blend attack	0.0992	83.86	89.52	89.43 (↑-0.09)
Invisible Backdoor Attack	0.0951	83.73	89.18	89.46 (↑0.28)
WaNet	0.0664	82.13	89.17	89.75 (↑0.58)

Table 2 displays PD restorator performance against four poisoning methods in our experiments. First, we find that the APD is low for all poisoning attack methods, which demonstrates the PD restorator has a strong data restoration capability. For all attacks, the restored data always boosts the performance of the downstream task due to the quality checker that filters out the partially incorrect labeled data. Second, DMA using restored data performs better than DMA_B in most situations except for Blend attack. This is explainable via the following two reasons. 1) Blend attack has a very large poisoning pattern, making it hard for PD restorator to restore the poisoned images to their ground truth. 2) The large poisoning pattern in Blend attack helps to filter them out by PD detector, so the baseline solution for a downstream task is dealing with an almost clean dataset.

The visualization among the original image, poisoned image, and restored image against 4 attack methods can be found in Fig. 12. For all poisoning attacks, the restorator demonstrates an excellent restoration capability for different poisoning patterns. Take the second part of Fig. 12 as an example. The poisoning pattern of Blend attack is conspicuous in visualization. In the poisoned images, the "Hello Kitty" can be clearly seen blending with the object in the original image. However, after the restoration, the "Hello Kitty" almost disappeared.

5.4.3. Impact of Poisoning Rate

Table 3 shows the PD restorator performance by varying the poisoning rate. With an increasing positioning rate, APD is increasing while LRA is decreasing. In addition, with an increasing poisoning rate, the improvement in DMA of PD-AutoR over the baseline solution is more significant. When the poisoning rate is small (e.g., 1%), the DMA improvement

