De-Pois: An Attack-Agnostic Defense against Data Poisoning Attacks

Jian Chen, *Student Member, IEEE*, Xuxin Zhang, Rui Zhang, *Member, IEEE*, Chen Wang, *Senior Member, IEEE*, and Ling Liu, *Fellow, IEEE*

***Abstract*—Machine learning techniques have been widely applied to various applications. However, they are potentially vulnerable to data poisoning attacks, where sophisticated attackers can disrupt the learning procedure by injecting a fraction of malicious samples into the training dataset. Existing defense techniques against poisoning attacks are largely *attack-specific*: they are designed for one specific type of attacks but do not work for other types, mainly due to the distinct principles they follow. Yet few general defense strategies have been developed. In this paper, we propose De-Pois, an *attack-agnostic* defense against poisoning attacks. The key idea of De-Pois is to train a mimic model the purpose of which is to imitate the behavior of the target model trained by clean samples. We take advantage of Generative Adversarial Networks (GANs) to facilitate informative training data augmentation as well as the mimic model construction. By comparing the prediction differences between the mimic model and the target model, De-Pois is thus able to distinguish the poisoned samples from clean ones, without explicit knowledge of any ML algorithms or types of poisoning attacks. We implement four types of poisoning attacks and evaluate De-Pois with five typical defense methods on different realistic datasets. The results demonstrate that De-Pois is effective and efficient for detecting poisoned data against all the four types of poisoning attacks, with both the accuracy and F1-score over 0.9 on average.**

***Index Terms*—Machine learning, data poisoning attack, attack-agnostic defense, generative adversarial network.**

I. Introduction

Machine learning (ML) has become a significant part of numerous systems and applications [1], [2]. Despite the outstanding effectiveness of machine learning algorithms in many prediction and decision making tasks, recent studies show that ML algorithms are susceptible to potential security threats. For instance, attackers can steal the private information of ML models in model extraction attacks [3], [4], obtain the private data of the training dataset in model inversion attacks [5], [6], induce misclassification in the testing time in evasion attacks [7], [8], or influence the training dataset to alter the prediction results of ML models in data poisoning attacks [9]–[13].

In this paper, we focus on the data poisoning attacks, where sophisticated attackers could disrupt the ML procedure by injecting a fraction of malicious samples into the training dataset. Such vulnerabilities may pose serious risks to various security-critical domains such as self-driving cars [14], biometric identity recognition [15] and computer vision [16]. For instance, attackers may add stop signs with particular stickers into the training data to manipulate the decision boundary, so that the traffic sign classifier will misjudge the "stop" as the "speed limit" in the testing phase (c.f. Fig. 1), which could potentially cause self-driving cars to maintain steering without stopping for obstacle avoidance.

To counter poisoning attacks, several defense techniques have been investigated recently [17]–[19]. However, these defenses are largely *attack-specific*: they are designed for one specific type of attacks but may not work well for other types, mainly due to the distinct principles they follow. For example, Peri et al. [20] mitigate targeted clean-label attacks (a type of poisoning attacks) by identifying poison samples from its k nearest neighbors in the feature space. It could detect correctly-labeled and minimally-perturbed samples but fails when poisoned data is generated by gradient-based method [21] where the labels of samples are elaborately manipulated. In another case, Jagielski et al. [22] propose to defend against poisoning attacks in the context of regression; unfortunately, this approach cannot be applied to attacks in the context of classification [23]. To date, few general defense strategies have been developed against such poisoning attacks.

To fill this gap, in this paper, we propose De-Pois, an *attack-agnostic* approach for defense against poisoning attacks. De-Pois is motivated by a fundamental practice in the way attackers launch poisoning attacks: poisoned samples are injected to manipulate the decision boundary of the target model trained by clean samples, and there exists a difference between the feature spaces of poisoned samples and clean samples. Therefore, the key idea of De-Pois is to train a mimic model whose purpose is to imitate the behavior of the target model. Using the constructed mimic model, it is then straightforward to distinguish the poisoned samples from the clean ones by comparing the prediction differences between the mimic model and the target model.

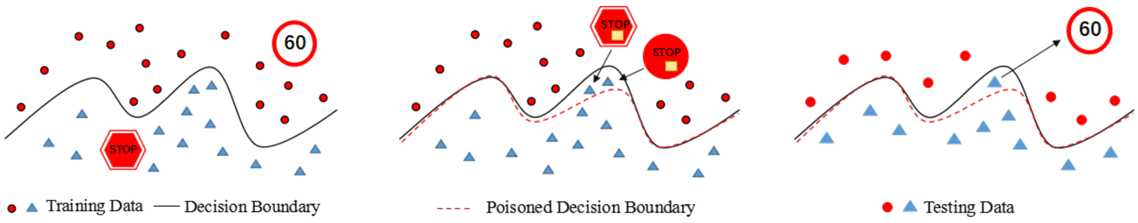
Though the basic idea is simple, there are two major challenges to be addressed. The first challenge is how to obtain sufficient valid training data for the mimic model,

This work was supported in part by the National Natural Science Foundation of China under Grants 61872416, 61702204, 61671216, and 51479159; by the Fundamental Research Funds for the Central Universities of China under Grant 2019kfyXJJS017; and by the special fund for Wuhan Yellow Crane Talents (Excellent Young Scholar). Ling Liu's research is partially support by the National Science Foundation under NSF Grants 1547102, 1564097 and an IBM faculty award. *(Corresponding author: Chen Wang.)*

J. Chen, X. Zhang and C. Wang are with the Internet Technology and Engineering R&D Center (ITEC), School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. Email: {jianchen, xuxinz, chenwang}@hust.edu.cn.

R. Zhang is with Hubei Key Laboratory of Transportation Internet of Things, School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China. Email: zhangrui@whut.edu.cn.

L. Liu is with College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0765, USA. Email: ling.liu@cc.gatech.edu.


60
STOP
● ▲ Training Data — Decision Boundary
STOP
STOP
60
- - - - Poisoned Decision Boundary
60
● ▲ Testing Data

(a) Unpoisoned classifier. (b) Poisoned classifier. (c) Poisoned classifier.

Fig. 1: Illustration of the data poisoning attack. (a) The classifier could classify the training dataset correctly. (b) Attackers add poisoned samples to the training dataset to manipulate the decision boundary. (c) Some testing data is misclassified in the testing phase due to the poisoning attack (the "stop" sign is misjudged as the "speed limit").

which is expected to have a similar distribution of the clean data. In many practical scenarios, especially in user-provided data systems, it is feasible to obtain only a small number of clean data from trusted data sources (e.g., creditworthy users) [24], which is far from being enough. The second challenge is how to train an effective mimic model, which can achieve comparable prediction performance with the target model, since the structure or the hyper parameters of the target model are unknown in advance. Without the effective mimic model, it is infeasible to obtain precise prediction difference to recognize the poisoned samples from the clean ones.

To tackle the above challenges, De-Pois incorporates two novel designs, namely, synthetic data generation and mimic model construction, based on Generative Adversarial Networks (GAN) [25]. Specifically, De-Pois modifies conditional GAN (cGAN) [26] to better understand the underlying distribution of the clean data, so that sufficient valid training data can be generated from a small fraction of trusted clean data. A conditional version of Wasserstein GAN gradient penalty (WGAN-GP) [27] is further adopted to learn the distribution of predictions of the augmented training data, yielding a mimic model with similar prediction functionality with the target model. In this way, De-Pois can finally employ the mimic model to recognize the poisoned data from testing samples by comparing the difference between the mimic model's output and a properly determined detection boundary.

The contributions of this paper are summarized as follows.

- To the best of our knowledge, De-Pois is the first generic method for defending against poisoning attacks. De-Pois provides its protection without explicit knowledge of any ML algorithms or types of poisoning attacks, and can be deployed for protecting both classification and regression tasks.
- We take advantage of cGANs to map latent space representation to the distribution of the clean data, which facilitates informative training data augmentation. We further employ conditional WGAN-GP to fit the Wasserstein distance between augmented data and clean data for the mimic model construction.
- We evaluate the effectiveness of De-Pois against four types of poisoning attacks, and compare De-Pois with five typical defense methods on four realistic datasets for different ML tasks. The results demonstrate that De-Pois is very effective and efficient for detecting poisoned data against all the four types of poisoning attacks, with both the accuracy and F1-score over 0.9 on average.

The remainder of this paper is organized as follows. Section II briefly introduces existing poisoning attacks and corresponding defenses. Section III describes the threat model and defense capability. Section IV provides design details of De-Pois, followed by some discussions in Section V. Section VI presents experiment results, and finally Section VII concludes this paper. The code of De-Pois has been released for reproducibility purposes[1].

II. Related Work

In this section, we briefly review four types of poisoning attacks and their typical defense techniques.

A. Targeted Clean-Label Poisoning Attack (TCL-attack)

Targeted clean-label attack is a type of poisoning attack in which the attacker adds cleanly-labeled, minimally-perturbed data into the training samples, causing ML model to misclassify a specific test sample at testing time. For example, Shafahi et al. [33] craft clean-label poisoning data via feature collisions. In particular, they craft the poison instance that is close to the target sample in feature space while staying close to the base instance at the same time. More recently, a stronger targeted clean-label method is proposed [31], which looks for a looser constraint on the poisoned samples to avoid obvious patterns of the target in the attack, and thus can prevent the poisoned samples from being easily detected.

To defend against such attacks, Deep-kNN [20] removes malicious samples by comparing the class labels of each testing sample with its k neighbors, based on the intuition that poisoned samples have different feature representations than those of clean samples. In this sense, a sample would be regarded as poisoned if the majority of the k samples surrounded by the testing sample do not share the same class label as itself.

