applied sciences

MDPI

Article

A Robust Countermeasures for Poisoning Attacks on Deep Neural Networks of Computer Interaction Systems

I-Hsien Liu [1], Jung-Shian Li [1], Yen-Chu Peng [1] and Chuan-Gang Liu [2,*]

[1] Department of Electrical Engineering, Institute of Computer and Communication Engineering, National Cheng Kung University, Tainan 701401, Taiwan; ihliu@cans.ee.ncku.edu.tw (I.-H.L.); jsli@mail.ncku.edu.tw (J.-S.L.); ycpeng@cans.ee.ncku.edu.tw (Y.-C.P.)
[2] Department of Artificial Intelligence and Computer Engineering, Chin-Yi University of Technology, Taichung 411030, Taiwan
* Correspondence: chgliu090210@gmail.com

Abstract: In recent years, human–computer interactions have begun to apply deep neural networks (DNNs), known as deep learning, to make them work more friendly. Nowadays, adversarial example attacks, poisoning attacks, and backdoor attacks are the typical attack examples for DNNs. In this paper, we focus on poisoning attacks and analyze three poisoning attacks on DNNs. We develop a countermeasure for poisoning attacks, which is Data Washing, an algorithm based on a denoising autoencoder. It can effectively alleviate the damages inflicted upon datasets caused by poisoning attacks. Furthermore, we also propose the Integrated Detection Algorithm (IDA) to detect various types of attacks. In our experiments, for Paralysis Attacks, Data Washing represents a significant improvement (0.5384) over accuracy increment, and can help IDA detect those attacks, while for Target Attacks, Data Washing makes it so that the false positive rate is reduced to just 1% and IDA can have a high accuracy detection rate of greater than 99%.

Keywords: poisoning attack; DNNs; detection algorithms

Citation: Liu, I.-H.; Li, J.-S.; Peng, Y.-C.; Liu, C.-G. A Robust Countermeasures for Poisoning Attacks on Deep Neural Networks of Computer Interaction Systems. *Appl. Sci.* **2022**, *12*, 7753. https://doi.org/10.3390/app12157753

Academic Editors: Charles Tijus, Teen-Hang Meen and Chun-Yen Chang

Received: 18 June 2022
Accepted: 5 July 2022
Published: 1 August 2022

1. Introduction

Machine learning (ML) has been widely used in many applications nowadays. Human–computer interaction systems also use such deep learning technology to make them work more friendly. However, there are also many threats to ML systems which are vulnerable to malicious attack. Recently, this issue has caught much attention. The authors in [1,2] discuss the data security of ML models. The correctness of the ML outcome depends on the integrity of its datasets. Fortunately, ML models are usually set up in a well-designed format, which causes the attacker difficulty in directly modifying or attacking ML models. However, the malicious attacker can easily manipulate the datasets that the ML models operate. For example, he or she can upload malicious datasets directly to the internet or can upload malicious data through Crowdsource systems [3]. The security threat to ML models is exacerbated by the fact that the datasets used by ML systems are typically very large and frequently contain data with high dimensions. Thus, it is not only relatively easy for an attacker to corrupt the dataset, but also extremely difficult for the user to detect such attacks.

Once ML model makes use of a poisoning dataset, it leads to serious misclassification errors, which even leads to complete paralysis of the system. Given the sensitive nature of many of today's ML applications, such errors may have disastrous effects. One study [4] surveys the challenges and countermeasures for adversarial attacks, and another study on medical datasets [5] showed that poisoning attacks can cause serious misclassification errors of cancer and disease samples. Furthermore, some studies [6,7] have shown that some attacks can induce misclassification errors while leaving the overall performance of the ML model intact. For example, in the case of facial recognition systems [8], the

attacker can pass the facial recognition test simply by wearing special glasses. Since such impersonation attacks are aimed at a particular individual, and have no wider effect on the whole system, they are almost impossible for the system administrator to detect.

Broadly speaking, adversarial attacks on ML systems can be classified as either backdoor attacks or poisoning attacks. In attacks of the former type, the model behaves as intended for clean inputs, but misbehaves in a manner predetermined by the attacker when fed with inputs stamped with a particular trigger. By contrast, in poisoning attacks, the adversary injects bad data into the training dataset in order to corrupt the learning process and distort the inference rules learned by the model. Poisoning attacks seriously affect the weight update process and training trend of the model, and thus have a significant effect on the correctness of the inference process. However, the literature contains very little information on the protection of deep neural networks (DNNs) against poisoning attacks [3,5,9]. Moreover, the literature lacks effective countermeasures for protecting DNNs against new attack algorithms. Hence, there is an urgent need to solve the problems of the poisoning and manipulation of data sets.

The main contributions in this study are as follows:

- This study proposes a Data Washing algorithm. It can recover the poisoned training dataset algorithm
- This study proposes Integrated Detection Algorithm (IDA) to resist the DNN poisoning attacks proposed in [6,7,10] and the Category Diverse attack proposed in the present study. the IDA algorithm provides an accurate means of detecting datasets containing abnormal data and thus provides effective protection against paralysis attacks, targeted attacks, and others.

The remainder of this paper is organized in the following sections. Section 2 describes background and related works in this paper, including an overview of deep learning, attack model on DNN, and countermeasures against those attacks. In Section 3, we introduce our proposed Data Washing algorithm and IDA algorithm in details. Next, we validate that our proposed algorithms can resist various attacks on different DNN models through several experiments. Finally, we conclude this paper.

2. Background and Related Works

2.1. Overview of Deep Learning

This section commences by describing the basic structures and operating principles of the three DNNs, VGG [11], GoogLeNet [12], and ResNet [13]. The general principles of the training process used to train such models are then introduced and discussed.

Deep Neural Networks (DNNs)

Neural networks usually have multiple layers, in which each layer is composed of multiple neurons. Deep neural networks (DNNs) mean neural networks having more than two layers; however, they usually have more than ten layers. Depending on the particular composition method, each layer may be either a fully connected layer or a convolutional layer. Ir-respective of the layer type, the neurons compute their outputs using a particular activation function, such as ReLU or sigmoid. Neural networks composed of fully connected layers are referred to as multiple layer perceptrons (MLPs), while those consisting mainly of convolutional layers are called convolutional neural networks (CNN). A deep neural network generally consists of both convolutional layers and fully connected layers, where the former layers account for most of the layers and are arranged at the front end of the model, while the smaller number of fully connected layers are arranged at the back end. The DNN model also may contain some special-purpose layers interspersed between the convolutional and fully connected layers. For example, pooling layers may be used to reduce the dimensions of the sample data, while dropout layers and batch normalization layers may be used to prevent overfitting.

As described above, three of the most common DNN architectures in use nowadays are VGG, GoogLeNet, and ResNet. The following paragraphs describe each architecture briefly.

