OPEN ACCESS
International Journal of Semantic Computing
Vol. 15, No. 2 (2021) 215–240

DOI: 10.1142/S1793351X21400043

World Scientific
www.worldscientific.com

A Countermeasure Method Using Poisonous Data Against Poisoning Attacks on IoT Machine Learning

Tomoki Chiba*, Yuichi Sei†, Yasuyuki Tahara‡ and Akihiko Ohsuga§

Graduate School of Informatics and Engineering
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan
**chiba.tomoki@ohsuga.lab.uec.ac.jp*
†*seiuny@uec.ac.jp*
‡*tahara@uec.ac.jp*
§*ohsuga@uec.ac.jp*

In the modern world, several areas of our lives can be improved, in the form of diverse additional dimensions, in terms of quality, by machine learning. When building machine learning models, open data are often used. Although this trend is on the rise, the monetary losses since the attacks on machine learning models are also rising. Preparation is, thus, believed to be indispensable in terms of embarking upon machine learning. In this field of endeavor, machine learning models may be compromised in various ways, including poisoning attacks. Assaults of this nature involve the incorporation of injurious data into the training data rendering the models to be substantively less accurate. The circumstances of every individual case will determine the degree to which the impairment due to such intrusions can lead to extensive disruption. A modus operandi is proffered in this research as a safeguard for machine learning models in the face of the poisoning menace, envisaging a milieu in which machine learning models make use of data that emanate from numerous sources. The information in question will be presented as training data, and the diversity of sources will constitute a barrier to poisoning attacks in such circumstances. Every source is evaluated separately, with the weight of each data component assessed in terms of its ability to affect the precision of the machine learning model. An appraisal is also conducted on the basis of the theoretical effect of the use of corrupt data as from each source. The extent to which the subgroup of data in question can undermine overall accuracy depends on the estimated data removal rate associated with each of the sources described above. The exclusion of such isolated data based on this figure ensures that the standard data will not be tainted. To evaluate the efficacy of our suggested preventive measure, we evaluated it in comparison with the well-known standard techniques to assess the degree to which the model was providing accurate conclusions in the wake of the change. It was demonstrated during this test that when the innovative mode of appraisal was applied, in circumstances in which 17% of the training data are corrupt, the degree of precision offered by the model is 89%, in contrast to the figure of 83% acquired through the traditional technique. The corrective technique suggested by us thus boosted the resilience of the model against harmful intrusion.

Keywords: Adversarial machine learning; security; defense; poisoning attack; detection.

*Corresponding author.

1. Introduction

Machine learning techniques have emerged as an adjunct in the human decision-making process by inferring patterns and features from large datasets. Systems using machine learning have become widespread, and have been used in a wide range of fields such as automated driving and supply chains in recent years [1, 2]. In this way, machine learning tends to be introduced into several systems, and it is playing an infrastructural role in society. Additionally, some methods for solving cybersecurity problems using machine learning technology have been proposed and are already in use [3– 7]. Hence, the security of systems and services using machine learning technology has become extremely necessary. If vulnerability exists in a machine learning system, it will harm many systems that use machine learning technology.

In this context, as the number of machine learning systems expands, it is becoming more and more valuable for an attacker to target machine learning systems to harm them. There are a wide variety of attack methods that target machine learning systems, one of which is not easily noticed even by the naked eye (as explained in Sec. 3.1). Thus in recent years, given that smart cities are becoming more and more advanced, if machine learning systems are easily introduced without taking countermeasures against such attack methods, there is a risk of large-scale accidents and incidents that may cause large amounts of damage and even harm human lives.

Building a machine learning model requires training data, and with the recent trend of using open data, open data are often used to build machine learning models [8]. For example, Linked Open Data (LOD) Cloud [9] has a knowledge base with a lot of data collected using LOD, and these data can be used (Fig. 1). However, malicious

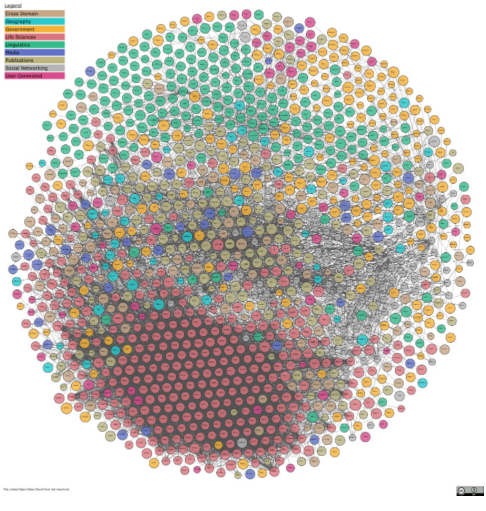


Fig. 1. The linked open data cloud insight center for data analytics.

data that are intentionally included in such open data will harm the machine learning models built using these data.

Recognizing these dangers are recognized, there have been several studies on attacks targeting machine learning systems and defense methods, and the knowledge of vulnerabilities of machine learning systems has been spread [10–12]. There are numerous reported attack methods against machine learning systems [13, 14], and in this study, we focus on the poisoning attack. A poisoning attack is an attack technique that reduces the prediction accuracy of a model by introducing poisonous data into the training data [15]. There are two main types of attack methods, namely, targeted attacks, in which poisonous data are generated and mixed into the training data in order to misclassify the target data to the intended label, and indiscriminate attacks, in which the entire input is misclassified to cause a denial of service.