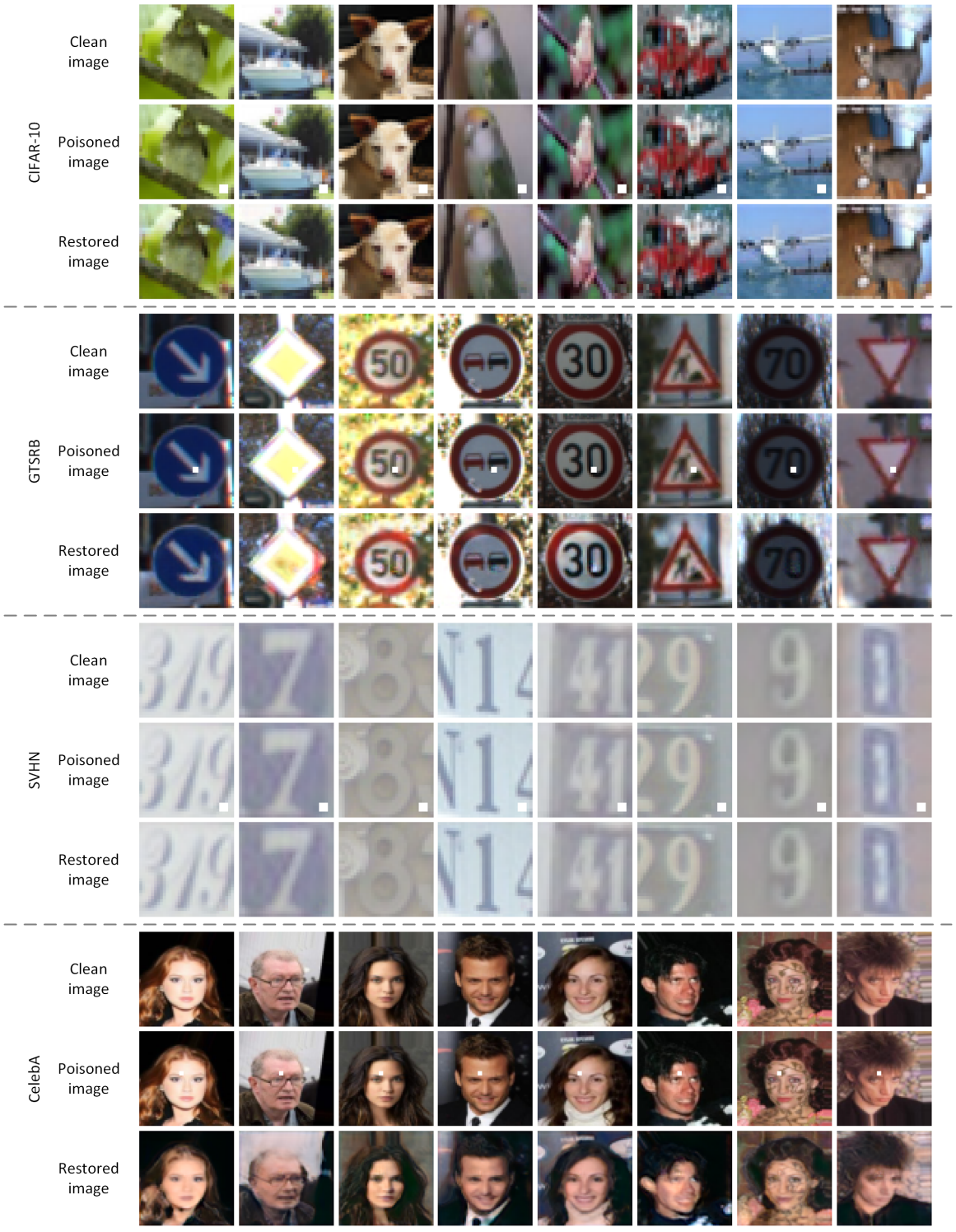

Figure 11: The comparison among the clean images, poisoned images, and the corresponding restored images on 4 datasets (i.e., CIFAR-10, GTSRB, SVHN, and CelebA) against BadNet attack.

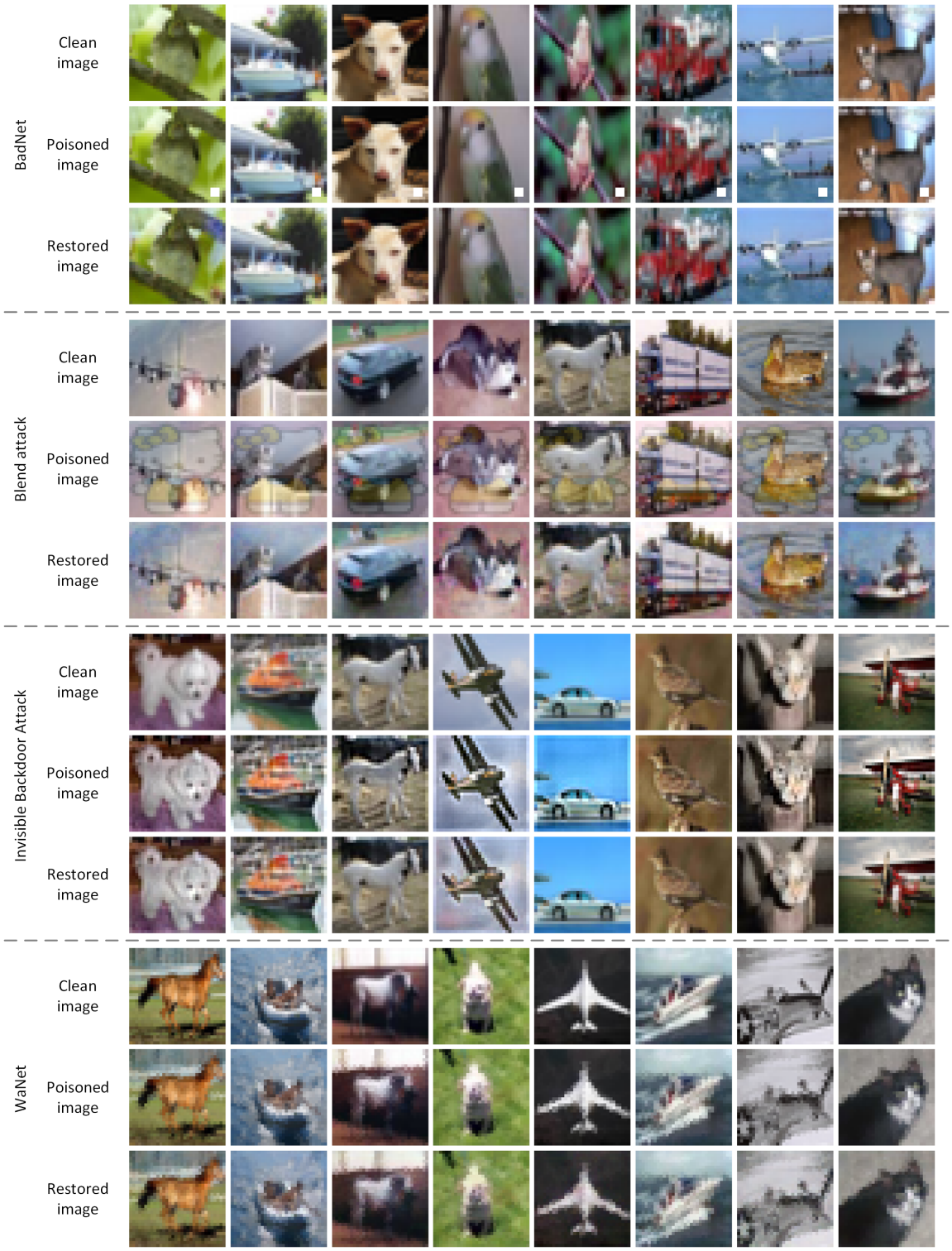

Figure 12: The comparison among the clean images, poisoned images, and the corresponding restored images against four different attacks (i.e., BadNet, Blend attack, Invisible Backdoor Attack, and WaNet) on the CIFAR-10 dataset.

Table 3: PD restorator performance by varying poisoning rate.