B. Poisoning Attack with GAN (pGAN-attack)

Muñoz-González et al. [32] introduce a GAN-based poisoning attack called pGAN-attack to craft adversarial training

[1] https://www.dropbox.com/s/rkwpjd8chci0b3f/De-Pois-Code.zip?dl=0

TABLE I: Summary of typical defenses against four types of poisoning attacks.

Defense Attack	Deep-kNN [20]	CD [28]	DUTI [29]	TRIM [22]	Sever [30]	**De-Pois**
TCL-attack [31]	√	○	○	○	○	√
pGAN-attack [32]	○	√	○	○	○	√
LF-attack [23]	○	√	√	○	√	√
R-attack [22]	○	○	√	√	√	√

Effective defense: √ Ineffective defense: ○

examples. This attack consists of the generator, the discriminator and the classifier, and utilizes the discriminator to distinguish the pristine and the generated poisoning data. A hyperparameter α is used to trade off the detectability and effectiveness of the poisoned samples, where a higher value of α indicates more chances for the crafted samples to evade the detection.

As an effective defense method against pGAN-attack, certified defense (CD) [28] first filters outliers outside the estimated feasible set and then minimizes a margin-based loss for the rest of the data. Afterwards, CD explores certified defenses by computing data-dependent upper bounds on the testing loss, where samples poisoned by pGAN-attack with longer distance from the true class centroids can be identified.

C. Label-Flipping Attack (LF-attack)

Biggio et al. [23] develop an adversarial label flips method to attack SVM. Particularly, they increase the probability of flipping the label of input data which are categorized with high confidence, and in this way the attacking success rate is improved.

Label sanitization (LS) [34] is then proposed to mitigate label-flipping attacks, which relies on the observation that poisoned samples are far away from the decision boundary of SVM and thus have more chance to be relabelled.

Meanwhile, Zhang et al. [29] devise a stronger defense, dubbed DUTI, against the label-flipping attack. Specifically, DUTI formulates a bi-level optimization problem to handle the teaching task. Given a small portion of the trusted items, DUTI would learn to find the potentially corrupted labels, and then gives them to a domain expert for further inspection to identify outliers.

D. Regression Attack (R-attack)

Jagielski et al. [22] mostly focus on poisoning attacks for linear regression. They formulate the poisoning attack as a bi-level optimization problem and Karush-Kuhn-Tucker conditions are employed to solve the non-convex problem. Furthermore, a statistical-based poisoning attack is put forward for black-box attacks in which attackers could query the target model to find statistical features of the training samples.

To counter such attacks, TRIM [22] trains a regression model on a subset of samples with poisoned data and iteratively estimates the residuals. It is indicated that the subset of samples with the smallest residual can be identified as pristine. Also, DUTI [29] can be applied to solving poisoning attack under regression setting.

Summary. Existing defense techniques against poisoning attacks are *attack-specific* and can only defend some specific type of poisoning attacks (c.f. Table I). In particular, these defenses identify poisoned data following distinct analysis of either specific training loss which contains class label or regression value, or using nearest neighbors techniques. Thus, it is difficult for one type of defense to detect poisoned data when it comes to attacks for different learning tasks. It is noticed that a recent model-agnostic defensive mechanism dubbed Sever [30] can be adopted to defend different types of attacks, but may fail for some deep neural network (DNN) targeted attacks such as TCL-attack and GP-attack, due to the difficulty in the model fitting of diverse unknown DNN models. In practice where the defender can hardly get aware in advance of which type of attacks has been carried out, those attack-specific defenses may become ineffective any more. In contrast, De-Pois is a *generic* and *attack-agnostic* defense approach which can work effectively for all the aforementioned types of attacks, and is more promising in real-world scenarios.

III. Threat Model and Defense Capability

In this section, we first provide a detailed threat model for the poisoning attacks described in the previous section and then describe the power of defenders. The threat model consists of the attacker's goal, the attacker's knowledge of the target model, as well as the attacker's capability of how to influence the training data. The defender's capability, on the other hand, is largely related to the obtained knowledge under the threat model.

Attacker's goal. The attacker's goal can be categorized into two classes [35]: the poisoning availability attack, which aims to affect the model's prediction performance indiscriminately, e.g., pGAN-attack [32], LF-attack [23] and R-attack [22], and the poisoning integrity attack, where the attacker aims to bring about specific prediction errors for some desired testing samples, e.g., TCL-attack [31].

Attacker's knowledge. The attacker may have different levels of knowledge of the target ML model. In white-box attacks [22], [23], [32], the attacker is expected to know the unadulterated training data, the feature values of each sample, the ML algorithm, and the parameters of the trained model. In the black-box attack [31], the attacker has no knowledge of the aforementioned information but can acquire a substitute dataset with a similar distribution of the original training data.

Attacker's capability. The attacker's capability is typically refers to how and to what extend the attacker controls the training data. Normally, the attacker can modify the feature values and labels of the training data [22], but one may also modify only feature values [31], [32] or only labels [23]). On the other hand, the attacker is often constrained by upper bounding the number of poisoning samples, and the ratio of poisoning samples below 30% is often mandatory.

Defender's Capability. The defender could obtain different levels of knowledge of the target model and the training data, according to different threat models. For example, Deep-kNN [20] assumes the availability of the ground-truth labels to compare the class labels with each sample's k neighbors. CD [28] supposes the outliers do not have a strong effect on the target model in order to make an approximation about the upper bounds on the testing loss across poisoning attacks under non-convex settings. TRIM [22] makes a strong assumption on the pre-knowledge of the ratio of poisoning samples which is controlled by the attacker, while Sever [30] makes underlying assumptions on small singular values of the gradients of the training data. Similar to DUTI [29], De-Pois assumes to have access to partial trusted training data. From the defender's perspective, one may need the access to corresponding resources so as to defend against different types of attacks in practice, and the defender's capability thus may change under different threat models.

IV. De-Pois Design

We formulate our problem as follows: for a training dataset $S_o = S_p \cup S_c$, which contains a potentially poisoned dataset S_p and a clean dataset S_c, De-Pois aims to determine whether a sample $s \in S_o$ is in S_p or not, given a small amount of (trusted) clean data $S_t \in S_o$. De-Pois relies on the observation that poisoned samples are more likely to have different predictions than clean samples do. Therefore, De-Pois tests out the poisoned samples by estimating their prediction difference, making use of a mimic model with a similar prediction behavior of the target model trained by S_c. To this end, De-Pois mainly consists of three steps (c.f. Fig. 2).

(1) **Synthetic Data Generation.** The first step of De-Pois is to generate sufficient synthetic training data with a similar distribution of S_c, in condition that only S_t can be obtained in practice. To better understand the underlying distribution of S_c, De-Pois leverages cGAN for data generation and devises an authenticator to supervise the data augmentation.

(2) **Mimic Model Construction.** After obtaining sufficient valid data, De-Pois next builds the mimic model by developing the conditional version of WGAN-GP to learn the distribution of predictions of the augmented training data. When the training of the conditional WGAN-GP is completed, we regard its *Discriminator* as our mimic model.

(3) **Poisoned Data Recognition.** Given the mimic model, De-Pois can thus employ a detection boundary to set apart the poisoned samples from clean ones. If the mimic model's output is lower than our detection boundary, the sample is then regarded as being poisoned.

A. Synthetic Data Generation

In general, our synthetic data generation module consists of two parts: a cGAN-based generator and an authenticator.

1) cGAN-based Generator: The original GAN contains two neural networks [25]: the Generator (G) and the Discriminator (D). G learns to generate synthesized samples $G(\boldsymbol{z})$ which captures the distribution of training data $P_{data}(\boldsymbol{x})$ from a prior noise distribution $\boldsymbol{z}$, while D learns to distinguish real data samples $\boldsymbol{x}$ from $G(\boldsymbol{z})$. G and D are learned simultaneously to achieve the following min-max objective:

$$\begin{aligned}\min_G \max_D V(G, D) = &\mathbb{E}_{\boldsymbol{x}\sim P_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] \\ &+ \mathbb{E}_{\boldsymbol{z}\sim P_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].\end{aligned} \tag{1}$$

It is noticed that in the original GAN, no control on modes of the generated data is employed. Therefore, it is not effective for GAN to guide the data generation process in this unconditioned generative model. By making G conditioned on additional information (e.g., the class labels), cGAN synthesizes new training samples by applying a random noise vector $\boldsymbol{z}_c$ and a condition constraint like class labels or other modalities to train the model [26].

To take advantage of cGAN, we thus propose to feed the additional information $\boldsymbol{y}$ into both the Generator and the Discriminator (i.e., G_c and D_c) in cGAN, and generate samples conditioned on $\boldsymbol{y}$ in a supervised way. It is important to note that the regression value $\boldsymbol{y}$ adding an additive unimodal noise $\boldsymbol{z}_r$ (e.g., Gaussian noise) is conditioned for regression tasks in order to cover the whole set of outputs as much as possible. The objective function can thus be as:

$$\begin{aligned}L_{cGAN} &= V(G_c, D_c) \\ &= \mathbb{E}_{\boldsymbol{x}\sim P_{data}(\boldsymbol{x})}[\log D_c(\boldsymbol{x}|\boldsymbol{y})] \\ &+ \mathbb{E}_{\boldsymbol{z}_c\sim P_{\boldsymbol{z}_c}(\boldsymbol{z}_c)}[\log(1 - D_c(G_c(\boldsymbol{z}_c|\boldsymbol{y})))].\end{aligned} \tag{2}$$

2) Authenticator: Intuitively, cGAN can generate sufficient data. However, the generated data generally has less diversified expressions mainly due to the single distribution (e.g. Gaussian distribution) fed to G_c and it is uncertain if cGAN can generate data with high fidelity and variety in low-volume data scenario [36].