The goal of this research is to reduce the loss of accuracy of machine learning models caused by poisoning attacks on machine learning.

Our contributions to this research are as follows:

- To protect the model from poisoning attacks, we propose a new approach that uses poisoning attacks in reverse.
- In verification experiments, we show that the proposed method is effective against poisoning attacks in the assumed environment.

This paper is an extended version of the conference publication [16]. The structure of this paper is as follows. In Sec. 2, the environment and the threat model assumed in this study are presented. In Sec. 3, related studies and their differences from this study are presented. The proposed method is described in Sec. 4 whereas the experimental setup and results obtained are presented in Sec. 5. In Sec. 6, a discussion on the proposed method is presented. Finally, the conclusions of this study are summarized in Sec. 7.

2. Problem Definition

2.1. *Assumed environments*

Machine learning systems are being used in a various situations. Especially in the IoT environment, there are many possible usage scenarios such as automated driving and automatic control of robots. In such a situation, if a machine learning system is attacked in the manner described in Sec. 1, the damage will not be small.

In this paper, we assume that there are multiple sources of data used for training models in an IoT environment. Figure 2 shows the assumed environment. As an example, we consider the acquisition of data from multiple IoT devices. In this section, we assume that a company constructs a recognition model of road signs for self-driving cars. This recognition model is installed in the self-driving car and recognizes road signs. Based on the recognition results, the system controls the speed

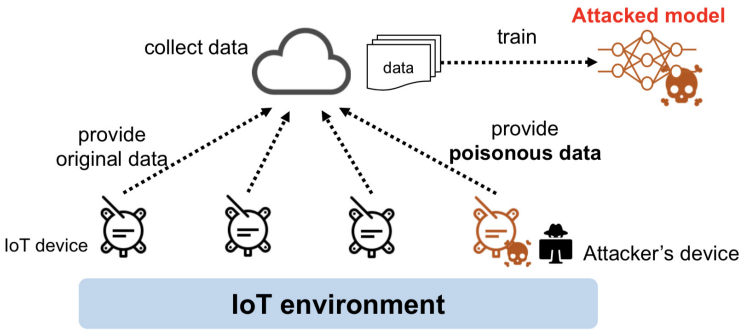


Fig. 2. Assumed environment.

of the self-driving car. In this scenario, the attacker intervenes in the training data collection process to conduct a poisoning attack.

Additionally, in this study, we assume that there are multiple people and IoT devices that have access to the datasets used for learning as described above and that there are entities with aggressive purposes lurking among them. In other words, some of the sources of the data to be used will provide poisonous data, which will be used to carry out poisoning attacks on the models constructed by the data collectors. In this paper, we will refer to the data providers as "devices".

In this research, we do not limit the structure of the attack target or the machine learning system to be protected. Because the method of the poisoning attack requires access to the training data used to build the model before training and using the model, it is relatively easy to assume that the target machine learning system is a machine learning system that learns online. However, the circumstances under which the attacker can access the training data depend on the scenario. For instance, if the attacker is inside the provider who builds the model and provides it as a service, the machine learning system may be poisoned even if it is not an online learning system.

2.2. *Threat model*

In this paper, we assume that the attacker performs a poisoning attack, i.e. the attacker aims to reduce the accuracy of the model by introducing poisonous data into the training data. In other words, we assume that the characteristic of the poisoning attack is not that the attacker intentionally misclassifies specific data, but that the attacker reduces the overall accuracy of the model. This objective is based on the motivation of the attack for the attacker, which is to cause a serious accident of the self-driving car in the case of the assumed environment in the previous section.

We also assume that the attacker has background knowledge of the learning algorithm and can extract or generate data from the distribution of the data used for training. This setting seems unrealistic in general, but in a real-world setting, proxy training data from the same distribution can be utilized [17].

As an example of a poisoning attack in a situation where training data are obtained from multiple devices, Fig. 3 shows an actual representation of Microsoft's chatbot "Tay". Tay is a program designed to generate human-like tweets by learning from actual conversations with humans. However, some people who talk to Tay intentionally learn offensive expressions, and the learned Tay tweets offensive expressions as well. Consequently, Microsoft decided to terminate the service about 16 h after its launch. In this paper, we assume that the data are collected from multiple devices, but the attacker can only access some of them.

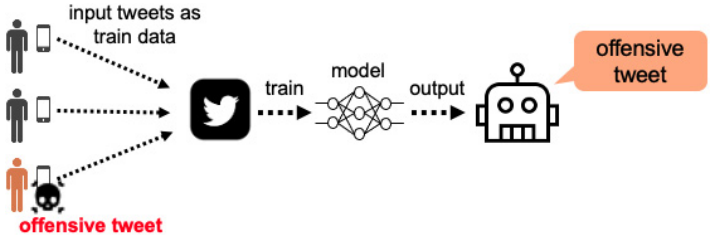


Fig. 3. An example of poisoning attack damage in a assumed environment.

3. Related Work

3.1. *Poisoning attack*

Since poisoning attacks are expected to occur during model training, it is assumed that the attacker has access to the training data in some way. For instance, an online tool for detecting malware in PDFs uses feedback from end-users to verify the detection results against the data provided [18]. In this context, the end-user has access to the training data of the classification system. Poisoning attack methods in cloud sensing data collection scenarios have also been reported [19] and the methods of accessing the training data vary from environment to environment. Figure 4 shows an example of a poisoning attack.