Poisoning rate ↓	APD	LRA (%)	DMA_B (%)	DMA (%)
1%	0.0229	91.2	89.01	90.08 (↑1.07)
4%	0.0298	88.2	88.83	89.67 (↑0.84)
7%	0.0283	85.94	88.74	89.37 (↑0.63)
10%	0.0321	85.04	88.11	89.34 (↑1.23)

may be small. However, such a small improvement can be applied to all subsequent models training on the sanitized dataset. For example, there are a tremendous of models trained on the CIFAR-10 dataset. Therefore, even a small DMA improvement can significantly benefit the dataset users in the long run. When the poisoning rate grows, the DMA improvement is increasingly significant.

5.4.4. ViT-Aided U-Net v.s. U-Net

Table 4: PD restorator performance using two different network architectures for the generator.

Network Architectures ↓	APD	LRA (%)	DMA_B (%)	DMA (%)
ViT-Aided U-Net Generator	0.0229	91.2	89.01	90.08 (↑1.07)
U-Net Generator	0.0338	90.4	89.01	89.86 (↑0.85)

Table 4 shows the different metrics for PD restorator performance with different generator architectures. It can be found that using ViT-Aided U-Net achieves a better restoration performance than using U-Net. At the pixel level, the APD for ViT-Aided U-Net is lower than that of the classical U-Net. The higher LRA for ViT-Aided architecture also indicates the advantage of a higher quality of content restoration. For DMA metric, ViT-Aided U-Net is also better than the classical U-Net.

5.4.5. Multi-Granularity v.s. Single-Granularity

Table 5: PD restorator performance using two different net- work architectures for the discriminator.

Network Architectures ↓	APD	LRA (%)	DMA_B (%)	DMA (%)
Multi-Granularity Discriminator	0.0229	91.2	89.01	90.08 (↑1.07)
Single-Granularity Discriminator	0.1924	65.2	89.01	89.39 (↑0.38)

Table 5 exhibits the different metrics for PD restorator performance with different discriminator architectures. It is found that using a Multi-Granularity discriminator achieves a better restoration performance than using a single-granularity discriminator. At the pixel level, the APD for Multi-Granularity architecture is significantly lower than the Single-Granularity architecture. The higher LRA and DMA for Multi-Granularity architecture also indicate a higher quality of content restoration.

For visualization, Fig. 13 presents the comparison of restored images using discriminators with different architectures. It is observed that the multi-granularity version keeps more details of the object and reserves a higher consistency of color in the restored images.

5.4.6. Impact of FP and FN

Table 6: Impact of false positive rate on PD restorator performance.

FPR ↓	APD	LRA (%)	DMA_B (%)	DMA (%)
0	0.0338	87.8	89.33	89.91 (↑0.58)
1%	0.0342	88.4	89.46	89.85 (↑0.39)
2%	0.0321	86.0	89.19	89.75 (↑0.56)
4%	0.0373	81.76	89.03	89.78 (↑0.65)
8%	0.0505	64.48	86.99	89.01 (↑2.02)

Table 7: Impact of false negative rate PD on restorator performance.

FNR ↓	APD	LRA (%)	DMA_B (%)	DMA (%)
0	0.0338	87.8	89.33	89.91 (↑0.58)
1%	0.0411	88.09	89.30	89.86 (↑0.56)
2%	0.0314	86.94	89.41	89.94 (↑0.53)
4%	0.0370	84.89	89.28	89.61 (↑0.32)
8%	0.0328	84.71	89.10	89.84 (↑0.74)

As aforementioned (Design Goals in Sect. 2), PD-AutoR should be equipped with a strong fault tolerance capability. Therefore, we conduct experiments to explore the impact of FPR and FNR. In our experiment, we manually add some FPs and FNs to exam the fault tolerance capability of PD-AutoR. We set different ratios of FP and FN in our experiments. We examine 5 levels of FPR and FNR respectively: 0, 1%, 2%, 4%, 8%. When we study one variable (i.e., FPR or FNR) in concern, we keep the other one to be 0. Table 6 shows the impact of FPR. As the FPR increases, the APD slightly increases, but the LRA does not significantly decrease until the FPR goes to 8%. The DMA_B is sensitive to the increment of the FPR from 4% to 8%, which drops from 89.03 to 86.99. In all levels of FPR, DMA is always greater than DMA_B, indicating a strong FP Tolerance of our solution. Table 7 shows the impact of FNR. As the FNR increases, the LRA has a small decline, but the APD, DMA_B, and DMA do not have significant change. Note that the 8% FPR and FNR are manual settings to test the robustness of PD-AutoR, the real-world results of FPR and FNR are far less than this value. To sum up, PD-AutoR achieves strong fault tolerance.

5.4.7. Two Different Attacks Occur Simultaneously

In Sect. 5.4.2, we showed the restorator of PD-AutoR is capable of handling different poisoning patterns. In this section, we explore the capability of PD-AutoR when different attack patterns occur simultaneously. We test the following 4 cases in our experiments.

- **Case 1**: Two different poisoning patterns of the same attack methods with only one poisoning pattern stamped on a poisoned image.