To generate more valid training data, we thus introduce an authenticator to supervise the data augmentation process in cGAN. In particular, we treat these newly synthesized samples generated by G_c at each iteration as instances of missing latent variables which lie in the space of training data. We then calculate the loss L_A between the authenticator's prediction output and the true class label or regression value for each synthesized sample. At last, we back-propagate the loss L_A to the cGAN part, with the loss of D_c (resp. G_c) as $L_{cGAN}+L_A$ (resp. $L_{cGAN} - L_A$). It is noted that the idea of adopting an authenticator is motivated by the Bayesian DA model [36], which deals with only classification tasks. We extend the authenticator to work with both classification and regression tasks, and improve the granularity of the training process.

Due to different poisoning attack tasks (i.e., classification and regression tasks), the authenticator calculates the loss L_A in a different way. For classification, the authenticator is designed as a convolutional neural network (CNN). We first get the output $\hat{\boldsymbol{y}}$ of the authenticator, and then use the cross entropy error function to calculate the loss for classification,

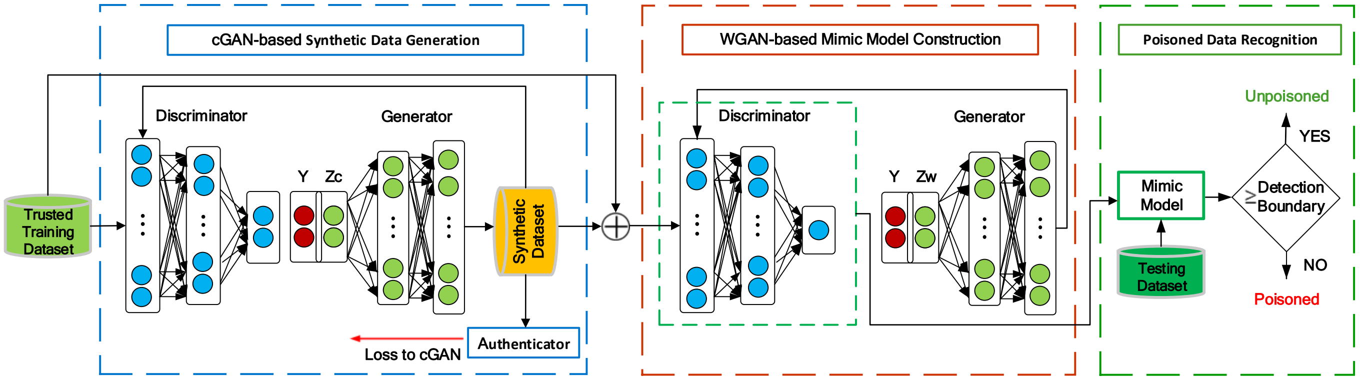


Fig. 2: The framework of De-Pois, which contains three parts: the cGAN-based synthetic data generation, the WGAN-based mimic model construction and the poisoned data recognition.

which can be formulated by:

$$L_A = -\frac{1}{M_s}\sum_{i=1}^{M_s}\sum_{j=1}^{N_c} y_i^j \log(\hat{y_i^j}), \tag{3}$$

where $\hat{y_i^j}$ is the prediction probability of the ith sample belonging to class j, N_c is the number of classes and M_s is the number of synthesized sample at each iteration. $y_i^j = 1$ if the label of the ith sample belongs to class j; otherwise $y_i^j = 0$.

For regression, the authenticator calls for a specific regression module (e.g., LASSO), and the loss is formulated by Equ. (4) using mean square error (MSE) of each synthesized sample:

$$L_A = \frac{1}{M_s}\sum_{i=1}^{M_s}(y_i - \hat{y}_i)^2, \tag{4}$$

where y_i represents the regression value of the ith sample and $\hat{y_i}$ is the prediction value of the authenticator for the ith sample at each iteration.

In this way, the authenticator encourages better distinction between the real data and the generated data, and thereby can enhance the data augmentation process.

3) Synthetic Data Generation: The cGAN training process of synthetic data generation involves two parts: for the discriminative part, we use the trusted clean samples S_t as its input, aiming to minimize $L_{cGAN} + L_A$. For the generative part, a noise prior $\boldsymbol{z}_c$ and additional information $\boldsymbol{y}$ are combined as its input, aiming to minimize $L_{cGAN} - L_A$. In the synthetic data generation process, these two parts are optimized in an adversarial way as normally done in GAN.

In order to determine the parameters in cGAN-based synthetic data generation process, we employ Monte Carlo Expectation Maximization (MCEM) and run it iteratively, where we first utilize Monte Carlo (MC) to estimate the value of model's parameters based on the estimation results of the previous iteration and then update these parameters with stochastic gradient decent (SGD) at each iteration.

In the expectation-maximization (EM) algorithm, we estimate the parameters θ of our synthetic data generation model using the trusted clean data $S_t = \{\boldsymbol{s}_i\}_{i=1}^{N_t}$ which corresponds to a given data $\boldsymbol{s} = (\boldsymbol{x}, \boldsymbol{y})$, with $\boldsymbol{x}$ representing data sample, $\boldsymbol{y}$ denoting the true label (or regression value), and $N_t = |S_t|$. The training process can be formalized as the following optimization problem:

$$\theta^* = \arg\max_{\theta} \log p(\theta|\boldsymbol{s}). \tag{5}$$

It is noticed that we cannot calculate the posterior $p(\theta|\boldsymbol{s})$ directly in the absence of information such sa the prior and likelihood function. So we increase the training data using synthesised data which can be represented by a latent variable $\boldsymbol{z}_s = (\boldsymbol{x}_s, \boldsymbol{y}_s)$, where $\boldsymbol{x}_s$ represents a synthesized data, and $\boldsymbol{y}_s$ refers to the associated class label (or regression value). Given both $\boldsymbol{s}$ and $\boldsymbol{z}_s$, we can estimate the augmented posterior $p(\theta|\boldsymbol{s}, \boldsymbol{z}_s)$ at the ith iteration in E-step:

$$Q(\theta, \theta^i) = \mathbb{E}_{p(\boldsymbol{z}_s|\theta^i, \boldsymbol{s})}[\log p(\theta|\boldsymbol{s}, \boldsymbol{z}_s)]. \tag{6}$$

Then, M-step maximizes the Q function in the next iteration:

$$\theta^{i+1} = \arg\max_{\theta}\; Q(\theta, \theta^i). \tag{7}$$

We stop the training process once $||\theta^{i+1} - \theta^i||$ is sufficiently small, and obtain the optimal θ^* from the last iteration.

We further employ MC strategy which uses large repeated random sampling to approximate the integration in E-step [37]. Furthermore, in M-step, we update θ^{i+1} by running SGD which uses only a sub-set of trusted clean data and augmented data in each iteration and can finally acquire the expected synthetic data generation model, leading to sufficient synthetic training data S_s with a similar distribution of S_c.

Note that our aim is to obtain the augmented data S_{aug}, such that $|S_{aug}| = |S_t| + |S'_s|$ is comparable to $|S_o|$. In most cases in our experiments, $|S_s| > |S_{aug}|$, thus we randomly choose a subset of S_s as S'_s. In some cases when $|S_t| + |S_s| < |S_{aug}|$, we can continue the training process to generate more data until $|S'_s|$ satisfies our setting.

B. Mimic Model Construction

After obtaining S_{aug}, De-Pois next aims to construct the mimic model which has similar prediction performance with the target model. The idea is straightforward: we can regard the mimic model as functionally equivalent with the target model if the prediction outputs of the mimic model trained on S_{aug} are indistinguishable from those of the target one.

Simply using GAN to construct our mimic model seems feasible. However, we find that the original GAN suffers from training instability, mainly due to the discontinuity of the generator's parameters when minimizing the Jensen-Shannon (JS) divergence in practice. To circumvent tractability issues, we propose a conditional version of WGAN-GP by making both the Generator and Discriminator parts of WGAN-GP conditioned on additional information $\boldsymbol{y}$. In this manner, we could construct our mimic model better in a supervised way.

In the paradigm of WGAN-GP [27], the training instability issue is solved by penalizing on the norm of weights for random samples $\hat{\boldsymbol{x}} \sim P_{\hat{\boldsymbol{x}}}$ from its Discriminator network to make it satisfy Lipschitz constraint, and the gradient penalty is directly added to the Wasserstein distance [38]. The objective function is formulated as:

$$\begin{aligned} L_{\mathit{WGAN\text{-}GP}} = & \mathbb{E}_{\widetilde{\boldsymbol{x}} \sim P_g}[D_w(\widetilde{\boldsymbol{x}})] - \mathbb{E}_{\boldsymbol{x} \sim P_r}[D_w(\boldsymbol{x})] \\ & + \lambda \mathbb{E}_{\hat{\boldsymbol{x}} \sim P_{\hat{\boldsymbol{x}}}}[(||\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})||_2 - 1)^2], \end{aligned} \tag{8}$$

where the last item represents the penalty on the gradient norm. P_r and P_g represent the real and generated data distribution, respectively. $P_{\hat{\boldsymbol{x}}}$ denotes a uniform sampling distribution sampled from P_r and P_g.

The WGAN-GP model provides us a more stable training environment. However, as mentioned before, it also suffers from inefficiency in data generation process under unconditioned generative model. In addition, the deep learning model will lead to training overfitting in low-data scenario. By introducing additional information into the WGAN-GP setting, we feed $\boldsymbol{y}$ into both G_w and D_w and are thus able to mimic the target model better in a supervised way. Accordingly, the objective function of our mimic model combining both WGAN-GP and cGAN can be modified as:

$$\begin{aligned} L_{\mathit{cWGAN\text{-}GP}} = & \mathbb{E}_{\widetilde{\boldsymbol{x}} \sim P_g}[D_w(\widetilde{\boldsymbol{x}}|\boldsymbol{y})] - \mathbb{E}_{\boldsymbol{x} \sim P_r}[D_w(\boldsymbol{x}|\boldsymbol{y})] \\ & + \lambda \mathbb{E}_{\hat{\boldsymbol{x}} \sim P_{\hat{\boldsymbol{x}}}}[(||\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}}|\boldsymbol{y})||_2 - 1)^2]. \end{aligned} \tag{9}$$

During the training process of our mimic model, we alternately optimize D_w and G_w. After adequate epochs of training, and when the objective of both parts are converged, we complete the mimic model construction and regard D_w as our expected mimic model.