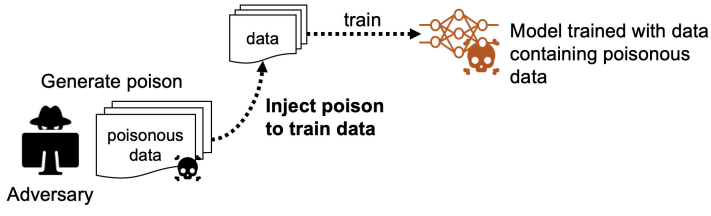


Fig. 4. An example of poisoning attack.

There is also an attack technique referred to as a backdoor attack, which is a derivative of the poisoning attack. This attack method involves hiding a backdoor key, a mark that does not affect the normal behavior of the model during training.

The data are manipulated and mixed into the training data so that the models behave unexpectedly only when a backdoor key is added to the input. Typically, a poisoning attack does not lead to the misidentification of a model by a specific backdoor key [20]. Thus, the general approach to protection against poisoning attacks is different from that against backdoor attacks.

The purpose of poisoning attacks can be divided into two main categories: availability degradation and integrity degradation. An attack on availability is aimed at indiscriminately decreasing the prediction accuracy of a model. An attack on integrity aims to cause misclassification of a specific sample while maintaining normal behavior [21].

The transferability of poisoning attacks has also been studied [22] by constructing a white-box environment to attack the same behavioral model that they generated [23]. Shafahi *et al.* showed that their proposed attack technique is effective against a target model even if the attacker cannot access the target model.

The damage assumed by a poisoning attack depends on the task of the model to be attacked. Reportedly, the definition of the feasibility of the attack remains unclear because of the difficulties in accessing and verifying the training data such as sanitization by the model producers and users.

3.2. *Defensive methods against poisoning attacks*

Various approaches for defending against poisoning attacks have been studied [24]. The main idea of protection methods against poisoning attacks so far is to detect specific data that adversely affect the accuracy of the model [20]. Other protection methods such as sanitizing outliers have also been reported [25]; however, several problems regarding this have been emphasized [26]. Thus, the defense method proposed in this paper aims to prevent poisoning attacks by detecting the data that adversely affect the accuracy of the model and the source of such data similarly to the former approach.

This paper assumes that training data are collected from multiple devices. Protection against poisoning attacks in the IoT environment has been studied [27]. Baracaldo *et al.* assumed that sensing data from multiple IoT devices are aggregated in the cloud and used as training data for the model. In this assumed environment, devices with malicious intent (poisonous devices) are included among the IoT devices that send data. The poisonous device performs a poisoning attack by sending poisonous data to the cloud.

They show that the "provenance data" linking the IoT devices and the data are robust against tampering and validate each IoT device by splitting the training data by its provenance data [28]. Because the environment assumed in this study is close to that of the above-mentioned study, the method of Baracaldo *et al.* is used as an existing method in this paper.

The Reject On Negative Impact (RONI) method by Nelson *et al.* [17], Nelson [29] is the basis of the defense approach of Baracaldo *et al.* [27]. This study proposes an

approach to exclude poisonous data from devices as a defense against spam filters. However, Balacaldo *et al.* compared the protection performance of their method with RONI and showed that their method is better than RONI. Thus, the existing methods compared in this paper are only those of Baracaldo *et al.*

3.3. *Other attacks on machine learning*

It has been emphasized that traditional machine learning algorithms do not consider adversarial settings and thus the performance of the classifier is significantly degraded when used in an adversarial setting [18].

Besides the poisoning attack described in Sec. 3.1, there are other attacks on machine learning models such as evasion attacks [30] and adversarial examples [31]. These attacks are the same as poisoning attacks in the sense that they target machine learning systems. Particularly, an evasion attack is an attack to manipulate a model so that an event that should be detected in an application such as anomaly detection cannot be detected.

Attacks to guess the parameters of the model after training have also been studied [32]. By guessing the parameters of a model in a service that uses an existing model, an attacker can generate a model that returns the same output as the attacked model for the input. This can be used to generate the adversarial example described above in a setting where the target model rejects a large amount of input. Membership inference attacks [33] also exist as an attack on privacy violations using machine learning systems.

4. Proposed Method

4.1. *Overview*

As shown in Sec. 1, our method aims to reduce the accuracy loss of the attacked model in the assumed environment. To achieve this, the method proposed in this paper removes data provided by devices that are suspected to be poisonous. Filtering in this paper means the process of excluding from the training data only the data provided by a device. Figure 5 presents the flowchart of the proposed method.

The proposed method can be roughly divided into three stages as follows:

(1) Construct a filtered train set and a filtered validation set, both of which are filtered out data obtained from each device. Generate a filtered model for each device using the filtered train set (Fig. 6). Also, the accuracy of the filtered and unfiltered models is compared using a filtered validation set. The results of the comparison are used to evaluate the impact of each device on the accuracy of the model.
(2) Perform the same operation as in (1), with the data of the devices as poisonous data. Define the value obtained from this comparison as "adversarial impact".