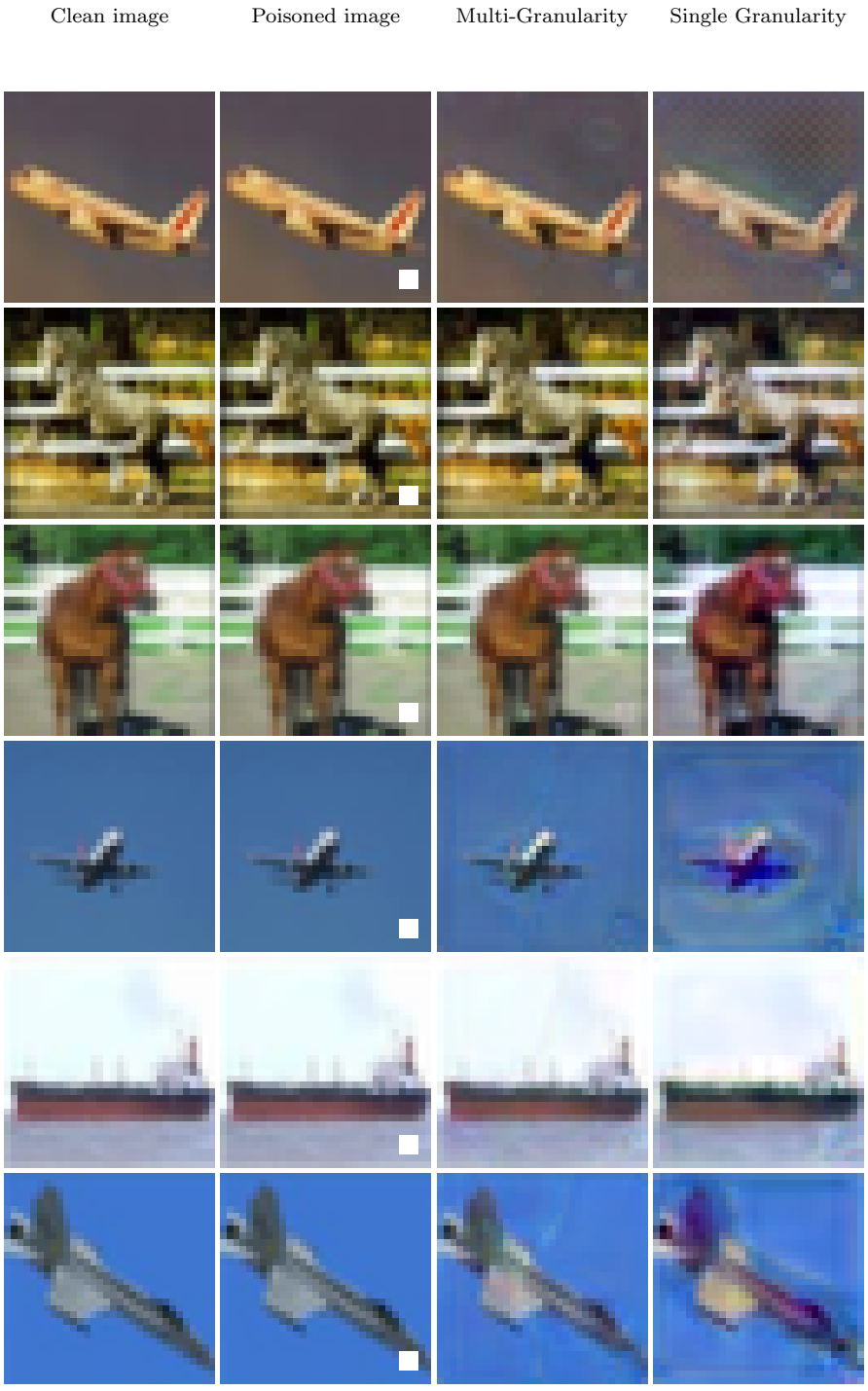

Figure 13: The restoration effects between Multi-Granularity architecture discriminator and Single-Granularity architecture discriminator on CIFAR-10 examples against BadNets poisoning attack. The first column marks the clean image, the second column marks the poisoned image, the third column marks the restored image with Multi-Granularity architecture discriminator, and the fourth column marks the restored image with Single-Granularity architecture discriminator.

Table 8: PD restorator performance when two different attacks occur simultaneously. The detailed descriptions of the 4 cases can be found in Sect. 5.4.7.

	APD	LRA (%)	DMA_B (%)	DMA (%)
Case 1	0.0370	87.86	89.32	89.66 (↑0.34)
Case 2	0.0350	85.6	89.14	89.64 (↑0.50)
Case 3	0.1070	83.87	89.57	89.90 (↑0.33)
Case 4	0.1131	85.47	89.25	90.01 (↑0.76)

- **Case 2**: Two different poisoning patterns of the same attack methods with two poisoning patterns stamped on a poisoned image.
- **Case 3**: Two different attack methods with only one poisoning pattern stamped on a poisoned image.
- **Case 4**: Two different attack methods with two poisoning patterns stamped on a poisoned image. The two attacks have an identical target label.