C. Poisoned Data Recognition

Using the mimic model, De-Pois can thus find out the poisoned samples in a straightforward way: simply setting a detection boundary, and comparing the value between the mimic model's output and the detection boundary. If the output's value is lower than our detection boundary, the sample is then regarded as being poisoned. Otherwise, the sample belongs to unpoisoned.

To be more concrete, in the context of both classification and regression tasks, each sample x is fed into the mimic model and its prediction value y_{pre} is output. In our mimic model, y_{pre} represents the Wasserstein distance between the generated sample and the real sample. In general, the prediction value y_{pre} of a clean sample is larger than that of a poisoned one. Thus the poisoned samples can be recognized from the clean ones given a proper detection boundary. Since in practice we cannot know the clean samples S_c in advance, and the distribution of our augmented samples S_{aug} is designed to be similar to that of S_c, we thus utilize S_{aug} to determine the detection boundary.

It is observed that the distribution of the prediction values of S_{aug} passing D_w (denoted as $P_{S_{aug}}$) almost fits a normal distribution. Thus it is feasible to determine the detection boundary only using S_{aug} without the knowledge of the poisoned samples. So in order to properly determine the detection boundary, we first obtain the mean μ and standard deviation σ of $P_{S_{aug}}$. After that, we calculate z-scores by standardizing the distribution of the prediction values of testing samples passing D_w (denoted as $P_{S_{test}}$), which can be formulated by $\tilde{z}^i = \frac{y^i_{pre} - \mu}{\sigma}$, where y^i_{pre} and $\tilde{z}^i$ are the prediction value and the corresponding z-score of the ith testing sample. We can then acquire N z-scores corresponding to N testing samples, which enables us to distinguish the difference in $P_{S_{test}}$ with different means and standard deviations. From practical rule, we regard testing samples with unilateral confidence interval higher than z_s as clean ones given the value of level of significance (e.g., 0.05 in our experiments). Then we can obtain corresponding proper z_s by looking up the standard normal distribution table (e.g., $z_s = -1.96$ corresponding to the level of significance 0.05).

Finally, we can establish the detection boundary $y_{thr} = z_s \times \sigma + \mu$ and compare it to a sample's prediction y_{pre}. If the following inequation holds,

$$y_{pre} < y_{thr}, \tag{10}$$

we then regard this sample as being poisoned.

After testing all the samples in $S_o \setminus S_t$, we can identify the poisoned data, which is further excluded from S_o. In this way, De-Pois can be utilized as a filter prior to training the ML model so that the trained model will not be affected by the poisoned samples.

V. Discussion

In this section, we discuss some issues regarding the applicability of De-Pois from the following two aspects: application scope, as well as defending scope.

A. Application Scope of De-Pois

There are generally two strategies for defending against poisoning attacks. The first one is to devise a robust optimization algorithm which focuses on reducing the impact of the poisoning samples on the prediction performance of the target model. For example, in TRIM [22] the model is trained on the dataset with the lowest residuals and the impact of poisoning samples is weakened iteratively. Similar strategy is also adopted by Sever [30] and DUTI [29]. The other strategy is to perform poisoned data removal on the training data and then employ proper ML algorithms on the sanitized data. Under this situation, the defense methods need to find out the poisoned data precisely, e.g., CD [28] and our De-Pois. In some scenarios, identifying the poisoned data is necessary and sometimes crucial, especially for tracing the poisoning sources. For example, in intelligent crowd-sensing systems that rely on continuously collecting samples from the physical world, it is more desirable to find out and remove the malicious participators. In this sense, De-Pois can be used not only to defend poisoning attacks, but also to facilitate system ML model developers fixing systematic training set errors caused by data poisoners.

B. *Defending Scope of De-Pois*

In Section II, we have described several types of poisoning attacks which De-Pois is able to defend against (c.f. Table I). Now that De-Pois is a GAN-based approach, one question is naturally raised: will De-Pois still work if an attacker adopts a similar GAN-based approach to generate the poisoning samples? Here, we introduce another interesting GAN-based method called GP-attack [21] that illustrates how DNN could be used to generate malicious training samples. A generative method was proposed to accelerate the poisoned sample generation, using an autoencoder to generate poisoned samples and a discriminator to calculate the loss with GAN. For the defense against GP-attack, CD [28] can be an option as it can remove poisoned data far away from the true class centroids. Our De-Pois also works and is validated by experiments to be more effective than CD (c.f. Section VI-G).

VI. Performance Evaluation

A. *Experiment Setup*

1) Datasets: We perform experiments using four datasets in different domains including hand-written digit recognition, image classification, non-linear binary classification, and house sale price prediction, which are also adopted in existing poisoning attacks and defenses [20], [22], [23], [39], [40].

MNIST[2]. This dataset consists of 28×28 gray-scaled images of the handwritten digits from 0 to 9, along with a training dataset of $60,000$ images and a testing dataset of $10,000$ images.

CIFAR-10[3]. This dataset contains $60,000$ 32×32 color images in 10 different classes, with $50,000$ training images and $10,000$ testing images. These 10 different classes include cats, deer, dogs, frogs, airplanes, cars, horses, ships, birds, and trucks.

Fourclass Dataset[4]. This dataset has preprocessed into two-class stored in LIB-SVM repository and is not linearly separable. It contains 862 samples in a two-dimensional, bounded data space.

House Pricing Dataset[5]. This dataset utilizes predictor variables such as the number of bedrooms and lot square footage to predict house sales prices. It contains $1,460$ houses and 81 features. For preprocessing, categorical features are converted by one-hot encoding as vectors, and each element in these vectors is regarded as an independent feature. All the features are then normalized into 275 total features.

In the experiment, we randomly split this dataset into a training dataset and a testing dataset, with 70% and 30% of data, respectively, for both Fourclass and House Pricing datasets. In addition, for each dataset, we randomly choose the corresponding proportion of the samples as the trusted clean dataset, which is then augmented to the same size as the original training dataset for the mimic model construction.

2) Baselines: We compare De-Pois to some typical defense techniques, against four types of poisoning attacks as described in Section II and Table I. It is noticed that existing defenses are attack-specific, and thus we only report the defending results against their corresponding attacks (i.e., those marked with $\surd$ in Table I) while neglecting the ineffective defending results (i.e., those marked with $\bigcirc$ in Table I).

3) Evaluation Metrics: To evaluate the performance of De-Pois on defense against poisoning attacks, we adopt the standard metrics: *Accuracy*, *Recall* and *F1-score*. Accuracy measures the ratio of samples predicted correctly in all the predicting samples, while F1-score is defined as the harmonic mean of *Recall* and *Precision*, expressed as

$$F1 = \frac{\mathit{Precision} \times \mathit{Recall}}{2 \times (\mathit{Precision} + \mathit{Recall})}, \tag{11}$$

where *Recall* (resp. *Precision*) measures the ratio of correctly predicted positive samples over all the positive samples (resp. all the predicted positive samples). It is known that the higher F1-score value indicates the better performance of the defense.

To evaluate the performance of the synthetic data generation in De-Pois, we adopt the following metrics: Inception Score (IS) [41], Fréchet Inception Distance (FID) [42], [43], Wasserstein Distance (WD) [38] and Average Euclidean Distance (AED).

Specifically, IS evaluates the quality of a generated image by calculating the KL-divergence between the conditional class distribution $p(y|\boldsymbol{x})$ and the marginal class distribution $p(y)$. Also, the conditional class distribution $p(y|\boldsymbol{x})$ is obtained by applying the Inception model to each image. The score is given by:

$$\mathit{IS} = \exp(\mathbb{E}_{\boldsymbol{x} \sim p_g}[D_{KL}(p(y|\boldsymbol{x}))||p(y)]). \tag{12}$$

FID measures the similarity of Gaussian distributions of Inception embedding of real and synthetic images, which is defined as:

$$\mathit{FID}(r,g) = ||\boldsymbol{m}_r - \boldsymbol{m}_g||^2 + Tr(\boldsymbol{C}_r + \boldsymbol{C}_g - 2(\boldsymbol{C}_r\boldsymbol{C}_g)^{\frac{1}{2}}), \tag{13}$$

where $\boldsymbol{m}_r$ (resp. $\boldsymbol{m}_g$) and $\boldsymbol{C}_r$ (resp. $\boldsymbol{C}_g$) represent the mean and covariance of the real (resp. synthetic) image.

WD denotes the optimal transport cost in order to transform the distribution P_r to P_f, which can be formulated as:

$$\mathit{WD}(P_r, P_f) = \inf_{\gamma \in \prod(P_r, P_f)} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}) \sim \gamma}\left[||\boldsymbol{x} - \boldsymbol{y}||\right], \tag{14}$$

where $\prod(P_r, P_f)$ is the set of joint distributions $\gamma(\boldsymbol{x}, \boldsymbol{y})$ and P_r, P_f represent the marginals of $\gamma(\boldsymbol{x}, \boldsymbol{y})$.