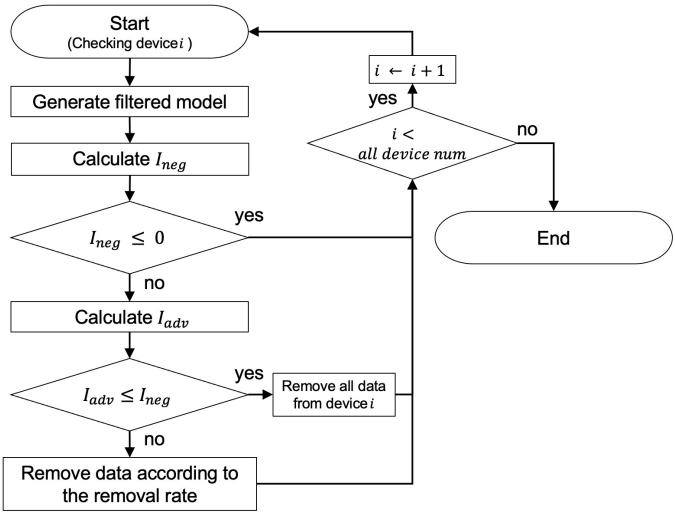


Fig. 5. Flowchart of the proposed method.

(3) Based on the values defined in (1) and (2), the confidence level for judging a device as poison is derived. The rate of removal of data according to the confidence level is defined for each device.

Repeat the above process for as many devices as there are.

Algorithm 1 shows the algorithm of the proposed method. Table 1 shows the definitions of the symbols used in the algorithm. The proposed method receives Dclfdevice_info as input; device_info is a list of device identifiers that represent the source of each data. First, the original dataset D is divided into training and validation data (D_{tr} and D_{val}) in the method. The validation data generated here is used as test data to check the accuracy of the model in the method. Ind_{tr} and Ind_{val} are lists of indices that represent the correspondence of D_{tr} and D_{val} in D, respectively. Next, from D_{tr} and D_{val}, the data are divided into data for each device using device_info and Ind_{tr} and Ind_{val}, respectively. In this method, it is necessary to process the data separately for each device to verify whether the data are normal or not for each device. From here, we start processing the data for each device. The train_segment and the valid_segment refer to the training data and the validation data provided by device i, respectively. In this section, we first define the filtered data by the filtering process filtering(). In the filtering process, we simply remove the data of device i from D_{tr} and D_{val}, respectively. The process up to this point corresponds to the process shown in Fig. 6.

In the "for statement", we calculate the negative impact as described in Sec. 4.2 and the adversarial impact as described in Sec. 4.3. Using these values, the function remove_data() removes as much data as it determines should be removed. Details of the function remove_data() are given in Sec. 4.4. By removing the data that are

Algorithm 1. Proposed method

Input: D, clf, device_info
Output: $D_{\text{protected}}$
M_{unfil}, M_{fil}, $M_{\text{adv}} \leftarrow$ clf.copy()
D_{tr}, $D_{\text{val}} \leftarrow$ train_valid_segment(D)
/* Store train and valid index list */
$\text{Ind}_{\text{tr}} \leftarrow$ List of index D_{tr} in D
$\text{Ind}_{\text{val}} \leftarrow$ List of index D_{val} in D
/* Split data by device */
train_segments $\leftarrow$ segment_by_class(D_{tr}, device_info[Ind_{tr}])
valid_segments $\leftarrow$ segment_by_class(D_{val}, device_info[Ind_{val}])
/* Validate each device and determine which data to remove */
for device_$i$dx, (train_segment, valid_segment) in zip(train_segments, valid_segments) **do**
    $i \leftarrow$ device_idx
    n_data $\leftarrow$ |train_segment| + |valid_segment|
    /* Generate filtered data */
    $\text{filtered\_train}_i \leftarrow$ filtering(D_{tr}, train_segment)
    $\text{filtered\_valid}_i \leftarrow$ filtering(D_{val}, valid_segment)
    /* Train each model */
    M_{unfil}.fit(D_{tr})
    M_{fil}.fit(filtered_train_i)
    /* Calculate I_{neg} and I_{adv} */
    $I_{\text{neg}} \leftarrow \text{calc\_neg}(\text{filtered\_valid}_i, M_{\text{unfil}}, M_{\text{fil}})$
    $I_{\text{adv}} \leftarrow \text{calc\_adv}(\text{train\_segment}, \text{filtered\_train}_i, \text{filtered\_valid}_i, M_{\text{fil}}, M_{\text{adv}})$
    /* Remove poisonous data */
    D_{tr}, $D_{\text{val}} \leftarrow$ remove_data(D_{tr}, D_{val}, filtered_train_i, filtered_valid_i, I_{neg}, I_{adv}, i, n_data)
end for
$D_{\text{protected}} \leftarrow \text{append}(D_{\text{tr}}, D_{\text{val}})$
return $D_{\text{protected}}$

Table 1. Notation.

Notation	Description
D	Dataset before protecting
clf	Classifier before training
device_info	Identifier of the device that corresponds to the data
$D_{\text{protected}}$	Dataset after protecting by the proposed method

considered poisonous for each device from D_{tr} and D_{val}, D_{tr} and D_{val} become the dataset after the protection by the proposed method when the for statement is processed for all devices. Finally, the two datasets are concatenated and $D_{\text{protected}}$ is output. The process up to this point corresponds to the process shown in Fig. 7.