1. VGG: A deep neural network with the front part of the network consisting of a cyclic arrangement of convolutional layers and pooling layers. This deep neural network exists in several variants, with different numbers of layers in each case. For instance, VGG16 owns 13 convolutional layers and three fully connected layers, and VGG19 has 16 convolutional layers and three fully connected layers.
2. GoogLeNet: GoogLeNet is a deeper neural network, having not only additional convolutional and pooling layers, but also the incorporation of inception modules. In its standard form, GoogLeNet owns 27 layers, where these layers include nine inception modules.
3. ResNet: ResNet inherits some of the features of AlexNet [14] and LeNett [15]. However, ResNet also incorporates additional residual blocks. In these blocks, the input is added to the output. Adding these residual modules makes ResNet possible to deepen the network, which can avoid the vanishing gradient problem. ResNet also owns several versions with different combinations of layers, including ResNet18, consisting of one convolutional layer, eight residual modules, and one fully connected layer.

2.2. Attacks on Machine Learning Models

Usually, the attacker tries his/her best to detect, as much as possible, the structure and parameters of the victim model. The attacker's knowledge of the target system can be categorized as either perfect or limited [16]. In the case of perfect knowledge, the attacker possesses all the information of the model, including its structure, weights, test dataset, and training data. By contrast, in the case of limited knowledge, the attacker possesses some, but not all, of the model information. Given perfect knowledge of the victim model, the attacker can initiate an attack directly by utilizing the victim's model parameters. However, given only limited knowledge, the attacker requires the use of an alternative technique to carry out the attack. One of the most common techniques is that of producing a surrogate model, which uses existing knowledge such as the structure and training dataset of the victim model to train an alternative model.

Attacks on ML models usually can be categorized into three kinds, namely, adversarial example attacks, poisoning attacks, and backdoor attacks. The distinction between them lies in the dataset targeted by the attacker. Table 1 shows a summary for the attack on ML models.

Table 1. Summary for the attack on ML models.

The Kinds of Attacks	Adversarial Example Attack	Poisoning Attacks	Backdoor Attacks
Attack target	The test dataset	The training dataset	The test dataset

The details of each type of attack are presented in the following.

2.2.1. Adversarial Example Attack

Adversarial examples, also known as evasion attacks [17], are aimed at the test dataset of the victim model. In such an attack, the victim trains a well-behaved model in accordance with the usual training procedure. However, the attacker then modifies the test dataset based on some knowledge of the victim model in order to deliberately produce errors during the testing process. Taking the Fast Gradient Sign (FGSM) attack [18] as an example, the attacker calculates the gradient of the loss function for the input, and then adds this value to the original input. The resulting increase in the loss function significantly increases the probability of misclassification errors (see Figure 1). Other attacks, such as DeepFool [19], aim to make the data of a certain category close to those of another category. In such attacks, the attacker calculates the gradient and resulting values iteratively and then manipulates the testing dataset accordingly, causing the resulting decision to approach and finally cross the decision boundary.

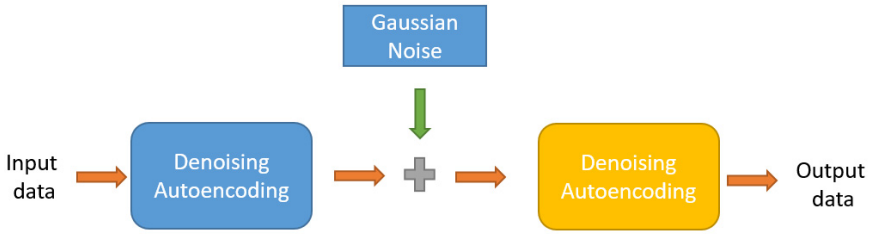

Figure 1. Flowchart of proposed Data Washing algorithm.

Adversarial examples can be classified as either targeted or nontargeted, where the former types of attacks cause the final category to be misclassified as a specified (incorrect) category, and the latter attacks aim simply to make the final category be misclassified. One of the most well-known targeted attacks is the Carlini and Wagner attack [20], originally designed to trick speech recognition systems.

Adversarial examples often have a certain degree of transferability. In other words, even though the attack is generated based on the weights, architecture, or parameters of one victim model, it may also be effective against other models. Black box attacks (i.e., attacks based on limited knowledge, see Table 1) can be achieved by training a surrogate model and then generating attack data based on the structure and parameters of this model. Furthermore, good transformability can greatly improve the attack ability and success probability of the attack, which is one of the goals pursued by many attack algorithms.

2.2.2. Poisoning Attacks

Poisoning attacks target the training dataset and aim to poison an existing dataset or propagate a new maliciously manipulated dataset for use by unsuspecting users. The machine learning model trained with a poisoning dataset will work abnormally. For non-DNNs, such attacks can paralyze the whole system with even a low poisoning rate. For example, in the poisoning attack proposed by Biggio, Battista et al. [21], the attacker simply maximized the hinge loss under the consideration of the information in the training dataset and caused a significant increase in the system error rate. For DNNs, by contrast, a relatively high poisoning rate is required to paralyze the system. In one such attack [6], the attacker tries to produce the output of the penultimate layer of one target ML model, which causes data collisions between the training dataset and target testing dataset.

Attack algorithms aimed at DNNs and non-DNNs, respectively, are somewhat different due to the different architectures of the network models in the two cases. For example, ML methods such as support vector machines (SVMs) and linear regression are usually optimized by applying the Karush–Kuhn–Tucker (KKT) conditions, whereas neural networks are generally optimized using a multiple iterative gradient descent method [16,21,22]. Thus, compared to adversarial examples, poisoning attacks have poorer transferability since the training characteristics of the model are usually very different for different model training settings.

2.2.3. Backdoor Attacks

Backdoor attacks add a signal to the test dataset of the victim network in order to cause the model to misclassify certain target data as a category which the attacker determines in advance. For pure inputs, the model behaves normally, and hence, backdoor attacks are notoriously difficult to detect. In practice, backdoor attacks can be implemented in two different modes. In the first mode, the user downloads and makes use of an apparently well-performing network model previously uploaded by the attacker. In the second mode, the attacker directly poisons the training dataset used by the victim to train their network model. For example, Chen, Xinyun et al. [8] proposed a backdoor attack against a facial recognition system, in which the target model was trained with a poisoned dataset such that the system could be tricked by individuals wearing special glasses.

2.3. The discussion on Poisoning Attacks

The present research focuses mainly on poisoning attacks. This attack renders the attack hard to detect, which may lead to serious damage to the learning model. Now, we give a deep discussion on this poisoning attacks. This section commences by describing several common poisoning attacks on non-DNNs and DNNs, respectively. A comparison is then made between poisoning attacks on DNNs and other forms of attacks.

2.3.1. Poisoning Attacks on Non-DNNs

The following discussions introduce three classic non-DNN poisoning attack algorithms targeted at SVM [21], linear regression [16], and multiple ML models [22], respectively.

- Poisoning Attacks against Support Vector Machines [21]

SVMs are usually developed using the hinge loss function. The main objective of poisoning attacks against SVMs is to maximize this hinge loss with the addition of a poisoned point (x_c, y_c) to a training dataset, as shown in Equation (1), where $f_{x_c}(x_k)$ is the output function of the SVM and y_k is the predicted label.

$$\max_{x_c} L(x_c) = \sum_{k=1}^{m} (1 - y_k f_{x_c}(x_k))_+ \tag{1}$$

In a poisoning attack, the attack algorithm produces a new input, $x_c^{(p)} = x_c^{(p-1)} + tu$, where u is a fabricated value. The optimization process of the SVM then obtains $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ under the KKT conditions. This outcome is then substituted into Equation (1) and partially differentiated with respect to u. Finally, the result is projected onto a unit vector and used to update u. The detailed attack process is described in [21], including the use of the KKT conditions and the LDU solutions, which is a gradient ascent method. (It is worth noting that w is a function of x, and is obtained from the KKT conditions rather than the gradient descent method).