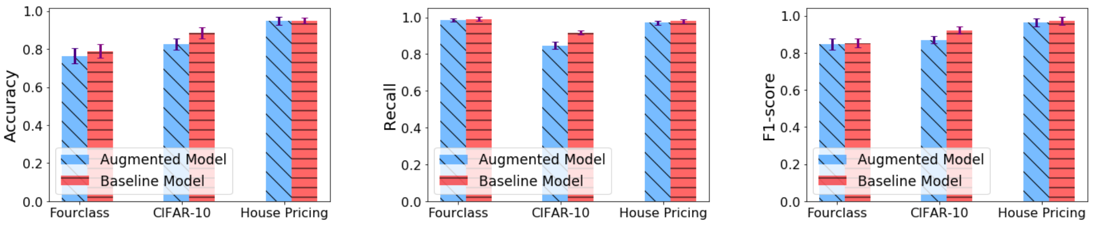
AED measures the mean distance between the pair elements of $X = \{x_{ik} : 1 \le i \le M, 1 \le k \le K\}$ and $Y =$

[2] http://yann.lecun.com/exdb/mnist

[3] https://www.cs.toronto.edu/ kriz/cifar.html

[4] https://www.csie.ntu.edu.tw/%7ecjlin/libsvmtools/datasets/binary.html

[5] https://www.kaggle.com/c/house-prices-advanced-regression-techniques



(a) Accuracy

(b) Recall

(c) F1-score

Fig. 4: Effectiveness of synthetic data generation.

$\{y_{jk} : 1 \leq j \leq N, 1 \leq k \leq K\}$, expressed as

$$AED = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left[\sum_{k=1}^{K} (x_{ik} - y_{jk})^2 \right]^{\frac{1}{2}}, \qquad (15)$$

where M (resp. N) is the total number of samples in X (resp. Y) and K is the number of features in each sample. x_{ik} (resp. y_{jk}) represents the value of the kth feature in the ith (resp. jth) sample.

Note that for each testing, we run the experiment for 5 times to average out the randomness.

4) Model Settings: For synthetic data generation, we utilize cGAN architecture and choose a fixed structure to generate data. In the generator network G_c, a noise prior $\boldsymbol{z}_c$ with 100 dimensional normal distribution and class labels or regression values $\boldsymbol{y}$ of trusted dataset with a mini-batch size of 128 (i.e., $|S_t| = 20\%|S_o|$) are combined as the input at the bottommost layer of G_c. Also, G_c use 3 full connection layers with Leaky Rectified Linear Unit (Leaky ReLu) activation and one full connection layer with sigmoid activation. The discriminator network D_c is constructed with 3 full connection layers and a sigmoid unit layer. Dropout is applied at intermediate layers of D_c and the dropout value is set to 0.4. The slope of the leak in the Leaky ReLu is set to 0.2 in both G_c and D_c.

For the authenticator, we adopt a specific classifier module (i.e., CNN for images) in the context of classification, and we use the cross entropy function to calculate its loss. For regression, the authenticator calls a specific regression module (i.e., LASSO) and the loss is obtained using MSE.

For the mimic model construction, we improve WGAN-GP by feeding the class labels or regression values $\boldsymbol{y}$ as in cGAN into both G_w and D_w as additional inputs. Similarly, a noise prior $\boldsymbol{z}_c$ with 100 dimensional normal distribution and class labels or regression values $\boldsymbol{y}$ of augmented dataset with a mini-batch size of 32 are combined as the input at the bottommost layer of G_w. We use full connection layer and activating layer in both G_w and D_w. Especially, we add convolutional layer for images in order to obtain a better mimic model. We use RMSprop optimizer with the learning rate equal to 0.00005 and D_w iterates 5 times when G_w iterates once.

B. Determining Detection Boundary

We begin with determining the detection boundary y_{thr} for each dataset using the method discussed in Section IV-C, and we set $z_s = -1.96$ by default. Taking CIFAR-10 dataset for example, we first obtain the augmented samples consisting of $|S_t| = 20\%|S_o|$ and $|S'_s| = 80\%|S_o|$. Then we calculate the mean $\mu = -0.5726$ and the standard deviation $\sigma = 0.2109$ of $P_{S_{aug}}$. After that, we can compute the detection boundary by $y_{thr} = z_s \times \sigma + \mu = -0.9859$ for CIFAR-10 dataset.

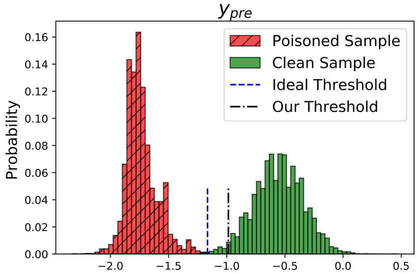
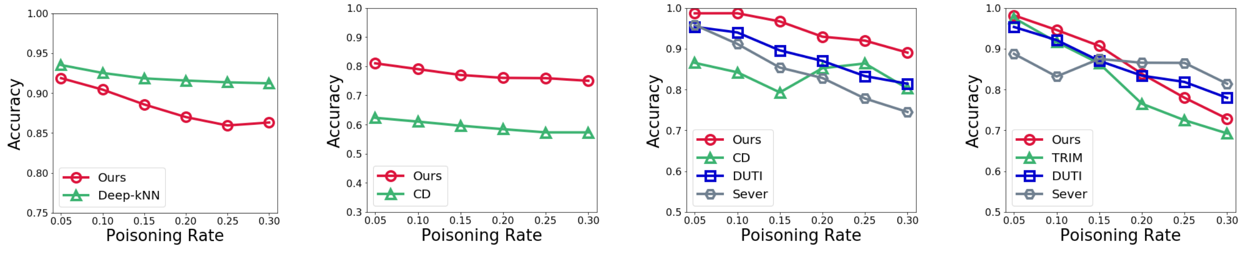


Fig. 3: Distributions of prediction values of the clean samples and the poisoned samples generated by TCL-attack.

In order to evaluate our detection boundary, we test 12,000 clean samples and 12,000 poisoned samples generated by TCL-attack for CIFAR-10 dataset. Then the prediction values of testing samples are calculated after being fed into the detection model trained on the augmented samples. The distribution of the prediction values are depicted in Fig. 3. We can observe that our detection boundary is very close to the ideal one and the poisoned sample can be distinguished from the clean sample given our detection boundary. Similarly, we determine the detection boundary $y_{thr} = -0.073$ for Fourclass dataset and $y_{thr} = 0.683$ for House Pricing dataset. In the following, we evaluate the performance of De-Pois using the obtained detection boundary for each dataset.

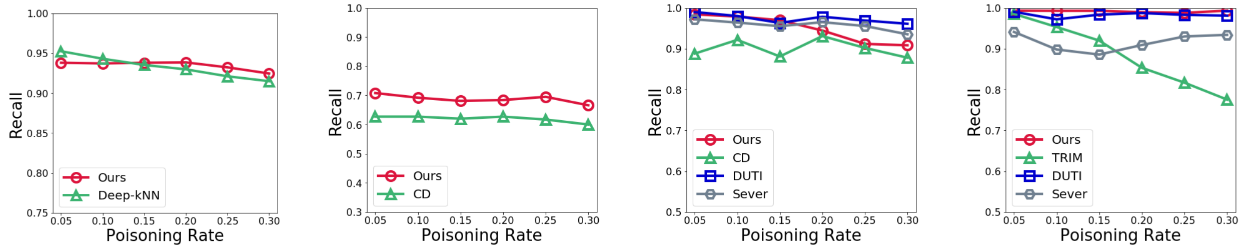
C. Effectiveness of Synthetic Data Generation

We first evaluate the effectiveness of the synthetic data generation, by comparing the accuracy, recall and F1-score of the two models: the *augmented model* trained on S_{aug}, and the *baseline model* trained on clean dataset with the same size as S_{aug}. The two models are trained with our cGAN and authenticator architecture on each dataset, and the trained discriminator is employed to evaluate the performance. To unify metrics for both classification and regression datasets, we test the two models on data with both clean and poisoned samples. For the regression dataset, we can thus obtain the results passing the model and then calculate the accuracy, the recall and F1-score as for the classification dataset.



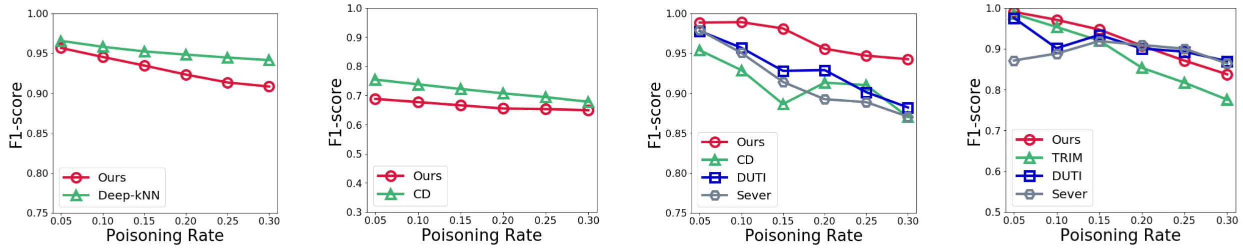
(a) TCL-attack (b) pGAN-attack (c) LF-attack (d) R-attack

Fig. 5: Impact of poisoning rate on accuracy in fixed trusted clean set environments.



(a) TCL-attack (b) pGAN-attack (c) LF-attack (d) R-attack

Fig. 6: Impact of poisoning rate on recall in fixed trusted clean set environments.



(a) TCL-attack (b) pGAN-attack (c) LF-attack (d) R-attack

Fig. 7: Impact of poisoning rate on F1-score in fixed trusted clean set environments.

For CIFAR-10, Fourclass and House Pricing datasets, we test $15,000$ clean data and $15,000$ poisoned data generated by TCL-attack, 500 clean data and 500 poisoned data generated by LF-attack, and $1,000$ clean data and $1,000$ poisoned data generated by R-attack, respectively. From Fig. 4 we can see that, the accuracy, recall and F1-score of the augmented model are quite close to those of the baseline model, thereby indicating that our synthetic data generation is effective for augmenting the trusted clean data.

Furthermore, Table II shows the IS and FID results on CIFAR-10 dataset, as well as WD and AED results on Fourclass and House Pricing datasets obtained by our synthetic data generation model and a comparison of baseline on real data and cGAN model. We can observe that our model can reach the performance as that from real data and outperforms the cGAN model, indicating that using authenticator could be beneficial to generate sufficient valid data.