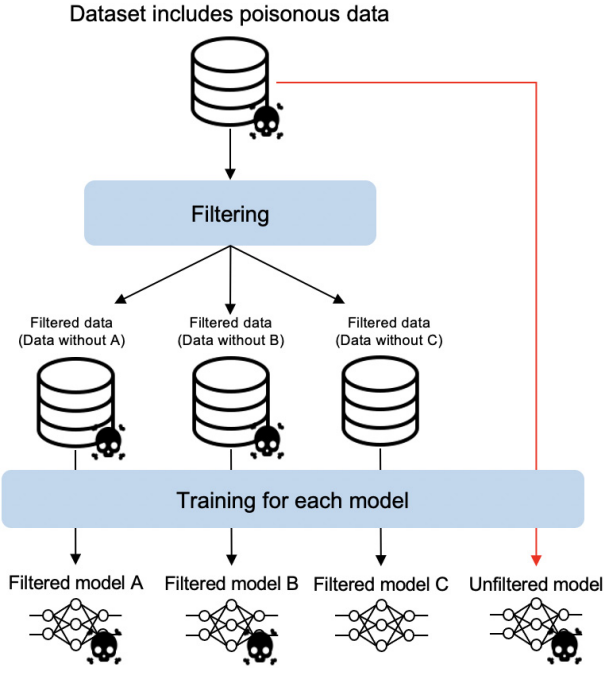


Fig. 6. Flow of generating filtered model.

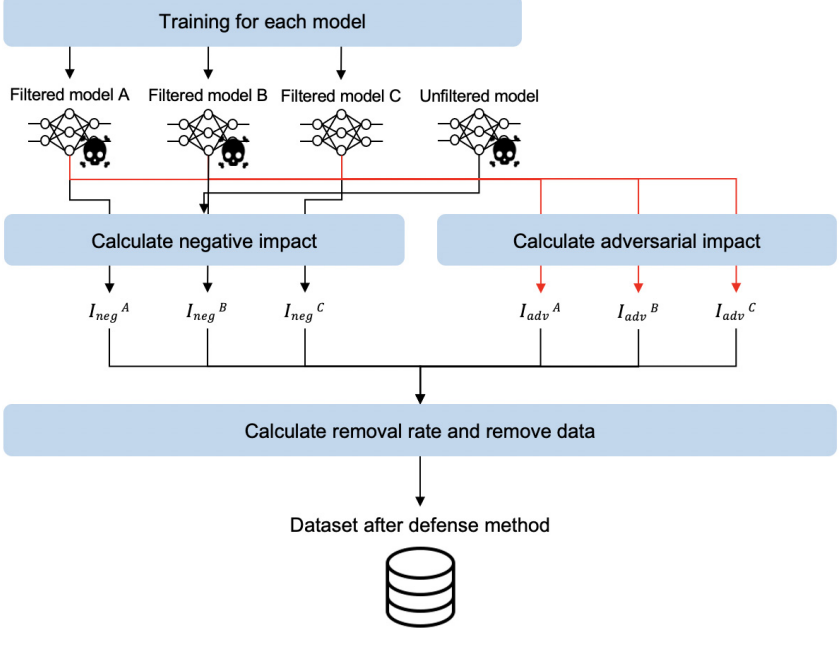


Fig. 7. Process flow from filtering to post-protection data generation.

4.2. *Negative impact*

As shown in Sec. 3.1, a poisoning attack is an attack that reduces the accuracy of a model by injecting poisonous data into the training data. In other words, when it comes to searching for and removing poisonous data from a dataset that contains poisonous data, one of the criteria to be considered is the "effect on the accuracy of the original model".

In this method, we use the concept of RONI [17] to evaluate the impact on the accuracy of the model for the training data of each device. The degree of influence on the accuracy of the model I_{neg} is defined by taking the difference between the accuracy of the unfiltered model with respect to the filtered validation set (unfiltered score) and the accuracy of the filtered model with respect to the filtered validation set (filtered score).

$$I_{\text{neg}} = \text{filtered score} - \text{unfiltered score}. \tag{1}$$

The higher the value of I_{neg}, the more negative is the effect on the model. If the value of Eq. (1) is less than or equal to 0, it implies that the effect on the accuracy of the model is at least not bad. Conversely, if the value of I_{neg} is greater than 0, it implies that it has at least a negative effect on the accuracy of the model. The proposed method in this paper uses the value of I_{neg} as one of its decision criteria. The decision criteria for each device are described in Sec. 4.4.

Algorithm 2 shows the function calc_neg for calculating I_{neg}. The input to the function requires the validation data filtered_valid$_i$, which is used to calculate the accuracy of the models, and each model M_{unfil}, M_{fil}. The algorithm simply calculates the accuracy of the two input models using the input validation data and stores the accuracy in unfiltered_score and filtered_score, respectively. The process of calculating the accuracy here corresponds to the function score() in the algorithm. Then, store the result of the expression (1) in I_{neg} and return it as the output of the function calc_neg.