- Poisoning Attacks against Linear Regression Models [17]

In linear regression models, the system is updated in accordance with a particular loss function and the goal of the attacker is to maximize this function, as shown in Equation (2), where $\ell(y_i, f(x_i))$ is the loss function and $\lambda\Omega(\mathrm{w})$ is the normalization term of linear regression.

$$\max_{x_c} L(x_c) = \frac{1}{m} \sum_{i=1}^{m} (\ell(y_i, f(x_i)) + \lambda\Omega(\mathrm{w})) \tag{2}$$

Through the KKT condition, we can obtain the weight and substitute it into the loss function. In a typical poisoning attack, techniques such as matrix manipulation are used to solve the partial differential and, having obtained this differential, the input value is updated in accordance with a particular operation to generate an attack. Generally speaking, the update method is similar to a gradient ascent method.

- Poisoning Attacks against SVM, Linear Regression, and Logistic Regression Models [22]

The authors in [22] designed an attack method that could be used against various ML models, including SVM, linear regression, and logistic regression. In contrast to the attack methods described above, the proposed method presets target weights of the victim network model and then tries to make the current weights approach these weights by updating the network model iteratively in accordance with Equation (3). As shown, the attack aims to minimize $O_A(D, \hat{\theta}_D)$ subject to $O_L(D, \theta)$ under the optimization conditions of $g(\theta)$ and $h(\theta)$, where D is the training data and $\hat{\theta}_D$ is the learned model.

$$\begin{gathered} \min_{D, \hat{\theta}_D} O_A(D, \hat{\theta}_D) \\ s.t. \hat{\theta}_D \in argmin_\theta O_L(D, \theta) \\ s.t. g(\theta) \leq 0, h(\theta) = 0 \end{gathered} \tag{3}$$

Equation (3) is a bilevel optimization problem. In particular, the KKT conditions are used to reduce the optimization problem to a single-layer problem, and this problem is then solved by updating the input data using methods such as projecting the gradient values. Having solved the equation, it is used to produce attack data to corrupt the target model.

2.3.2. Poisoning Attack on DNNs

Poisoning attacks on DNNs can be categorized as either target attacks or paralysis attacks, depending on their particular effect. In target attacks, the aim is to cause the model to misclassify certain specific targets without reducing the overall accuracy of the whole system. By contrast, in paralysis attacks, the aim is to degrade the entire system. The following discussions describe three of the most common poisoning attacks on DNNs, namely, TensorClog [10], a form of paralysis attack; and Feature Collisions [6] and Convex Polytope [7], both of which are forms of target attack.

- TensorClog [10]

When updating a neural network model, the update amount of each weight is determined according to the weight gradient of the loss function. That is,

$$dw = \nabla_w L(f\ (x, w),\ y) \tag{4}$$

The main idea of TensorClog is to make the weight gradient of the loss function approach zero by fabricating the input value in such a way as to achieve gradient descent of the loss function shown in Equation (5), where this loss function is the L2-norm of the weight gradient of the loss function plus a suppression term.

$$L_T\ (x',\ f\ ) = \frac{1}{2}\sum_{l=0}^{n} dw_l^2 + \lambda\left(x' - x\right)^2 \tag{5}$$

As with any paralysis attack method, the aim of TensorClog is to reduce the overall accuracy of the neural network model. In contrast to poisoning attacks against non-DNN models, in which the attackers only require a small poisoning rate that is enough to cause a significant effect, poisoning attacks against DNN models attempt to poison the entire dataset. Consequently, launching poisoning attacks is more challenging. However, due to the scale and complexity of DNN models, they are also more difficult for the system operator to detect and guard against.

- Feature Collisions [6]

In general, the objective of ML attack models is to disrupt the parameters of the neural network in such a way to cause the loss function to perform badly. However, the Feature Collisions attack method breaks this mold and aims instead to manipulate the output of the penultimate layer of the neural network in such a way to produce feature collisions between the training dataset and the test dataset.

Assume that x_j and y_j are data points in the training set and test set, respectively. Assume also that the output function of the penultimate layer is $f(x)$. The aim of the feature collision attack is to make $f(x_j)$ close to $f(y_j)$, such that the neural network model believes that the two data points are the same. Equation (6) shows the objective function of such an attack, where a suppression term is deliberately used to alleviate the modification effect.

$$L_T\ (x',\ y) = \| f(x') - f(y) \|_2^2 + \beta \| x' - x \|_2^2 \tag{6}$$

In other words, the feature collision attack is an optimization problem based on the target function, and is generally solved using the forward–backward–splitting method.

- Convex Polytope Attack [7]

The convex polytope attack is similar in concept to the feature collisions attack in that it also tries to control the output of the penultimate layer of the neural network.

Again, assume that x_j and y_j are data points in the training set and test set, respectively, and $f(x)$ is the output function of the penultimate layer. The underlying principle of the convex polytope attack is that as long as $f(y_j)$ is a convex combination of $f\left(x_{j,\ j=1,\ldots,n}\right)$, and all $f(x_j)$ are in the same category, $f(y_j)$ will also be classified as the same category. Note that convex combination is a special case of linear combination, in which all of the coefficients are non-negative and their sum is equal to one. In other words, the target point is enclosed in a polytope composed of multiple points. Compared with the feature collisions attack, the convex polytope attack aims to produce collisions between a single point and multiple points. Although it is computationally expensive and time consuming, it has good transferability.

2.3.3. Effect Comparisons for Poisoning Attacks on Non-DNN and DNN Models

Poisoning attacks perform very differently for neural networks and non-neural networks, respectively. From the perspective of model training and updating, many non-neural network ML algorithms update the model after running the entire dataset. However, neural networks are updated after running each small input batch. Thus, poisoning attacks can more easily succeed on non-neural networks than on neural networks. Muñoz-González, Luis et al. [23] compared the performance of several attack algorithms, including the simple label flipping model, on linear regression and three-layer convolutional neural networks, respectively, and found that the algorithms were far more effective against the former networks than the latter. Furthermore, DNNs are more effective than non-neural networks in resisting poisoning attacks conducted using random white noise. For example, Shen, Juncheng et al. [10] showed that paralyzing attack models implemented using small white noise interference are ineffective even if the entire dataset is poisoned. In summary, for low poison rates, both label flipping and high-level noise attacks are both ineffective when applied to DNNs. However, both attack methods are more effective when the poison rate exceeds 0.5, and can paralyze the neural network model if the poisoning rate reaches a sufficiently high level. For some non-neural network ML models such as SVM [24] and linear regression, the optimization solutions can be obtained relatively easily using optimization theory and an appropriate algorithm. For example, the KKT conditions can be applied to simplify the optimization problem, and the problem can then be solved using a quadratic programming (QP) technique. This approach is typically highly successful for nondeep neural networks, and allows the attacker to paralyze the entire system with only a small amount of data. However, it cannot be similarly applied to DNNs since the loss functions of such networks are nonconvex, and hence, the use of the KKT conditions to optimize the attack algorithm is infeasible. Furthermore, for DNNs, the number of weight parameters contained in each layer is extremely large. Thus, even if the optimization method described above can be used, the solution process is prohibitively time consuming. Yang, Chaofei et al. [25] tried to apply attack methods designed for non-neural networks to neural networks. The results showed that for such an approach to be successful, the data points would have to be poisoned before the start of each update step. However, this is infeasible since neural network models operate continuously and extremely rapidly, and it is thus almost impossible to obtain the current weights in real time. Moreover, the input data are randomly selected in the training process, and hence, there is no guarantee that the newly added poisoned point will actually be chosen.