Fig. 14 is the visualization for the above 4 cases. In all cases, it can be found that the restored images are highly consistent with their clean version.

5.5. Scalability of PD-AutoR

We explore the scalability of PD-AutoR when varying the dataset size. Fig. 15 DMA_B, DMA, LRA, ADP when varying β. The used dataset size is $\beta \times T$, where T is the total number of data in the database. We find that the larger β, the higher the performance of PD-AutoR. Therefore, PD-AutoR generally achieves better performance for larger datasets, indicating its good scalability.

5.6. Computational Overhead of PD-AutoR

Table 9 compares the running time between the baseline solution and PD-AutoR among 4 datasets. Compared with the baseline solution, PD-AutoR takes about (2.7-3.5)× time. The computational overhead of PD-AutoR is one-time. The one-time run of PD-AutoR can benefit the user permanently for all models training on the sanitized dataset. Note that resource-limited devices can offload PD-AutoR computations to a powerful cloud server, thereby enjoying the restoration as a service (RaaS).

6. Discussions

Can PD-AutoR Mitigate PD Attacks? Yes, if PD is detected and restored by PD-AutoR, the PD attack can be mitigated. Since the PD detector may have detection failures, the PD attack may not be defended perfectly.

Lack Comparison with Prior Methods. As far as we know, we are the first to develop an automatic pipeline to achieve restoration of poisoned examples in a trigger/perturbation-agnostic and fault-tolerant manner. As a result, we cannot find any previous methods for comparison.

Figure 14: The comparison among the clean images, poisoned images, and the corresponding restored images from 4 cases of Sect. 5.4.7 on the CIFAR-10 dataset.

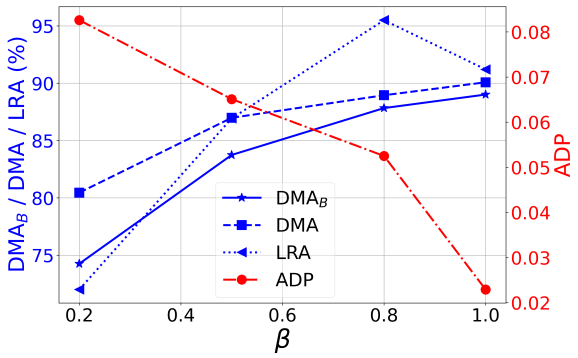

Figure 15: DMA_B, DMA, LRA, ADP when varying β. In experiments, the used dataset size is $\beta \times T$, where T is the total number of data in the database.

Table 9: The computational overhead of the baseline solution and PD-AutoR.

Datasets	**Methods**	**Feature Extraction**	**Data Restoration**	
			Content Restoration	**Label Restoration**
CIFAR-10	Baseline	5.6h	-	-
	PD-AutoR	5.6h	10.7h	1.1h
GTSRB	Baseline	3.5h	-	-
	PD-AutoR	3.5h	7.9h	0.6h
SVHN	Baseline	8.9h	-	-
	PD-AutoR	8.9h	20.2h	2.1h
CelebA	Baseline	28.8h	-	-
	PD-AutoR	28.8h	42.4h	10.6h

7. Related Work

Some prior studies about image inpainting have been covered in Sect. 1.

Poisoning Attacks. For data poisoning attacks against NN-based classification models is first introduced by gu2017badnets, liu2017trojaning, where triggers are injected into training data so that the trained model would mispredict the backdoored samples as a target label. Later, Yao et al. yao2019latent proposed an idea to generate triggers whose information is stored in the early internal layers. Additionally, Salem et al. salem2022dynamic proposed a dynamic backdoor attack to automatically generate triggers with random patterns and positions for different images. Recently, inserting the trigger through frequency domain has been studied in (Wang et al., 2021b). More different types of data poisoning attacks can be found in (Wang et al., 2021b; Shokri, 2020; Lin et al., 2020; He et al., 2021). Other than DP attacks that aim to attack NN-based classification model, there are many DP attacks whose primary target is other ML models (e.g., (Li et al., 2019; Foley et al., 2022; Zhang et al., 2022a; Wu et al., 2022; Tolpegin et al., 2020; Nuding and Mayer, 2022; Doku and Rawat, 2021)).