Algorithm 2. calc_neg

1: **Input:** filtered_valid$_i$, M_{unfil}, M_{fil}
2: **Output:** I_{neg}
3: unfiltered_score $\leftarrow$ score(M_{unfil}, filtered_valid$_i$)
4: filtered_score $\leftarrow$ score(M_{fil}, filtered_valid$_i$)
5: $I_{\text{neg}} \leftarrow$ filtered_score-unfiltered_score
6: **return** I_{neg}

4.3. *Adversarial impact*

In this section, we explain the definition of adversarial impact. In short, this is the negative impact of a device being poisonous. We define adversarial score as the

unfiltered score for the device that is assumed to be poisonous, and adversarial impact I_{adv} is defined by the following equation:

$$I_{\text{adv}} = \text{filtered score} - \text{adversarial score}. \tag{2}$$

In this study, it is assumed that further harmful editing of the originally poisonous data will preserve the effects of harmful effects on the model. This effect is based on the results of the preliminary experiments we performed.

When a normal device is inspected, the ratio of poisonous data to total data simply increases when the data of the device are poison. Conversely, because of the assumptions mentioned above the total number of poisonous data does not increase when the poisonous devices are checked because the data are poisonous originally. This means that the adversarial score is low for normal devices and relatively high for poisonous devices. Additionally, comparing the unfiltered score and the adversarial score, the adversarial score is high for normal devices and about the same for poisonous devices. Therefore, I_{neg} and I_{adv} are almost equal for poisonous devices.

The function calc_adv for calculating I_{adv} is shown in Algorithm 3. It requires as input the train_segment data of the target device, the filtered_train$_i$ and filtered_valid_i training and validation data after filtering, and the model M_{fil} trained with filtered_train$_i$ and the model M_{adv} before training. Unlike the algorithm to calculate I_{neg}, the function calc_adv must generate the poisonous data first. Therefore, the function generate_poison is used to generate the poisonous data using the train_segment. Since generate_poison is arbitrarily determined by the user of the defense system (the proposed method), the specific algorithm depends on the attack method used.

Algorithm 3. calc_adv

Input: train_segment, filtered_train$_i$, filtered_valid$_i$, M_{fil}, M_{adv}
Output: I_{adv}
$D^p \leftarrow$ generate_poison(train_segment)
adversarial_data $\leftarrow$ append(filtered_train$_i$, D^p)
M_{adv}.fit(adversarial_data)
filtered_score $\leftarrow$ score(M_{fil}, filtered_valid$_i$)
adversarial_score $\leftarrow$ score(M_{adv}, filtered_valid$_i$)
$I_{\text{adv}} \leftarrow$ filtered_score $-$ adversarial_score
return I_{adv}

The generated poisonous data are stored in D^p, and then the data obtained by combining filtered_train$_i$ and D^p are stored in adversarial_data. This means that adversarial_data can virtually reproduce the unfiltered data when the device i is assumed to be a poisonous device. Then, the accuracy of the filtered_score and the adversarial_score is calculated by the function score(). Calculate the calculated

accuracy based on formula (2), store the result in I_{adv}, and return I_{adv} as the output of the function calc_adv.

4.4. *Remove data*

Finally, the percentage of data to be removed is determined using the relationship between the impact on the accuracy of the model I_{neg} defined in Sec. 4.2 and the value of I_{adv} defined in Sec. 4.3.

The formula for the data removal rate is shown as follows:

$$\text{removal rate} = \begin{cases} 0.0 & \text{if } I_{\text{neg}} \leq 0; \\ \dfrac{I_{\text{neg}}}{I_{\text{adv}}} & \text{if } 0 < I_{\text{neg}} < I_{\text{adv}}; \\ 1.0 & \text{else;} \end{cases} \tag{3}$$

In the proposed method in this paper, if the negative impact value I_{neg} is less than or equal to 0, the device is judged to be a normal device and the corresponding data are not removed. This strategy is based on the definition of negative impact, which states that the data of a device with a value less than 0 have at least no negative impact on other data (i.e. it does not deteriorate the accuracy). If I_{neg} is greater than 0 and less than I_{adv}, the removal rate is calculated by dividing I_{neg} by I_{adv}. When I_{neg} is greater than 0, the data are subject to removal because it has a negative impact on the accuracy of the model. However, in this research, instead of removing all the data of the devices targeted for removal, the proposed method calculates an independent removal rate for each device and removes the data. Since the value of I_{adv} is the value of I_{neg} under the assumption that all data of the target device is poisonous, the closer I_{neg} is to I_{adv}, the more likely it is that the data from that device is poisonous. Thus, the removal rate is determined according to the proximity to the value of I_{adv}. If I_{neg} exceeds the value of I_{adv}, it is considered to be worse than the value assumed to be poisonous. Thus, the removal rate is set to 1.0 and all the data of the target device is subject to removal.

The data removal process also removes data from all data that will be used to verify the next device. In other words, the data that have been determined to be poisonous will not be present in the data that are used after the determination, and hence, the data that have been determined to be poisonous will not affect the determination for the device that is subsequently verified. Algorithm 3 shows the algorithm to calculate the data removal rate. In the function remove_data, the training and validation data D_{tr}, D_{val}; the filtered data filtered_train$_i$, filtered_valid$_i$; the negative impact I_{neg}; the adversarial impact I_{adv}; and the number of data of the target device, n_data, as input values.