Overall, poisoning attacks have a good success rate for nondeep neural networks, but perform less well when applied to DNNs. However, given a sufficiently high poisoning rate, they can nevertheless achieve paralysis of the target DNN and are almost imperceptible to the user.

2.4. Countermeasures against Attacks on Network Models

2.4.1. Countermeasures against Adversarial Examples

Countermeasures against adversarial examples can be divided into two different types, namely, those aimed at making the neural network model inherently more robust, and those

designed to detect or defend against such attacks [26]. Many methods have been proposed for enhancing the robustness of neural network models by modifying their architectures in some way. For example, Papernot, Nicolas et al. [27] used a neural network distillation method to condense a neural network with a large structure into an equivalent network with a small structure. Goodfellow, Ian J et al. [18] proposed a robust defense method known as adversarial training, in which the training dataset was purposely poisoned with a certain percentage of precomputed adversarial examples and the model was then trained accordingly. However, while this method can improve the robustness of the neural network model, it is effective only against attack algorithms sharing the current settings. That is, it is ineffective against the same attack with different attack parameter settings or different attack methods. Overall, a review of the adversarial training method shows that while increasing the loss function does not provide adequate protection against poisoning attacks, training the neural network model using a dataset containing malicious data can yield an effective improvement in the system robustness.

Metzen, Jan Hendrik et al. [28] proposed an intuitive detection method in which defense against adversarial examples was provided by training the neural network classifier using a dataset consisting of manipulated images and original images as abnormal and normal data, respectively. However, while the method performed well initially, its protection performance declined rapidly if the attacker subsequently reset the related attack parameters. Song, Yang et al. [29] proposed a detection and protection method based on PixelCNN, a generative model capable of inferring the likelihood of a particular pixel being selected through all the pixels selected before it. The experimental results revealed that the probability density distributions obtained from poisoned data and nonpoisoned data, respectively, were very different. Consequently, abnormalities could be identified by setting suitable threshold values, and noise-reduced images could then be generated by predicting the correct probability distribution. Gu, Shixiang et al. [30] used an autoencoder as a protection method by deliberately adding noise to interfere with the malicious signal of the adversarial example. However, the noise was found to interfere with the entire dataset, thereby degrading the overall performance of the training process. Meng, Dongyuu et al. [31] designed a detection and protection architecture based on the use of a self-encoder. The detection component of the architecture consisted of two stages, namely, calculating the reconstruction error and evaluating the degree of divergence, where the process of calculating the reconstruction error was based on the difference between the inputs and outputs of the autoencoder. However, the reconstruction error proved effective as a protection mechanism only when the malicious signal added to the training dataset was relatively strong. By contrast, the degree of convergence of the output data was far more sensitive to the presence of abnormal conditions, i.e., malicious signal injection, and therefore provided a more reliable indicator of anomalies in the neural network model.

2.4.2. Countermeasures against Poisoning Attacks

To the best of the authors' knowledge, existing countermeasures against poisoning attacks all depend on abnormality detection. That is, the system employs some form of detection algorithm to recognize abnormal data and then removes this data accordingly. In general, the algorithms and update methods of DNNs and non-DNNs are very different. Consequently, the corresponding attacks are also different. Therefore, in discussing existing countermeasures against poisoning attacks, this section commences by considering the detection and protection schemes proposed for non-DNNs and then moves on to consider the schemes proposed for DNNs. Jagielski, Matthew et al. [17] proposed an algorithm designated as TRIM for the defense of non-DNNs, in which the algorithm first found a subset D of the training data which resulted in a normal well-behaved ML model and then used this subset to calculate the loss of the model with all of the data point inputs in order to identify abnormalities. Steinhardt, Jacob et al. [32] tried to design a set of defense schemes that could be widely used to various machine learning models. In the proposed approach, the authors first generated a poisoned dataset by implementing a poisoning attack on their

network model in order to maximize the loss function. A threshold value below which the data points were assumed to be normal was then determined by computing the Euclidean distance between the averages of the loss functions with the normal points and all the poisoned points, respectively. This method is also called data sanitization; however, it cannot be extended to nonconvex models such as DNNs.

In 2016, Shen, Shiqi et al. [3] proposed a protection system for collaborative DNNs in which the distribution of the datasets uploaded by the users was checked using a K-means clustering approach. However, the proposed method was designed specifically for collaborative deep learning structures rather than general DNNs. Moreover, the efficacy of the proposed method was evaluated using only a simple attack method similar to label flipping. Subedar, Mahesh et al. [9] identified abnormalities in the inputs to a DNN by using an algorithm, designated as DeepFeatures, to fit the distribution of the features in the input data and then determining anomalies in the data based on the likelihood of the data points. Mozaffari-Kermani, Mehran et al. [3] implemented several multilayer neural networks and non-neural networks for datasets related to the medical field and proposed a method for detecting abnormalities in the testing data using a threshold value based on the CCI or Kappa statistics. The proposed method proved to be effective in countering simple label flipping poisoning attacks. However, its robustness toward more sophisticated attacks was not evaluated.

Overall, most existing countermeasures against poisoning attacks provide adequate protection against relatively simple attack algorithms designed to reduce the accuracy of the model. Generally speaking, such countermeasures firstly analyze the accuracy and loss function performance of the model and then detect abnormalities using a predefined threshold value.

3. System Structure

This section introduces system assumptions used in this study, attack models on DNN, and our proposed algorithms, including Data Washing and IDA algorithms. Then we commence system assumption in this section.

3.1. System Assumptions

The present study adopts three main assumptions, described as follows.

- The status (i.e., normal or poisoned) of the dataset input to the system is unknown. However, the labels of the dataset are well-confirmed, and hence, the attacker cannot launch label flipping attacks or randomly destroy the data for this dataset. Furthermore, the method used by the attacker to poison the dataset is ruled to be out of the scope of the present research.
- Model training is performed using frozen transfer learning with all of the layers frozen except for the last fully connected layer. It can greatly reduce the model training time.
- Four poisoning attacks are considered, namely, TensorClog, Feature Collisions, Convex Polytope, and a newly developed Category Diverse attack proposed in [33]. Note that the TensorClog and Category Diverse attacks are paralysis attacks, while the Feature Collisions and Convex Polytope attacks are target attacks.

3.2. Robust Denoising Algorithm and IDA Detection Algorithm

In addition to the Category Diverse attack model, this paper also proposes a robust denoising method, designated as Data Washing, for recovering a poisoned dataset by cleaning any data items containing malicious signals. The paper additionally proposes an integrated detection algorithm (IDA) designed to detect various attack scenarios. The details of the two algorithms are described in the following.