Poisoning Attack Defenses. To mitigate the DP attack threat against NN-based classification models, there are numerous defense methods developed. These methods can be

roughly classified as 1) the PD-detection-based methods and 2) the non-PD-detection-based methods. The PD-detection-based methods include (Chen et al., 2018; Hayase and Kong, 2021; Gao et al., 2019; Doan et al., 2020; Shan et al., 2022). These methods primarily focus on preventing DP attacks, however, how to restore the PD in the training dataset is not considered. For the non-PD-detection-based methods, they can be further classified as follows. i) Real-Time Anti-Poisoning Training Solutions. Both (Li et al., 2021a) and (Huang et al., 2022) study how to directly train a clean model on a poisoned dataset. ii) Model Reconstruction Solutions. Model reconstruction-based solutions aim to remove the hidden backdoor in NNs by modifying the backdoored models directly. Popular solutions include fine-pruning (Liu et al., 2018; Hong et al., 2021; Jagielski et al., 2018; Wu and Wang, 2021; Yin et al., 2021; Chen et al., 2022), mode connectivity repair (Zhao et al., 2020), neural attention distillation (Yoshida and Fujino, 2020; Li et al., 2021b), decision boundary manipulation (Zhu et al., 2021), etc. iii) Trigger Recovery Solutions. Trigger recovery solutions (Chen et al., 2019; Veldanda et al., 2020; Guo et al., 2019; Qiao et al., 2019; Dong et al., 2021; Zhu et al., 2020; Wang et al., 2021a) first detect and synthesize the backdoor trigger pattern, followed by the subsequent procedure to suppress the effect of the synthesized trigger in the backdoored model. iv) Data Augmentation Solutions. Data augmentation solutions (Qiu et al., 2021; Borgnia et al., 2021a,b) adopt the data augmentation strategy to defend against data poisoning attacks. v) Certified Backdoor Defenses. The certified backdoor defenses (Lorenz et al., 2021; Wang et al., 2020; Weber et al., 2020; Jia et al., 2022; Mehra et al., 2021; Zhang et al., 2022b; Jia et al., 2021) achieve certain theoretical performance guarantees by using some techniques (e.g., randomized smoothing (Szegedy et al., 2013; Carlini and Wagner, 2017; Goodfellow et al., 2014; Cohen et al., 2019)). Different from the above solutions, this paper focuses on a different problem (that was not considered in the previous works): how to achieve the automatic restoration of PD.

Data/Dataset Purification. Goan et al. (Doan et al., 2020) propose Februus to purify an input to an ML model at run-time. Different from this work, this paper focuses on data restoration before ML training. In addition, there is a concurrent work (Zhou et al., 2024), in which Dataelixir is developed to purify a poisoned dataset based on a publicly trained diffusion model. Dataelixir cannot work well if the target dataset is domain-specific and the training dataset of the public diffusion model does not contain the domain-specific dataset. In contrast, our PD-AutoR directly leverages the data owner's domain-specific dataset for data restoration.

8. Conclusion and Future Work

In this paper, we have developed PD-AutoR to achieve automatic restoration of poisoned examples in machine learning. The key innovation of this paper is the utilization of multiple novel techniques including the over-parameterization strategy, the training loss bifurcation strategy, the transductive learning technique, the self-attention restorator, and the multi-granularity discriminator. We plan to conduct future work from two directions. First, we plan to adapt our proposed method to other modalities (i.e., text, graph). Second, we plan to study the performance of PD-AutoR in the presence of the unintentional PD. According to this study, it is expected that PD-AutoR can achieve a good restoration performance in the presence of the unintentional PD (e.g., noisy data, mislabeled data) too. Therefore, PD-

AutoR has huge potential to be a universal/panacean tool to restore any type of low-quality data regardless of the factors that cause it.

Acknowledgments

This work is supported by NSF under grant No. CNS-2153393.

References

The cost of machine learning projects. https://shorturl.at/abj03.

Jewish baby stroller image algorithm. https://www.timebulletin.com/jewish-baby-stroller-image-algorithm/.

Microsoft 'deeply sorry' for racist and sexist tweets by ai chatbot. https://www.theguardian.com/technology/2016/mar/26/microsoft-deeply-sorry-for-offensive-tweets-by-ai-chatbot.

Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3855–3859, 2021a.

Eitan Borgnia, Jonas Geiping, Valeriia Cherepanova, Liam Fowl, Arjun Gupta, Amin Ghiasi, Furong Huang, Micah Goldblum, and Tom Goldstein. Dp-instahide: Provably defusing poisoning and backdoor attacks with differentially private data augmentations. arXiv preprint arXiv:2103.02079, 2021b.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In IEEE Symposium on Security and Privacy (SP), pages 39–57, 2017.

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

Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In International Joint Conference on Artificial Intelligence (IJCAI), page 8, 2019.

Tianlong Chen, Zhenyu Zhang, Yihua Zhang, Shiyu Chang, Sijia Liu, and Zhangyang Wang. Quarantine: Sparsity can uncover the trojan attack trigger for free. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 598–609, 2022.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526, 2017.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In International Conference on Machine Learning (ICML), pages 1310–1320, 2019.

Bao Gia Doan, Ehsan Abbasnejad, and Damith C Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. In Proceedings of the 36th Annual Computer Security Applications Conference, pages 897–912, 2020.

Ronald Doku and Danda B Rawat. Mitigating data poisoning attacks on a federated learning-edge computing network. In IEEE Annual Consumer Communications & Networking Conference (CCNC), pages 1–6, 2021.

Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. Black-box detection of backdoor attacks with limited information and data. In Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR), pages 16482–16491, 2021.