The process of this function involves conditional branching according to Eq. (3), in which the removal rate is determined. The first conditional branch of the algorithm determines if I_{neg} exceeds the value of I_{adv}, i.e. if it is the bottom case in Eq. (3). If this condition is met, the data removal rate is 100%. Thus, by replacing the

Algorithm 4. remove_data

Input: D_{tr}, D_{val}, filtered_train$_i$, filtered_valid$_i$, I_{neg}, I_{adv}, n_data
Output: D_{tr}, D_{val}
if $I_{\mathrm{neg}} > 0 \wedge I_{\mathrm{neg}} \geq I_{\mathrm{adv}}$ **then**
    /* Remove all data obtained from device i */
    $D_{\mathrm{tr}} \leftarrow$ filtered_data$_i$
    $D_{\mathrm{val}} \leftarrow$ filtered_valid$_i$
else if $I_{\mathrm{neg}} > 0$ **then**
    /* Remove some of the data obtained from device i */
    removal_rate $\leftarrow I_{\mathrm{neg}}/I_{\mathrm{adv}}$
    n_delete $\leftarrow n$_data $*$ removal_rate
    n_delete_train, n_delete_valid $\leftarrow n$_delete$/2$
    $D_{\mathrm{tr}} \leftarrow$ delete$(D_{\mathrm{tr}}, n$_delete_train$)$
    $D_{\mathrm{val}} \leftarrow$ delete$(D_{\mathrm{val}}, n$_delete_valid$)$
end if
return D_{tr}, D_{val}

training and validation data with the filtered data, the data of the target device is removed. In the second conditional branch, our method determines whether I_{neg} is greater than 0 and not in the previous case, i.e. if it is the middle case in Eq. (3). If this condition is met, the removal rate is calculated by dividing I_{neg} by I_{adv} according to the definition of the removal rate. Once the removal rate is calculated, determine the number of data to be removed for training and verification, using the number of data in the target device, n_data. Then, the original data to be removed and the number of data to be removed are passed to the data removal function delete(), respectively, to perform data removal. Finally, it returns the training and validation data D_{tr} and D_{val}, respectively, after the deletion. If neither of the above two cases applies, I_{neg} is less than or equal to 0, then it is judged to be normal and no data removal is performed.

5. Evaluation

We conducted experiments to verify the effectiveness of the defense method against the poisoning attack shown in this paper.

The experiments compare the accuracy of the following four models:

- Model trained on data without any poisonous data (perfect defense).
- Model trained without removing poisonous data (no defense).
- Model defended by the proposed method (proposal).
- Model defended by the existing method (existing).

The attack methods used in this study and the existing ones were implemented based on the open source code in [34]. To shorten the experiment time, we prepared and

used the poisonous data generated by the attack methods beforehand. The number of poisonous devices was varied, and the accuracy of the post-protection model was compared against the percentage of poisonous data. The threshold values of the existing methods for determining poisonous devices were set to 0.0 and 0.2 (the default value set in the open source [34]). All experimental results in this paper are based on an average of 10 times.

In this study, we conducted experiments with each of the following datasets:

- MNIST
- Synthetic data
- Environmental sensor telemetry data
- Air quality data

The experimental setup and results for each dataset are described in the following sections.

5.1. *MNIST*

5.1.1. *Setting*

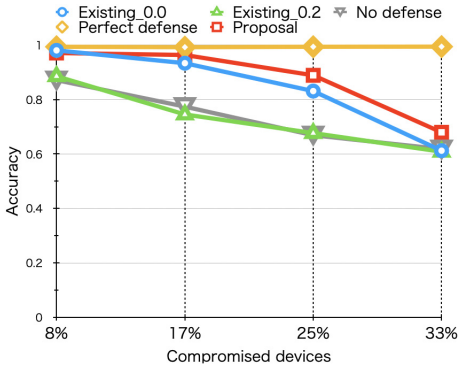
In this experiment, we used handwritten text MNIST. Originally, there are 10 classes from 0 to 9, but we used a binary classification task of 0 and 4 as in the experimental setup of Biggio *et al.* We also generated poisonous data using the attack method of Biggio *et al.* [15]. This method is an algorithm for generating poisonous data in a poisoning attack targeting SVM. The total number of devices is 10 and the data for each device are set to 100.

5.1.2. *Results*

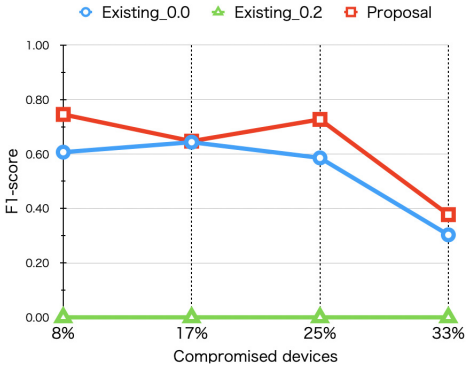
Figure 8(a) shows the accuracy graph of the model after the defense. As shown in Fig. 8(a), both the existing and proposed methods significantly exceed the accuracy of the model without protection. Also, the accuracy of the models protected by the proposed method is generally higher than that of the existing method. In the case where the ratio of poisonous data is 8%, the accuracy of the existing method is slightly higher than that of the proposed method. In this case, the accuracy of the existing method is more than 0.98 and it is difficult to improve further.

Additionally, the difference between the proposed method and no defense is smaller for a setting with 33% of poisonous data than the other conditions. This is because the removal rate of the data is reduced due to high rate of poisonous data. As a result, the performance of the proposed protection method also reduced.