3.2.1. Data Washing Algorithm

In general, denoising autoencoders, which remove the malicious signal added by the attacker, achieve good protection against target attacks, but are less effective against

paralysis attacks. By contrast, the Data Washing algorithm proposed in this study provides an effective protection against both types of attacks. The details of the proposed algorithm are described as follows.

As shown in Figure 1, the data are first passed through a denoising autoencoder. A small amount of Gaussian noise (also called a lightweight Gaussian white noise) is then added to the data and the data are then passed through the autoencoder once again to obtain the restored data. Through denoising autoencoder, the algorithm effectively eliminates the malicious signal added by the attacker.

Intuitively, the accuracy of the denoising process should improve as the number of washing operations increases. However, the preliminary experimental results showed that the modification effect of the picture data became too large, and some of the pictures became blurred, when the washing process was repeated multiple times. Furthermore, limiting the modification effect to a certain maximum value was also found to reduce the cleaning effect. Hence, in processing the poisoned dataset, the number of washing operations was limited to just one.

The denoising autoencoder proposed in the present study comprises three encoding layers and one convolutional layer on the encoding side and three decoding layers and one convolutional layer on the decoding side. Although the main aim of the denoising autoencoder is to restore the signal with Gaussian noise to a clean signal, the autoencoder also can eliminate some non-Gaussian noise. Thus, the proposed algorithm provides an effective defense against not only specific attack algorithms, but also general unknown algorithms.

3.2.2. IDA Detection Algorithm

It is generally difficult to detect malicious signals in the datasets used by DNNs, and hence, determining whether or not a particular data point is abnormal is also extremely challenging. Anomaly detection algorithms, such as Isolation Forest and One Class SVM, do not perform well, and hence, it is difficult to distinguish between normal and contaminated values in the related cluster classification algorithms. Furthermore, analyzing the output of the penultimate layer of the model does not provide much help either. Finally, a single detection method fails to provide a robust defense capability against all attacks since the algorithms and effects of paralysis attacks and target attacks (or some other unknown attacks) are all different. To tackle this problem, the present study proposes an integrated detection algorithm to provide a robust detection performance for various attack scenarios, including paralysis attacks, target attacks, and others. The proposed algorithm, designated as IDA, consists of three stages, namely, detection of paralysis attacks, detection of target attacks, and detection of other attacks. Figure 2 shows the basic steps in the proposed IDA algorithm. Paralysis attacks have the ability to shut the entire system down, causing serious damage to the system's operation as a result. Consequently, the algorithm commences by detecting paralysis attacks. In the event that no such attack is detected, the algorithm proceeds to check for the presence of any other form of attack. If no obvious attack is detected, the algorithm makes a final check for a target attack. If the data are still determined to be normal, the algorithm terminates. If the IDA algorithm detects the attacks in the detection procedure, it claims the occurrence of a particular attack and ends this detection directly. Note that detecting a target attack requires the computation of a collision table, and hence, the detection process is deliberately performed as the final step in the detection algorithm in order to reduce unnecessary computational cost.

Detection of Paralysis Attacks: In normal situations without attacks, the accuracy rates of the models trained with normal and cleaned data, respectively, are approximately the same. However, for the model suffering from poisoned attacks, the accuracy rate obtained when using the Data Washing algorithm was much better than that obtained when the algorithm was not employed. Thus, in the IDA algorithm, the difference between the accuracy results obtained with and without data washing, respectively, was used to detect the occurrence of a paralysis attack. In particular, the DNN model was trained twice, once with uncleaned data and once with washed data. The two models were then applied

to the same testing dataset and the difference between the resulting accuracy rates (referred to as the differential accuracy rate) was computed in order to evaluate the occurrence of a paralysis attack in accordance with a certain preset threshold value. Figure 3 shows the detection of paralysis attacks in IDA algorithm.

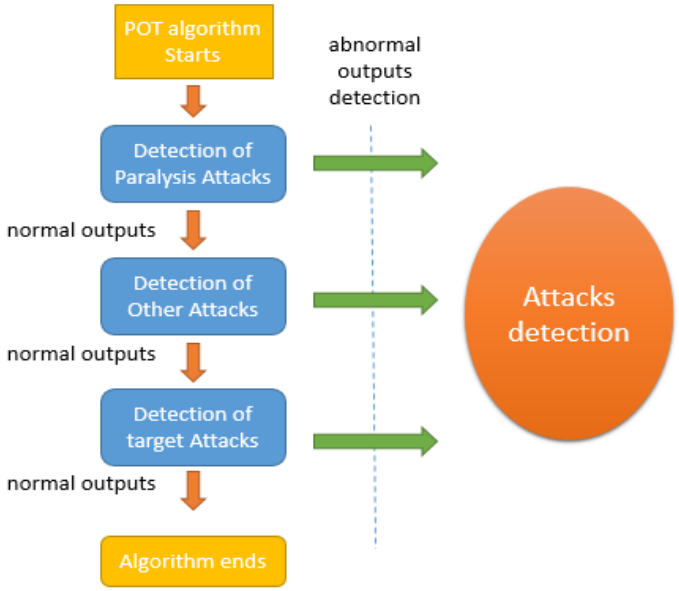

Figure 2. Flowchart of proposed IDA detection algorithm.

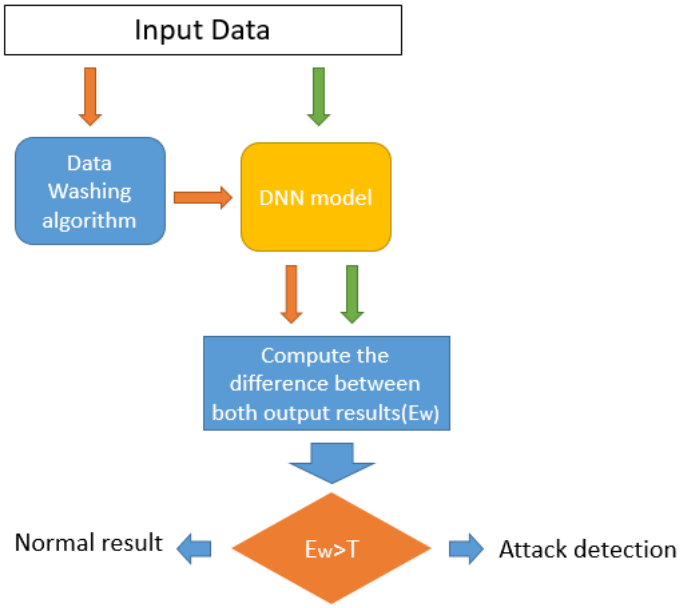

Figure 3. Flowchart showing detection of paralysis attacks in IDA algorithm.

Notably, the proposed detection method is applicable to different DNN models and different paralysis attack algorithms. However, it cannot detect the presence of single poisoned data points in the dataset. Thus, the IDA detection algorithm proceeds to check for any further signs of malicious activity in the dataset, as described in the following sections.