TABLE II: Evaluation of synthetic data generation.

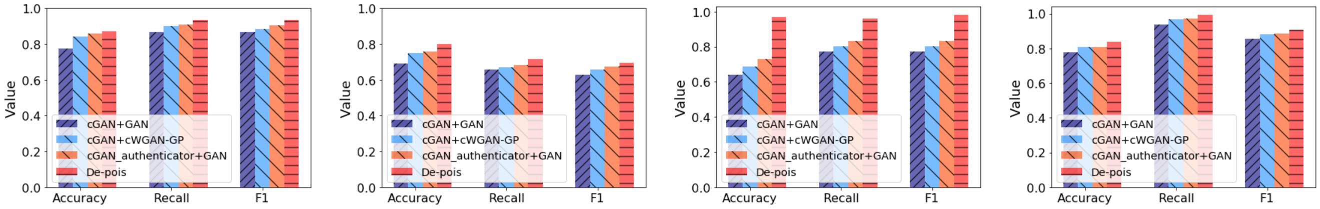
Dataset	CIFAR-10		Fourclass		House Pricing	
Metric Model	IS	FID	WD	AED	WD	AED
Real data	11.33	2.1	0.00	87.38	0.00	5.01
cGAN	6.10	51.03	0.89	91.65	2.20	7.93
De-Pois	8.23	15.2	0.61	87.97	1.98	7.84

IS: higher is better. FID / WD / AED: lower is better.

D. Impact of Poisoning Rate

We next evaluate the impact of poisoning rate $R_p = |S_p|/|S_c|$. In our settings, we set $|S_t| = 20\%|S_o|$. For each type of poisoning attacks, the results are reported in Figs. 5∼7, and it is confirmed that De-Pois is attack-agnostic, while other defenses are attack-specific. We describe the details as follows.

1) Effectiveness under TCL-attack on CIFAR-10 Dataset: We first assess De-Pois against TCL-attack [31]. We use CIFAR-10 dataset and select "ship" as the target class to craft the poisons, and "frog" as the targeted class. We compare De-Pois with Deep-kNN. From the results shown in Figs. 5∼7, we observe that Deep-kNN maintains relatively higher accuracy, recall and F1-score. This is largely because Deep-kNN defense against this kind of feature collision attack in feature space and find the characteristic that the plurality of poisoned data's



(a) TCL-attack (b) pGAN-attack (c) LF-attack (d) R-attack

Fig. 8: Impact of each part in De-Pois.

neighbors as clean ones in the target class. It is also noted that although the accuracy, recall and F1-score of De-Pois is over 3% lower than Deep-kNN on average, De-Pois still works in general, with the accuracy over 0.85 and F1-score over 0.9 in most cases.

2) Effectiveness under pGAN-attack on CIFAR-10 Dataset: We then evaluate De-Pois against pGAN-attack [32]. We use CIFAR-10 dataset and set $|S_t| = 20\%|S_o|$. We consider detectability constraints in pGAN-attack in realistic scenarios and observe that pGAN-attack can produce effective poisoning data when $\alpha = 0.5$. If α is small, the performance of De-Pois will be better because pGAN-attack considers less detectability constraint then; when the value of α increases, the generated poisoning data will become hard to detect for both CD and De-Pois. Since α controls the stealth of the attack and, as a trade-off, $\alpha = 0.5$ is set in our experiment. We compare De-Pois with CD, and the results are shown in Figs. 5∼7. As can be observed, the accuracy and recall of De-Pois are always higher than CD when the poisoning data increases, with an average 15% and 7% improvement, respectively. It is also shown that the F1-score of CD is 3% higher on average than that of De-Pois. The results indicate that De-Pois can defense pGAN-attack well.

3) Effectiveness under LF-attack on Fourclass Dataset: We next assess De-Pois under LF-attack strategy [23]. We use Fourclass dataset and set $|S_t| = 20\%|S_o|$. We compare De-Pois with DUTI, Sever and CD. The results in Figs. 5∼7 illustrate that De-Pois always performs better than other three methods as R_p increases, as De-Pois is good at learning the distribution of features especially for classification tasks. Also, DUTI has comparable performance with Sever. The accuracy and F1-score of DUTI (resp. Sever) is over 0.85 and 0.85 (resp. 0.8 and 0.85), respectively, as R_p increases.

4) Effectiveness under R-attack on House Pricing Dataset: Finally, we analyze De-Pois against R-attack [22]. We use House Pricing dataset and set $|S_t| = 20\%|S_o|$. We compare De-Pois with TRIM, DUTI and Sever. In Figs. 5∼7, the results indicate that De-Pois always performs well than TRIM. The main reason is that TRIM identifies the poisoned data with the lowest residual. In contrast, De-Pois can identify poisoned samples whose prediction values of the mimic model are lower than those of the clean ones. Moreover, De-Pois performs better than Sever and DUTI when the poisoned rate is lower than 20%, while Sever and DUTI perform better when the poisoned rate continues increasing. This is largely because Sever and DUTI contain several retrain process after they detect outliers and thus have more chance to filter out the possible poisoned samples. It can also be seen that De-Pois performs well regardless of the way how poisoning samples are manipulated, which makes De-Pois a more generic solution.

E. Impact of Trusted Clean Data Size

To evaluate the impact of the size of S_t, we set R_p to a fixed value (e.g. $R_p = 20\%$ in our settings), and conduct experiments in a similar way to the previous section. The results are shown in Table III, which reveal that the accuracy, recall and F1-score of De-Pois are higher than 0.9 on average and fluctuate less than 10% as the size of S_t increases under TCL-attack, LF-attack and R-attack. It is noticed that De-Pois performs well against pGAN-attack with rational detectability constraints. Also, it has comparable effectiveness with Deep-kNN against TCL-attack, even when $|S_t| = 5\%|S_o|$, which verifies again that De-Pois is attack-agnostic, and is effective for both classification and regression tasks.

F. Impact of Each Component in De-Pois

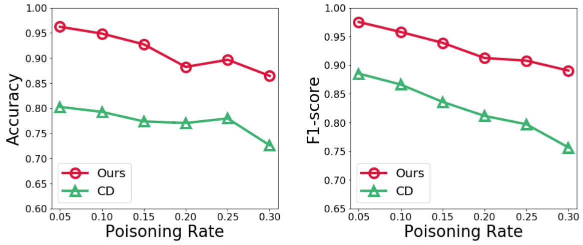
We also evaluate the impact of each component in De-Pois. We compare De-Pois (i.e. cGAN_authenticator+cWGAN-GP, representing cGAN with the authenticator for synthetic data generation, and cWGAN-GP for mimic model construction) with three other different compositions, namely cGAN+GAN, cGAN+cWGAN-GP and cGAN_authenticator+GAN. In the experiments $R_p = 20\%$ and $|S_t| = 20\%|S_o|$. The results are shown in Fig. 8, our authenticator improves on accuracy, recall, and F1 by at least 0.03, 0.04 and 0.03, respectively. Also, our mimic model construction part improves on accuracy, recall, and F1 by at least 0.05, 0.04 and 0.03, respectively. In a word, De-Pois demonstrates the effectiveness and performance improvements of all the components used simultaneously.

G. Effectiveness under GP-attack on MNIST Dataset

We then evaluate De-Pois against GP-attack developed in [21]. We apply the direct gradient method to craft poisoned training data. We also use the MNIST dataset and set $|S_t| = 20\%|S_o|$. We compare De-Pois with CD. As can be observed in Fig. 9, the accuracy and F1-score of De-Pois are always higher than CD when R_p increases, with an average 13% and 10% improvement, respectively. The primary cause is that CD was initially designed for LF-attack, and thus is not that well-directed for GP-attack. mainly due to the high sensitivity of the presence of the poisoned data.

TABLE III: Comparison of accuracy and F1-score in fixed poisoning rate environments.