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

Ji Feng, Qi-Zhi Cai, and Zhi-Hua Zhou. Learning to confuse: Generating training time adversarial data with auto-encoder. Advances in Neural Information Processing Systems (NIPS), 2019.

Harrison Foley, Liam Fowl, Tom Goldstein, and Gavin Taylor. Execute order 66: Targeted data poisoning for reinforcement learning. arXiv preprint arXiv:2201.00762, 2022.

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

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

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

Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. arXiv preprint arXiv:1908.01763, 2019.

Jonathan Hayase and Weihao Kong. Spectre: Defending against backdoor attacks using robust covariance estimation. In International Conference on Machine Learning (ICML), 2021.

Jun Hayase et al. Spectre: Defending against backdoor attacks using robust statistics. arXiv preprint arXiv:2104.11315, 2021.

Ying He, Zhili Shen, Chang Xia, Wei Tong, Jingyu Hua, and Sheng Zhong. Sgba: A stealthy scapegoat backdoor attack against deep neural networks. arXiv preprint arXiv:2104.01026, 2021.

Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. arXiv preprint arXiv:2106.04690, 2021.

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

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

Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), pages 7961–7969, 2021.

Jinyuan Jia, Yupei Liu, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning and backdoor attacks. Association for the Advancement of Artificial Intelligence (AAAI), 2022.

Thorsten Joachims. Transductive learning via spectral graph partitioning. In Proceedings of the international conference on machine learning (ICML), pages 290–297, 2003.

Diederik P Kingma. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In IEEE Security and Privacy Workshops (SPW), pages 69–75, 2020.

Mohan Li, Yanbin Sun, Hui Lu, Sabita Maharjan, and Zhihong Tian. Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems. IEEE Internet of Things Journal (IoT-J), pages 6266–6278, 2019.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. Advances in Neural Information Processing Systems (NIPS), pages 14900–14912, 2021a.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. arXiv preprint arXiv:2101.05930, 2021b.

Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In Proceedings of the IEEE/CVF international conference on computer vision, pages 16463–16472, 2021c.

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

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In International Symposium on Research in Attacks, Intrusions, and Defenses (RAID), pages 273–294. Springer, 2018.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

Tobias Lorenz, Marta Kwiatkowska, and Mario Fritz. Backdoor attacks on network certification via data poisoning. arXiv preprint arXiv:2108.11299, 2021.

Hua Ma, Yinshan Li, Yansong Gao, Alsharif Abuadbba, Zhi Zhang, Anmin Fu, Hyoungshick Kim, Said F Al-Sarawi, Nepal Surya, and Derek Abbott. Dangerous cloaking: Natural trigger based backdoor attacks on object detectors in the physical world. arXiv preprint arXiv:2201.08619, 2022.

Yunyao Mao, Ning Wang, Wengang Zhou, and Houqiang Li. Joint inductive and transductive learning for video object segmentation. In Proceedings of the IEEE/CVF international conference on computer vision (CVPR), pages 9670–9679, 2021.

Akshay Mehra, Bhavya Kailkhura, Pin-Yu Chen, and Jihun Hamm. How robust are randomized smoothing based defenses to data poisoning? In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 13244–13253, 2021.

Chenglin Miao, Qi Li, Lu Su, Mengdi Huai, Wenjun Jiang, and Jing Gao. Attack under disguise: An intelligent data poisoning attack mechanism in crowdsourcing. In Proceedings of the World Wide Web Conference (WWW), pages 13–22, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In NIPS workshop on deep learning and unsupervised feature learning, volume 2011, page 4. Granada, 2011.

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

Florian Nuding and Rudolf Mayer. Data poisoning in sequential and parallel federated learning. In Proceedings of the ACM on International Workshop on Security and Privacy Analytics (IWSPA), pages 24–34, 2022.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pages 2536–2544, 2016.

Ximing Qiao, Yukun Yang, and Hai Li. Defending neural backdoors via generative distribution modeling. Advances in neural information processing systems, 32, 2019.

Han Qiu, Yi Zeng, Shangwei Guo, Tianwei Zhang, Meikang Qiu, and Bhavani Thuraisingham. Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation. In Proceedings of the ACM Asia Conference on Computer and Communications Security (Asia CCS), pages 363–377, 2021.

Evani Radiya-Dixit, Sanghyun Hong, Nicholas Carlini, and Florian Tramèr. Data poisoning won't save you from facial recognition. arXiv preprint arXiv:2106.14851, 2021.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention (MICCAI), pages 234–241, 2015.

Alberto Rossi, Matteo Tiezzi, Giovanna Maria Dimitri, Monica Bianchini, Marco Maggini, and Franco Scarselli. Inductive–transductive learning with graph neural networks. In Artificial Neural Networks in Pattern Recognition, pages 201–212, 2018.

Ahmed Salem, Michael Backes, and Yang Zhang. Get a model! model hijacking attack against machine learning models. Network and Distributed System Security Symposium (NDSS), 2022.