Detection of Other Attacks: Having confirmed the absence of a paralysis attack, the IDA algorithm checks for the occurrence of any other form of poisoning attack based on an inspection of the reconstruction error of the proposed denoising autoencoder. As shown in Figure 4, the training dataset is input to the denoising autoencoder and the L2-norm between the original picture data and the picture data exported from the denoising

autoencoder is computed. If the L2-norm is greater than a certain preset threshold value, the data are classified as abnormal and the algorithm terminates. Otherwise, the detection algorithm proceeds to check for the occurrence of a target attack.

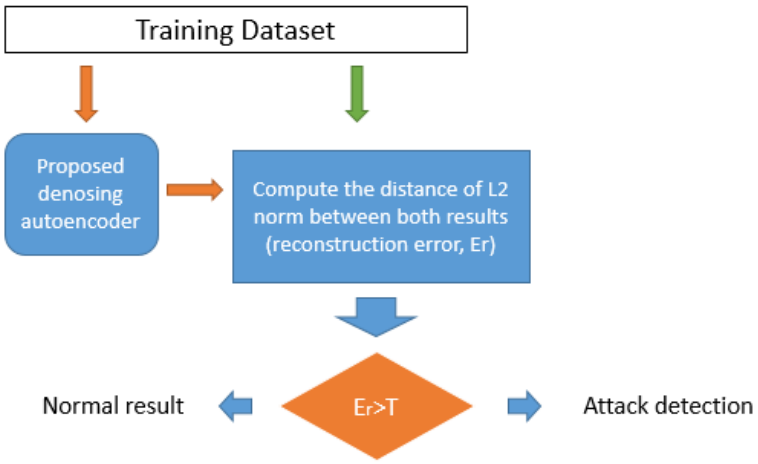

Figure 4. Flowchart showing detection of other attacks in IDA algorithm.

Detection of Target Attacks: The goal of target attacks is to make the outputs of the penultimate layer of the learning model be the same for the training data and testing data, respectively. In other words, the training dataset and target data appear to be the same to the ML model even though their resulting labels are actually different. That is, the ML model misclassifies the true label of the testing data point. Consequently, poisoned data points can be identified by comparing the outputs of the penultimate layer of the model for the testing data points with the outputs obtained for the training data points. As shown in Figure 5, the ML model is thus trained, as usual, using the training dataset, and the labels produced at the penultimate layer of the model are stored in a collision table. For each subsequent testing point, the output of the penultimate layer is compared with the entries in the collision table. If the output label is found to be very close to one of the values in the collision table, the data point is judged to be abnormal.

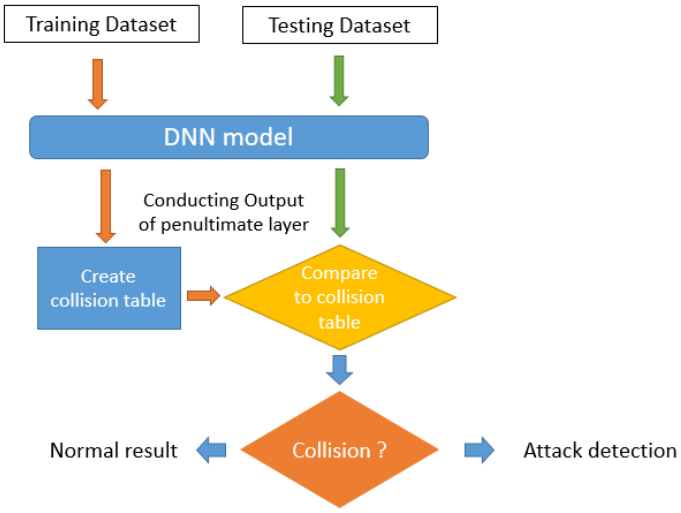

Figure 5. Flowchart showing detection of target attacks in IDA algorithm.

4. Experimental Results

The experiments were performed on a PC with Windows 10, an i7-7700 CPU, and 16 G of RAM. Model building was performed with Tensorflow [34] and PyTorch using Python as the programming language. Since the experiments involved DNNs, the calculations

were accelerated using a GPU GeForce GTX1060 (6 GB) graphics card. The training process was conducted for 40 epochs using a transfer learning approach and the model showing the highest accuracy for the test dataset between the 30th and 40th epochs was selected as the optimal model.

4.1. Data Washing Algorithm

The Data Washing algorithm was implemented using the denoising autoencoder proposed in [35]. As described in Section 3.2.1, a lightweight Gaussian white noise, which here equals 0.05 times Gaussian white noise, was added prior to the second denoising process. The autoencoder was trained using the Cifar100 [36] dataset. The performance of the Data Washing algorithm was evaluated for four different attack models, namely, the TensorClog and Category Diverse paralysis attack [33] models, and the Feature Collisions (FC) and Convex Polytope (CP) target attack models. The paralysis attacks were conducted using the parameters described in Section 4.1. The FC attacks [6] were implemented using the model results obtained after 100 training epochs, with parameter setting $\beta = 0.2 \times 2048^2/\dim_b{}^2$, a learning rate of 127,500, and a maximum number of iterations equal to 1500. Finally, the CP attacks [4] were implemented using the results obtained from 60 training epochs, namely, an initial value of $c^{(i)} = 1/10$, a maximum change limit of $\epsilon = 0.1$, a learning rate of 0.04, and a maximum number of iterations equal to 4000. CP attack produced five attack data items for each target.

4.1.1. Effectiveness of Data Washing Algorithm against Paralysis Attacks

The experiments commenced by observing the performance of the proposed Data Washing algorithm when applied to datasets corrupted by the TensorClog and Category Diverse attacks with a clip coefficient of 0.1 in both cases. Figure 6 presents the typical experimental results obtained for the two attack methods. Comparing the images in the first two rows, it is evident that the Data Washing algorithm achieves a significant reduction in the noise, ir-respective of the attack method applied. Moreover, comparing the images in the second and third rows, it is seen that the apparent effectiveness of the Data Washing algorithm increases with an increasing amount of noise in the image.

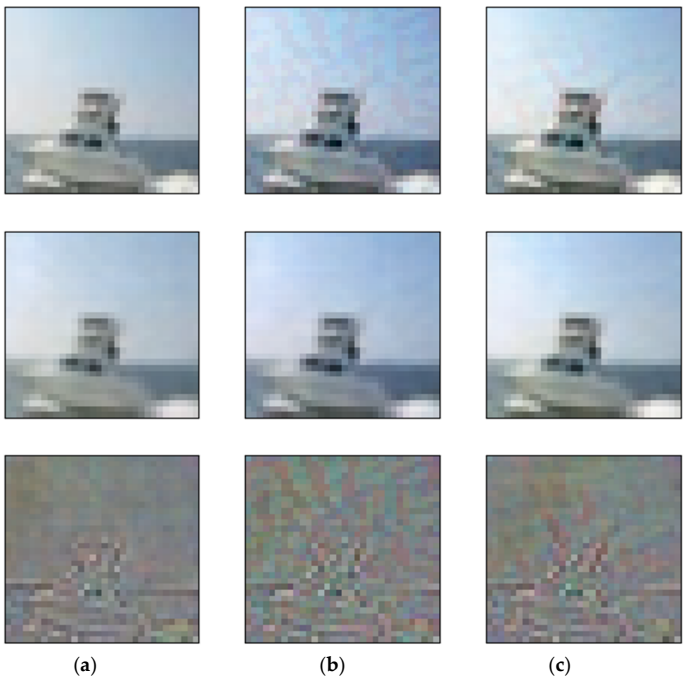

Figure 6. Effectiveness of Data Washing algorithm. First row: without Data Washing. Second row: with Data Washing. Third Row: enhanced Data Washing. (**a**) Without attack. (**b**) TensorClog. (**c**) Category Diverse.