			$\|S_t\|$					
Metric	**Attack**	**Defense**	$5\%\|S_o\|$	$10\%\|S_o\|$	$15\%\|S_o\|$	$20\%\|S_o\|$	$25\%\|S_o\|$	$30\%\|S_o\|$
Accuracy	TCL-attack	Ours	0.756±0.05	0.843±0.05	0.848±0.03	0.869±0.03	0.851±0.03	0.927±0.02
		Deep-kNN	**0.915±0.02**	**0.905±0.02**	**0.910±0.02**	**0.909±0.02**	**0.918±0.02**	**0.912±0.02**
	pGAN-attack	**Ours**	**0.756±0.03**	**0.767±0.03**	**0.781±0.03**	**0.783±0.02**	**0.790±0.02**	**0.828±0.02**
		CD	0.635±0.03	0.631±0.03	0.626±0.03	0.636±0.03	0.631±0.03	0.632±0.03
	LF-attack	**Ours**	**0.889±0.05**	**0.919±0.05**	**0.922±0.04**	**0.929±0.04**	**0.933±0.04**	**0.931±0.03**
		CD	0.736±0.03	0.745±0.03	0.762±0.03	0.769±0.03	0.771±0.02	0.782±0.02
		DUTI	0.766±0.04	0.785±0.04	0.809±0.04	0.821±0.04	0.853±0.03	0.873±0.03
		Sever	0.845±0.04	0.857±0.03	0.852±0.03	0.840±0.04	0.848±0.04	0.837±0.04
	R-attack	Ours	0.832±0.03	0.836±0.03	0.839±0.03	0.838±0.03	0.836±0.03	0.840±0.03
		TRIM	0.772±0.05	0.775±0.05	0.771±0.05	0.768±0.05	0.772±0.05	0.775±0.05
		DUTI	0.885±0.02	0.890±0.02	0.890±0.02	0.886±0.02	0.885±0.02	0.890±0.02
		Sever	**0.856±0.03**	**0.865±0.02**	**0.860±0.02**	**0.863±0.02**	**0.848±0.02**	**0.852±0.02**
Recall	TCL-attack	Ours	0.744±0.05	0.887±0.05	0.867±0.03	**0.938±0.03**	0.873±0.03	**0.967±0.01**
		Deep-kNN	**0.927±0.02**	**0.913±0.03**	**0.925±0.02**	0.925±0.02	**0.935±0.03**	0.930±0.03
	pGAN-attack	**Ours**	**0.674±0.04**	**0.688±0.03**	**0.702±0.04**	**0.711±0.03**	**0.722±0.03**	**0.735±0.03**
		CD	0.638±0.02	0.641±0.02	0.638±0.02	0.636±0.02	0.631±0.02	0.640±0.02
	LF-attack	**Ours**	0.930±0.04	**0.943±0.04**	**0.950±0.04**	**0.954±0.04**	**0.961±0.04**	**0.958±0.04**
		CD	0.926±0.03	0.931±0.03	0.932±0.03	0.937±0.03	0.916±0.03	0.922±0.03
		DUTI	0.916±0.02	0.922±0.02	0.934±0.02	0.938±0.02	0.947±0.02	0.953±0.02
		Sever	**0.941±0.03**	0.955±0.02	0.950±0.02	0.939±0.03	0.947±0.03	0.936±0.03
	R-attack	**Ours**	**0.992±0.03**	**0.985±0.03**	**0.967±0.03**	**0.980±0.03**	**0.988±0.02**	**0.973±0.02**
		TRIM	0.857±0.05	0.863±0.05	0.855±0.05	0.861±0.05	0.864±0.05	0.885±0.05
		DUTI	0.953±0.02	0.960±0.02	0.960±0.02	0.957±0.02	0.955±0.02	0.955±0.02
		Sever	0.893±0.03	0.912±0.02	0.903±0.03	0.908±0.03	0.889±0.03	0.890±0.03
F1-score	TCL-attack	Ours	0.851±0.05	0.904±0.05	0.903±0.03	0.923±0.03	0.892±0.03	**0.956±0.02**
		Deep-kNN	**0.948±0.02**	**0.933±0.02**	**0.943±0.03**	**0.939±0.03**	**0.952±0.02**	0.945±0.02
	pGAN-attack	**Ours**	0.678±0.03	0.685±0.03	0.700±0.03	0.703±0.03	**0.713±0.03**	0.722±0.03
		CD	**0.725±0.02**	**0.713±0.02**	**0.721±0.02**	**0.718±0.02**	0.711±0.02	**0.724±0.02**
	LF-attack	**Ours**	**0.930±0.05**	**0.943±0.05**	**0.950±0.05**	**0.954±0.05**	**0.961±0.05**	**0.958±0.05**
		CD	0.826±0.02	0.828±0.02	0.832±0.02	0.837±0.02	0.842±0.01	0.844±0.01
		DUTI	0.871±0.03	0.883±0.03	0.898±0.03	0.911±0.03	0.924±0.02	0.935±0.02
		Sever	0.885±0.03	0.905±0.02	0.897±0.02	0.876±0.03	0.882±0.03	0.869±0.03
	R-attack	Ours	0.907±0.03	0.907±0.03	0.908±0.03	0.910±0.03	**0.908±0.03**	0.906±0.03
		TRIM	0.858±0.05	0.880±0.05	0.882±0.05	0.885±0.05	0.886±0.05	0.885±0.05
		DUTI	0.895±0.02	0.899±0.02	0.895±0.02	0.90±0.02	0.897±0.02	0.90±0.02
		Sever	**0.912±0.03**	**0.923±0.02**	**0.915±0.03**	**0.918±0.03**	0.901±0.02	**0.906±0.02**



(a) Accuracy (b) F1-score

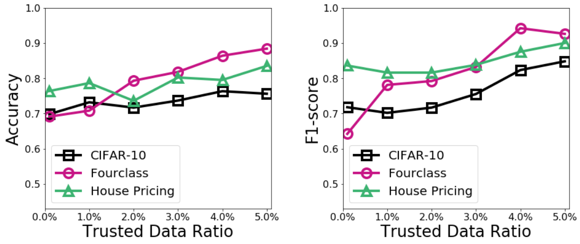
Fig. 9: Effectiveness under GP-attack on MNIST dataset.

H. Evaluation on Limitations of De-Pois

Since De-Pois largely relies on the small portion of the trusted clean data, it is thus crucial to explore the impact of the trusted clean data regarding the limitations of De-Pois.

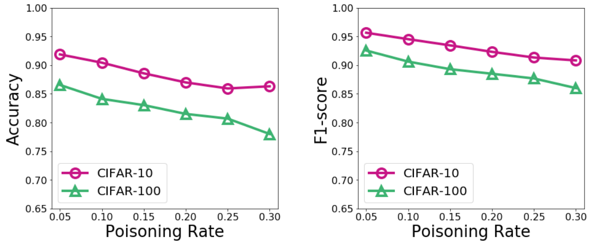
Intuitively, different sizes of the trusted clean training dataset could result in different prediction performances. It is thus reasonable to wonder that whether De-Pois is still applicable even when the amount of trust dataset is extremely small. We push the trusted data ratio to the extreme to test where De-Pois would fail. We set $|S_t|$ to $0.5\%|S_o|$, $1\%|S_o|$, $2\%|S_o|$, $3\%|S_o|$, $4\%|S_o|$, $5\%|S_o|$ and test TCL-attack on CIFAR-10 dataset, LF-attack on Fourclass dataset, and R-attack on House Pricing dataset. We observe from Fig. 10 that the accuracy and F1-score decrease when the trusted data ratio decreases. The accuracy and F1-score decrease from 90% to 70% when $|S_t|$ decreases from $5\%|S_o|$ to $0.5\%|S_o|$ on average. This illustrates that the trusted data ratio could have a great impact on the performance of De-Pois and it performs not so well when the trusted data ratio is quite low.

In practice, obtaining 100% clean data may be difficult, and it is sometimes inevitable that there could exist a small



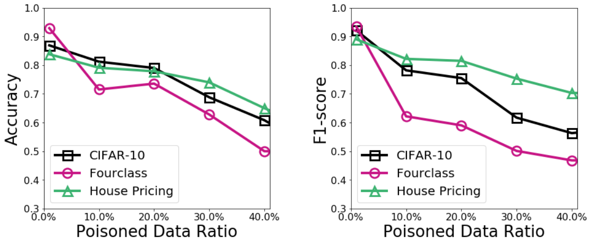
(a) Accuracy (b) F1-score

Fig. 10: Performance with insufficient trusted data.



(a) (b)

Fig. 12: Performance on CIFAR-100 dataset.



(a) Accuracy (b) F1-score

Fig. 11: Performance with poisoned data mixed in trusted data.

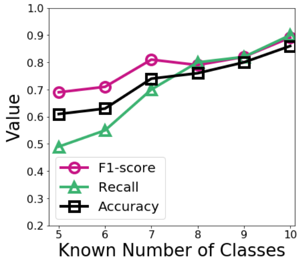


Fig. 13: Performance of different known number of classes.

percentage of poisoned samples in the trusted data. We thus alter the ratio of poisoned samples in the trusted data, and test TCL-attack on CIFAR-10 dataset, LF-attack on Fourclass dataset, and R-attack on House Pricing dataset to see how De-Pois works in this situation. As illustrated in Fig. 11, we notice that the accuracy and F1-score for three datasets decrease 10% on average when there exist poisoned data in the trusted training dataset. Also, the accuracy and F1-score fluctuate less than 5% when the poisoned data ratio continue increases. It should be emphasized that the performance of De-Pois drops fast when the poisoned data ratio is higher than 20%. This illustrates that the performance of De-Pois would be impacted when there is poisoned data mixed in the trusted data.

In addition, complex datasets with more classes could cause the performance degradation of De-Pois. We test De-Pois on CIFAR-100 dataset under TCL-attack and compare the results with those on CIFAR-10 dataset under the same experimental settings. We set $|S_t|$=20%$|S_o|$ and alter the poisoning rate from 5% to 30%. We can observe from Fig. 12 that the accuracy and F1-score of CIFAR-10 are 6% and 3% higher than those of CIFAR-100 on average. This illustrates that the performance of De-Pois will degrade on some more complex datasets.

Moreover, the synthetic data generation of De-Pois aims to enlarge the training dataset, but it depends on annotated datasets and would become less effective to generate synthetic training samples with labels not belonging to the category of these annotated datasets. In other words, our data augmentation method implicitly assumes to obtain the training samples of every class. We obtain the trusted samples from part of 10 classes with CIFAR-10 under TCL-attack, and randomly select the known number of classes in CIFAR-10 dataset (e.g., from 5 classes to 10 classes). Also, we set the trusted data $|S_t|$=20%$|S_o|$. Then we test the recall, accuracy and F1-score on $10,000$ clean samples and $10,000$ poisoned samples. The results in Fig. 13 have shown that the recall, accuracy and F1-score indeed decrease when the number of classes of trusted dataset decreases.

I. Evaluation on Run-time Overhead

Lastly, we evaluate De-Pois' run-time overhead, and compare with other defenses. We ran our experiments on NVIDIA Geforce GTX 1060 6GB and the experimental results are shown in Tables IV and V. The detection time for De-Pois is calculated after we obtain the mimic model and we count the total time for detection $1,000$ samples. For CIFAR-10 dataset, De-Pois takes 0.108 seconds for detecting $1,000$ images, while CD needs less time than De-Pois. For Fourclass and House Pricing datasets, De-Pois also takes the longest time. This is mainly due to the extra loading time of the mimic model of De-Pois. It is also noticed that the run-time overhead is largely affected by the dataset size where De-Pois spends longer time on CIFAR-10 dataset.