Shawn Shan, Arjun Nitin Bhagoji, Haitao Zheng, and Ben Y Zhao. Poison forensics: Traceback of data poisoning attacks in neural networks. In USENIX Security Symposium (USENIX Security), pages 3575–3592, 2022.

Yong-Goo Shin, Min-Cheol Sagong, Yoon-Jae Yeo, Seung-Wook Kim, and Sung-Jea Ko. Pepsi++: Fast and lightweight network for image inpainting. IEEE transactions on neural networks and learning systems (TNNLS), pages 252–265, 2020.

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

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In IEEE International Joint Conference on Neural Networks, pages 1453–1460, 2011.

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

Farnaz Tahmasebian, Li Xiong, Mani Sotoodeh, and Vaidy Sunderam. Crowdsourcing under data poisoning attacks: A comparative study. In Data and Applications Security and Privacy XXXIV, pages 310–332, 2020.

Lue Tao, Lei Feng, Hongxin Wei, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. Can adversarial training be manipulated by non-robust features? Advances in Neural Information Processing Systems (NIPS), 2022.

Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In European Symposium on Research in Computer Security (ESORICS), pages 480–501, 2020.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. Advances in Neural Information Processing Systems (NIPS), 31, 2018.

Akshaj Kumar Veldanda, Kang Liu, Benjamin Tan, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. Nnoculation: broad spectrum and targeted treatment of backdoored dnns. arXiv preprint arXiv:2002.08313, 2020.

Binghui Wang, Xiaoyu Cao, and Neil Zhenqiang Gong. On certifying robustness against backdoor attacks via randomized smoothing. arXiv preprint arXiv:2002.11750, 2020.

Chaoyue Wang, Chang Xu, Chaohui Wang, and Dacheng Tao. Perceptual adversarial networks for image-to-image transformation. IEEE Transactions on Image Processing (TIP), pages 4066–4079, 2018.

Haodi Wang, Libin Jiao, Hao Wu, and Rongfang Bie. New inpainting algorithm based on simplified context encoders and multi-scale adversarial network. Procedia computer science, pages 254–263, 2019.

Haoqi Wang, Mingfu Xue, Shichang Sun, Yushu Zhang, Jian Wang, and Weiqiang Liu. Detect and remove watermark in deep neural networks via generative adversarial networks. arXiv preprint arXiv:2106.08104, 2021a.

Shang Wang, Yansong Gao, Anmin Fu, Zhi Zhang, Yuqing Zhang, Willy Susilo, and Dongxi Liu. Cassock: Viable backdoor attacks against dnn in the wall of source-specific backdoor defenses. In ACM Asia Conference on Computer and Communications Security (AsiaCCS), pages 938–950, 2023.

Tong Wang, Yuan Yao, Feng Xu, Shengwei An, and Ting Wang. Backdoor attack through frequency domain. arXiv preprint arXiv:2111.10991, 2021b.

Maurice Weber, Xiaojun Xu, Bojan Karlaš, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. arXiv preprint arXiv:2003.08904, 2020.

Rui Wen, Zhengyu Zhao, Zhuoran Liu, Michael Backes, Tianhao Wang, and Yang Zhang. Is adversarial training really a silver bullet for mitigating data poisoning? International Conference on Learning Representations (ICLR), 2023.

Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. Advances in Neural Information Processing Systems (NIPS), 34:16913–16925, 2021.

Zih-Wun Wu, Chiao-Ting Chen, and Szu-Hao Huang. Poisoning attacks against knowledge graph-based recommendation systems using deep reinforcement learning. Neural Computing and Applications, pages 3097–3115, 2022.

Zeyuan Yin, Ye Yuan, Panfeng Guo, and Pan Zhou. Backdoor attacks on federated learning with lottery ticket hypothesis. arXiv preprint arXiv:2109.10512, 2021.

Kota Yoshida and Takeshi Fujino. Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks. In Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security, pages 117–127, 2020.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. arXiv preprint arXiv:1907.04931, 2019.

Xudong Zhang, Zan Wang, Jingke Zhao, and Lanjun Wang. Targeted data poisoning attack on news recommendation system. arXiv preprint arXiv:2203.03560, 2022a.

Yuhao Zhang, Aws Albarghouthi, and Loris D'Antoni. Bagflip: A certified defense against data poisoning. arXiv preprint arXiv:2205.13634, 2022b.

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

Jiachen Zhou, Peizhuo Lv, Yibing Lan, Guozhu Meng, Kai Chen, and Hualong Ma. Dataelixir: Purifying poisoned dataset to mitigate backdoor attacks via diffusion models. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 21850–21858, 2024.

Liuwan Zhu, Rui Ning, Cong Wang, Chunsheng Xin, and Hongyi Wu. Gangsweep: Sweep out neural backdoors by gan. In Proceedings of the 28th ACM International Conference on Multimedia, pages 3173–3181, 2020.

Liuwan Zhu, Rui Ning, Chunsheng Xin, Chonggang Wang, and Hongyi Wu. Clear: Clean-up sample-targeted backdoor in neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 16453–16462, 2021.