Table 2 shows the effect of the Data Washing algorithm in improving the accuracy of the ResNet [17] model when subjected to TensorClog and Category Diverse paralysis attacks, respectively. In the case where the Data Washing algorithm is applied to the original clean dataset, the accuracy reduces by 0.0138. Following the TensorClog and Category Diverse attacks, the accuracy reduces to 0.6166 and 0.1660, respectively. In other words, the Category Diverse attack renders the dataset virtually unusable. However, following Data Washing, the accuracy rate is improved to 0.7044. While this result is not as good as that obtained for the original dataset (0.8316), it represents a significant improvement (0.5384) over that of the corrupted dataset.

Table 2. Effectiveness of Data Washing algorithm in mitigating effects of paralysis attacks on Resnet18 [10].

Statistic \ Algorithm	Original	TensorClog	Category Diverse
Accuracy	0.8316	0.6166	0.1660
Accuracy after DW	0.8178	0.7446	0.7044
Accuracy increment	−0.0138	0.1280	0.5384

Table 3 compares the effectiveness of the Data Washing algorithm with that of a simple denoising autoencoder (DAE). It is seen that for both attack models, the Data Washing algorithm results in a greater improvement in the ResNet performance than the conventional denoising autoencoder. In other words, the Data Washing algorithm provides a superior cleaning performance. The improvement is particularly apparent in the case of the Category Diverse attack.

Table 3. Effectiveness of Data Washing algorithm and denoising autoencoder in mitigating effects of paralysis attacks on Resnet18 [10].

Statistic \ Algorithm	Original	TensorClog	Category Diverse
Accuracy after DW	0.8178	0.7446	0.7044
Accuracy after DAE	0.8270	0.6936	0.5372
Accuracy increment	−0.0092	0.0510	0.1672

4.1.2. Effectiveness of Data Washing Algorithm against Target Attacks

Table 4 shows the effects of the FC and CP attacks on the ImageNet [37] and Cifar10 [36] datasets. Table 5 shows the equivalent results for the case where the Data Washing algorithm is applied. It is seen that the false positive rate is reduced to just 1% for the FC attack on the ImageNet dataset and 0% for the FC and CP attacks on the Cifar10 dataset.

Table 4. Effects of FC and CP target attacks without Data Washing algorithm.

Statistic \ Algorithm	FC Attack [6] ImageNet [37]	FC Attack [6] Cifar10 [36]	CP Attack [7] Cifar10 [36]
Accuracy	98.00%	99.50%	93.34%
Target Number	100	200	10
Misclassified	98	185	9
Target's False Positive Rate	98.00%	92.50%	90%

Table 5. Effects of FC and CP target attacks with Data Washing algorithm.

Statistic \ Algorithm	FC Attack [6] ImageNet [37]	FC Attack [6] Cifar10 [36]	CP Attack [7] Cifar10 [36]
Accuracy	99.00%	100%	93.33%
Target Number	100	200	10
Misclassified	1	0	0
Target's False Positive Rate	1.00%	0.00%	0.00%

4.2. Integrated Detection Algorithm

4.2.1. Effectiveness of IDA Detection Algorithm against Paralysis Attacks

In practice, the ability of the IDA algorithm to detect paralysis attacks depends on the effectiveness of the Data Washing algorithm in improving the model accuracy. Table 6 shows the accuracy improvements obtained by the Data Washing algorithm for the original dataset and the datasets attacked by the TensorClog and Category Diverse algorithms with two different clip rates (0.05 and 0.1). In the absence of any attack, the Data Washing algorithm results in almost no change in the accuracy. However, for both attack methods, the Data Washing algorithm results in a notable improvement in the accuracy. Hence, both attacks can be theoretically detected using the IDA algorithm, provided that an appropriate threshold value is set (see Figure 3). It should be noted that the IDA algorithm does not seek to identify every attacked data point in the dataset, but simply to determine whether or not the dataset has been attacked.

Table 6. Accuracy improvements obtained by Data Washing algorithm for TensorClog and Category Diverse paralysis attacks.

Statistic \ Algorithm	TensorClog Clip 0.05	Category Diverse Clip 0.05	TensorClog Clip 0.1	Category Diverse Clip 0.1
Accuracy Increment	0.0266	0.1990	0.0748	0.3690

4.2.2. Effectiveness of IDA Detection Algorithm against Target Attacks

The effectiveness of the IDA detection algorithm against target attacks was evaluated using the FC attack algorithm with 100 attack items and 984 normal data items in the ImageNet dataset and 200 attack items and 1800 normal data items in the Cifar10 dataset. As described in Section 3.2.2, the detection of target attacks in the IDA algorithm is performed based on the distance between the output values of the test items and those of the training items, as recorded in a collision table. Table 7 shows the experimental results for the mean and standard deviation values of the L2-norm distance in the FC attacks against the ImageNet and Cifar datasets. It is observed that the mean L2-norm distance for the normal data is very large, whereas that for the abnormal data is very small. Table 8 shows the corresponding detection statistics for the two datasets. For both datasets, the accuracy rate is greater than 99%. In other words, the ability of the IDA algorithm to detect the FC attack is confirmed for both datasets.

Table 7. L2-norm statistics between target test data and collision table data for FC attacks against ImageNet [37] and Cifar [10] datasets.

L2-Norm \ Algorithm	Original ImageNet	FC Attack ImageNet	Original Cifar10	FC Attack Cifar10
Mean	18.2909	2.9688	19.3545	3.6706
Standard deviation	2.6572	0.2671	2.7426	0.5807

Table 8. IDA detection algorithm performance for FC attacks against ImageNet and Cifar datasets. (Note that target attack detection threshold is 0.5.).

Statistic \ Algorithm	FC Attack ImageNet	FC Attack Cifar10
Accuracy	0.9991	0.9970
Precision	0.9901	1.0000
Recall	1.0000	0.9700
F1 Score	0.9950	0.9848

4.2.3. Effectiveness of IDA Detection Algorithm against Other Attacks

The CP attack is a target attack. However, due to the large-scale change it produces in the targeted dataset, it is used here to illustrate the performance of the IDA detection algorithm in detecting "other forms of attack" (see Section 3.2.2). The experiments considered 50 CP attacks, 25,000 Gaussian attacks, and 50,000 normal data items. As described in Section 3.2.2, the IDA algorithm detects "other attacks" based on an inspection of the reconstruction error. Table 9 shows the mean and standard deviation values of the reconstruction error for the present detection experiments. It is apparent that the CP attack produces a discernible mean reconstruction error between the normal data and the abnormal data. Table 10 shows the corresponding detection results obtained when using a reconstruction error threshold value of 2.5. It is evident that the IDA algorithm achieves a good detection performance for both attacks, i.e., CP and Gaussian attacks.

Table 9. Mean and standard deviation values of reconstruction error for attacks on Cifar10 dataset.