In addition, we report the training time of the synthetic data generation and the mimic model construction in De-Pois, so as to better understand the efficiency of De-Pois. For synthesizing every 500 samples of CIFAR-10, Fourclass, and House Pricing datasets, De-Pois takes 544 seconds, 27.1 seconds, and 32.3 seconds, respectively. It is noted that De-Pois takes longer time in training image dataset due to longer convergence time of the model. For mimic model construction on CIFAR-10, Fourclass, and House Pricing datasets, De-Pois takes 654 seconds, 30.75 seconds, and 179.5 seconds for every 500 samples, respectively. We can also observe that the dimension of the training data has an impact on the training time. CIFAR-10 and House Pricing datasets, which have higher dimensions, have much longer training time than that of Fourclass dataset.

TABLE IV: Comparison of detection time (s).

Dataset	CIFAR-10		Fourclass	
Methods	De-Pois	CD	De-Pois	DUTI
Detection Time	0.108	0.05	0.306	0.069

Dataset	House Pricing		
Methods	De-Pois	TRIM	Sever
Detection Time	0.068	0.007	0.027

TABLE V: Training time for De-Pois' components (s).

Training Time Dataset Component	CIFAR-10	Fourclass	House Pricing
Synthetic Data Generation	544	27.1	32.3
Mimic Model Construction	654	30.75	179.5

VII. Conclusion

In this paper, we have presented De-Pois, the first generic method for defending against data poisoning attacks. Given a small quantity of trusted clean data, we modify cGAN to obtain sufficient valid training data with a similar distribution of the trusted clean data. We further utilize conditional WGAN-GP to train an effective mimic model, which can achieve comparable prediction performance with the target model. By doing so, we can thus distinguish the poisoned data from the clean data by recognizing the prediction differences. Experiment results on four realistic datasets against several typical kinds of poisoning attacks validate that De-Pois is attack-agnostic and very effective, and performs better in most cases compared with existing defense techniques. We believe our work may deepen the understanding about the data poisoning attacking/defending mechanisms in practical settings and shed light on developing more efficient outlier detection techniques in broader areas.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[2] M. De Cock, R. Dowsley, C. Horst, R. Katti, A. C. A. Nascimento, W. Poon, and S. Truex, "Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 217–230, 2019.

[3] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proceedings of USENIX Security*, 2016, pp. 601–618.

[4] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proceedings of IEEE S&P*, 2017, pp. 3–18.

[5] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of ACM CCS*, 2015, pp. 1322–1333.

[6] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Proceedings of ACM CCS*, 2017, pp. 603–618.

[7] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proceedings of ECML PKDD*, 2013, pp. 387–402.

[8] J. Wu, B. Chen, W. Luo, and Y. Fang, "Audio steganography based on iterative adversarial attacks against convolutional neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2282–2294, 2020.

[9] B. Battista, N. Blaine, and L. Pavel, "Poisoning attacks against support vector machines," in *Proceedings of ICML*, 2012, pp. 1467–1474.

[10] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models," in *Proceedings of AAAI*, 2016, pp. 1452–1458.

[11] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners." in *Proceedings of AAAI*, 2015, pp. 2871–2877.

[12] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of ACM Workshop on AISec*, 2017, pp. 27–38.

[13] P. W. Koh, J. Steinhardt, and P. Liang, "Stronger data poisoning attacks break data sanitization defenses," *CoRR, arXiv: 1811.00741*, 2018.

[14] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *CoRR, arXiv: 1708.06733*, 2017.

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

[16] C. Chen, X. Zhao, and M. C. Stamm, "Generative adversarial attacks against deep-learning-based camera model identification," *IEEE Transactions on Information Forensics and Security*, to appear. DOI: 10.1109/TIFS.2019.2945198.

[17] A. Paudice, L. Muñoz-González, A. Gyorgy, and E. C. Lupu, "Detection of adversarial training examples in poisoning attacks through anomaly detection," *CoRR, arXiv: 1802.03041*, 2018.

[18] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proceedings of RAID*, 2018, pp. 273–294.

[19] J. Carnerero-Cano, L. Muñoz-González, P. Spencer, and E. C. Lupu, "Regularisation can mitigate poisoning attacks: A novel analysis based on multiobjective bilevel optimisation," *CoRR, arXiv: 2003.00040*, 2020.

[20] N. Peri, N. Gupta, W. Ronny Huang, L. Fowl, C. Zhu, S. Feizi, T. Goldstein, and J. P. Dickerson, "Deep k-NN defense against clean-label data poisoning attacks," *CoRR, arXiv: 1909.13374*, 2019.

[21] C. Yang, Q. Wu, H. Li, and Y. Chen, "Generative poisoning attack method against neural networks," *CoRR, arXiv: 1703.01340*, 2017.

[22] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proceedings of IEEE S&P*, 2018, pp. 19–35.

[23] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Proceedings of ACML*, 2011, pp. 97–112.

[24] C. Miao, Q. Li, L. Su, M. Huai, W. Jiang, and J. Gao, "Attack under disguise: An intelligent data poisoning attack mechanism in crowdsourcing," in *Proceedings of WWW*, 2018, pp. 13–22.

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

[26] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR, arXiv: 1411.1784*, 2014.

[27] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proceedings of NeurIPS*, 2017, pp. 5767–5777.

[28] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Proceedings of NeurIPS*, 2017, pp. 3520–3532.

[29] X. Zhang, X. Zhu, and S. J. Wright, "Training set debugging using trusted items," in *Proceedings of AAAI*, 2018.

[30] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart, "Sever: A robust meta-algorithm for stochastic optimization," in *Proceedings of ICML*, 2019, pp. 1596–1606.

[31] C. Zhu, W. R. Huang, A. Shafahi, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *Proceedings of ICML*, 2019, pp. 7614–7623.

[32] L. Muñoz-González, B. Pfitzner, M. Russo, J. Carnerero-Cano, and E. C. Lupu, "Poisoning attacks with generative adversarial nets," *CoRR, arXiv: 1906.07773*, 2019.

[33] 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 *Proceedings of NeurIPS*, 2018, pp. 6103–6113.

[34] A. Paudice, L. Muñoz-González, and E. C. Lupu, "Label sanitization against label flipping poisoning attacks," in *Proceedings of ECML PKDD*, 2018, pp. 5–15.

[35] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[36] T. Tran, T. Pham, G. Carneiro, L. Palmer, and I. Reid, "A bayesian data augmentation approach for learning deep models," in *Proceedings of NeurIPS*, 2017, pp. 2797–2806.

[37] M. A. Tanner, *Tools for statistical inference: observed data and data augmentation methods*. Springer, 2012.

[38] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of ICML*, 2017, pp. 214–223.

[39] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: A defence against trojan attacks on deep neural networks," in *Proceedings of ACSAC*, 2019, pp. 113–125.

[40] S. Udeshi, S. Peng, G. Woo, L. Loh, L. Rawshan, and S. Chattopadhyay, "Model agnostic defence against backdoor attacks in machine learning," *CoRR, arXiv: 1908.02203*, 2019.

[41] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proceedings of NeurIPS*, 2016, pp. 2234–2242.

[42] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proceedings of NeurIPS*, 2017, pp. 6626–6637.

[43] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *Journal of Multivariate Analysis*, vol. 12, no. 3, pp. 450–455, 1982.

Jian Chen (S'20) received B.S. degree from Hubei University of Technology in 2014 and the M.S. degree from Huazhong University of Science and Technology in 2018. He is currently working toward the Ph.D. degree in School of Electronic Information and Communications, Huazhong University of Science and Technology, China. His recent research interests focus on machine learning and data privacy. He is a student member of IEEE.

Xuxin Zhang received the B.E. degree from University of Electronic Science and Technology of China, China, in 2019. He is currently pursuing the M.S. degree in Electronics and Information Engineering at Huazhong University of Science and Technology, China. His research interests include data poisoning attacks and recommender systems.

Rui Zhang (M'12) is an Associate Professor in School of Computer Science and Technology at Wuhan University of Technology, China. She received her B.S. degree in Computer Application Technology from Jianghan Petroleum Institution, China in 2000, the M.A. degree and Ph.D. degree in Computer Science from Huazhong University of Science and Technology, China in 2003 and 2012, respectively. Her research interests include network analysis, mobile computing and data mining.

Chen Wang (S'10-M'13-SM'19) received the B.S. and Ph.D. degrees from the Department of Automation, Wuhan University, China, in 2008 and 2013, respectively. From 2013 to 2017, he was a postdoctoral research fellow in the Networked and Communication Systems Research Lab, Huazhong University of Science and Technology, China. Thereafter, he joined the faculty of Huazhong University of Science and Technology where he is currently an associate professor. His research interests are in the broad areas of wireless networking, Internet of Things, and mobile computing, with a recent focus on privacy issues in wireless and mobile systems. He is a senior member of IEEE and ACM.

Ling Liu (F'15) is currently a Professor at the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA. She directs the research programs at the Distributed Data Intensive Systems Lab (DiSL), examining various aspects of large-scale big data systems and analytics, including performance, availability, security, privacy, and trust. Her current research is sponsored primarily by the National Science Foundation and IBM. She has published over 300 international journal and conference articles. Dr. Liu was a recipient of the IEEE Computer Society Technical Achievement Award in 2012 and the Best Paper Award from numerous top venues, including ICDCS, WWW, IEEE Cloud, IEEE ICWS, and ACM/IEEE CCGrid. She served as the general chair and the PC chair for numerous IEEE and ACM conferences in big data, distributed computing, cloud computing, data engineering, very large databases, and the World Wide Web fields. She served as the Editor-in-Chief for the IEEE Transactions on Services Computing from 2013 to 2016. She is the Editor-in-Chief of the ACM Transactions on Internet Technology.