Statistic \ Algorithm	Original Cifar10	CP Attack Cifar10	0.1 Gaussian Cifar10
Mean	1.0319	2.9455	4.4168
Standard deviation	0.0485	0.0132	0.1720

Table 10. Detection performance of IDA algorithm for "other attacks". (Note that reconstruction error threshold is 2.5.).

Statistic \ Algorithm	CP Attack [4] Cifar10 [36]	Gaussian Cifar10 [36]
Accuracy	0.9978	1.0
Precision	0.8197	1.0
Recall	1.0	1.0
F1 Score	0.9009	1.0

5. Conclusions and Future Work

In this research, poisoning attacks on DNNs, including three new poisoning attacks, are studied and discussed deeply. We found that the poisoning attacks on non-DNN models may not perform well on DNN models. However, because DNNs are usually more complex and bigger than non-DNN models, the poisoning attacks on DNNs are usually hard to detect even if they should successfully attack DNN models under a relatively high poisoning rate. In this paper, we also discovered a new paralysis attack, known as the Category Diverse attack, which has a stronger paralysis effect than TensorClog. It leads to serious accuracy drops compared to TenserClog under the same truncation coefficient. We also found that the standard deviation standardization can make the attack more imperceptible. In order to resist these new poisoning attacks, we proposed Data Washing algorithms developed based on autoencoders and integrated detections algorithms. Data

Washing algorithms greatly reduce the effect of paralysis attacks and our experiments show that our algorithm makes the false positive rate of target attack drop to nearly zero. Integrated Detection algorithms can accurately detect three attacks through threshold-based schemes and collision tables, and our experiments also confirmed the excellent detection performance of our IDA algorithm. Based on the research conducted in this study, DNNs are expected to have a more powerful ability to resist poisoning attacks.

Author Contributions: Formal analysis, Y.-C.P.; Investigation, Y.-C.P. and C.-G.L.; Methodology, Y.-C.P. and C.-G.L.; Project administration, I.-H.L., J.-S.L. and Y.-C.P.; Writing—original draft, Y.-C.P.; Writing—review & editing, C.-G.L. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge the support of the Ministry of Science and Technology of the Taiwan under Grant MOST 108-2221-E-006-110-MY3, MOST 109-2221-E-041-001-, and MOST 111-2218-E-006-010-MBK.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The software code and data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Goldblum, M.; Tsipras, D.; Xie, C.; Chen, X.; Schwarzschild, A.; Song, D.; Madry, A.; Li, B.; Goldstein, T. Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. *IEEE Trans. Pattern Anal. Mach. Intell.* 2022; *in press*. [CrossRef] [PubMed]
2. Sun, G.; Cong, Y.; Dong, J.; Wang, Q.; Lyu, L.; Liu, J. Data Poisoning Attacks on Federated Machine Learning. *IEEE Internet Things J.* **2022**, *9*, 11365–11375. [CrossRef]
3. Shen, S.; Tople, S.; Saxena, P. Auror: Defending against poisoning attacks in collaborative deep learning systems. In Proceedings of the 32nd Annual Conference on Computer Security Applications, Los Angeles, CA, USA, 5–9 December 2016.
4. Ilahi, I.; Usama, M.; Qadir, J.; Janjua, M.U.; Al-Fuqaha, A.; Hoang, D.T.; Niyato, D. Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning. *IEEE Trans. Artif. Intell.* **2022**, *3*, 90–109. [CrossRef]
5. Mozaffari-Kermani, M.; Sur-Kolay, S.; Raghunathan, A.; Jha, N.K. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE J. Biomed. Health Inform.* **2014**, *19*, 1893–1905. [CrossRef] [PubMed]
6. Shafahi, A.; Huang, W.R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 2–8 December 2018.
7. Zhu, C.; Huang, W.R.; Li, H.; Taylor, G.; Studer, C.; Goldstein, T. Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019.
8. Chen, X.; Liu, C.; Li, B.; Lu, K.; Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv* **2017**. [CrossRef]
9. Subedar, M.; Ahuja, N.; Krishnan, R.; Ndiour, I.J.; Tickoo, O. Deep Probabilistic Models to Detect Data Poisoning. In Proceedings of the Fourth Workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, BC, Canada, 13 December 2019.
10. Shen, J.; Zhu, X.; Ma, D. TensorClog: An Imperceptible Poisoning Attack on Deep Neural Network Application. *IEEE Access* **2019**, *7*, 41498–41506. [CrossRef]
11. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
12. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erha, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
13. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2015.
14. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, CA, USA, 3–8 December 2012.
15. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
16. Xiao, H.; Biggio, B.; Brown, G.; Fumera, G.; Eckert, C.; Roli, F. Is feature selection secure against training data poisoning? In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015.

17. Jagielski, M.; Oprea, A.; Biggio, B.; Liu, C.; Nita-Rotaru, C.; Li, B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018.
18. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
19. Moosavi-Dezfooli, S.-M.; Fawzi, A.; Frossard, P. DeepFool: A simple and accurate method to fool deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
20. Carlini, N.; Wagner, D. Towards evaluating the robustness of neural networks. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–24 May 2017.
21. Biggio, B.; Nelson, B.; Laskov, P. Poisoning attacks against support vector machines. In Proceedings of the International Conference on Machine Learning, Edinburgh, UK, 26 June–1 July 2012.
22. Mei, S.; Zhu, X. Using machine teaching to identify optimal training-set attacks on machine learners. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–29 January 2015.
23. Muñoz-González, L.; Biggio, B.; Demontis, A.; Paudice, A.; Wongrassamee, V.; Lupu, E.C.; Roli, F. Towards poisoning of deep learning algorithms with back-gradient optimization. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Dallas, TX, USA, 3 November 2017.
24. Hearst, M.A. Support vector machines. *IEEE Intell. Syst. Appl.* **1998**, *13*, 18–28. [CrossRef]
25. Yang, C.; Wu, Q.; Li, H.; Chen, Y. Generative poisoning attack method against neural networks. *arXiv* **2017**. [CrossRef]
26. Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2805–2824. [CrossRef] [PubMed]
27. Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 23–25 May 2016.
28. Metzen, J.H.; Genewein, T.; Fischer, V.; Bischoff, B. On detecting adversarial perturbations. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
29. Song, Y.; Kim, T.; Nowozin, S.; Ermon, S.; Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
30. Gu, S.; Rigazio, L. Towards deep neural network architectures robust to adversarial examples. In Proceedings of the International Conference on Learning Representations Workshop, San Diego, CA, USA, 8 May 2015.
31. Meng, D.; Chen, H. Magnet: A two-pronged defense against adversarial examples. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 3 October–3 November 2017.
32. Steinhardt, J.; Koh, P.W.; Liang, P. Certified defenses for data poisoning attacks. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
33. Li, J.-S.; Peng, Y.-C.; Liu, I.-H.; Liu, C.-G. A New Poisoning Attacks on Deep Neural Networks. In Proceedings of the ICMHI 2022, Kyoto, Japan, 13–15 May 2022.
34. Tensorflow. Available online: https://www.tensorflow.org/ (accessed on 5 August 2020).
35. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008.
36. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Master's Thesis, University of Toronto, Toronto, ON, Canada, 8 April 2009.
37. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]