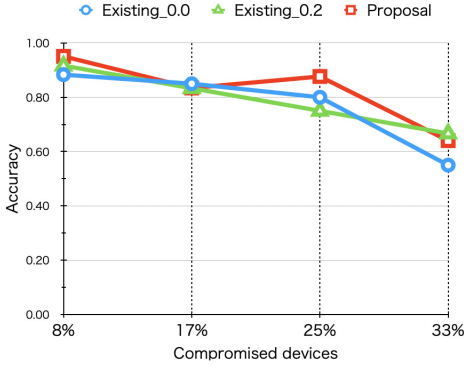
Figures 8(b) and 8(c) present the results of the $f1$-score and accuracy of the poisonous data detection rate, respectively. Regarding $f1$-score, except for 17%, the value obtained using the proposed method is higher than that of the existing method. In general, the accuracy of the proposed method is higher than that of the other methods.



(a) Accuracy of the model after each defenses



(b) Detection rate of poisonous data (F1-score)



(c) Detection rate of poisonous data (Accuracy)

Fig. 8. Results of model accuracy and poisonous data detection rate on MNIST.

5.2. *Synthetic data*

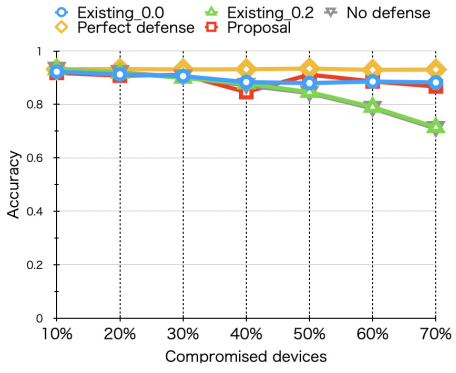
5.2.1. *Setting*

Here, we used the dataset and the attack method used in [35]. In this attack algorithm, the attack strength parameter (attack_parameter) is a number between 0 and 1. The strength of the attack here indicates how far from the original distribution the poisonous data are generated, with the stronger the data, the further from the original distribution it is.

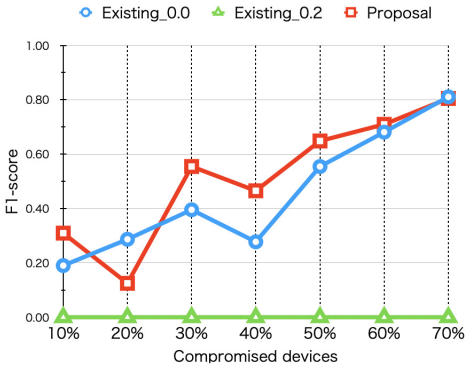
The experimental setup is the same as in the experiments in Sec. 5.1. The total number of devices set to 10, and the number of data for each device is set to 100.

5.2.2. *Results*

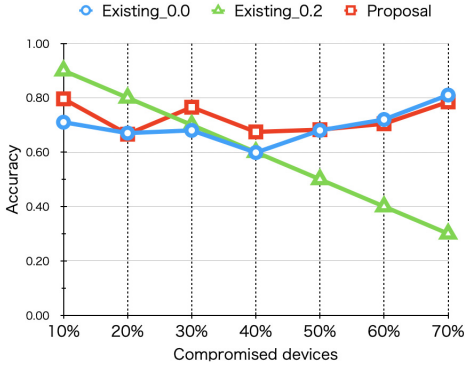
The results of accuracy of the model after applying each defensive methods are shown in Figs. 9(a), 10(a) and 11(a), respectively. The accuracy of the models trained



(a) Accuracy of the model after each defenses



(b) Detection rate of poisonous data (F1-score)

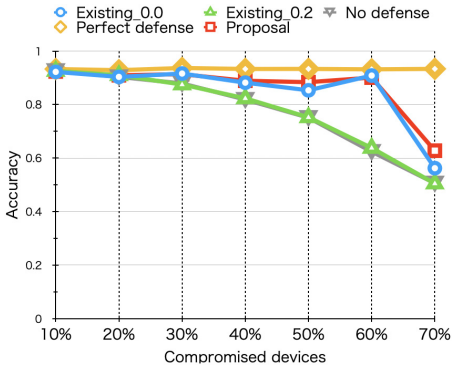


(c) Detection rate of poisonous data (Accuracy)

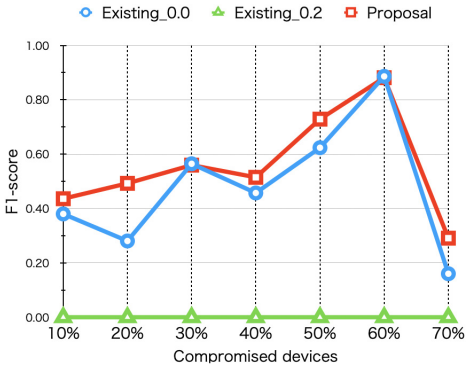
Fig. 9. Results of model accuracy and poisonous data detection rate on synthetic data (attack parameter = 0.3).

without applying the defense techniques decreases as the percentage of the poisonous data increases, while the existing and proposed methods maintain their accuracy relatively well. Also, the value of attack parameter increases, the accuracy of the model applied to each defense method decreases.

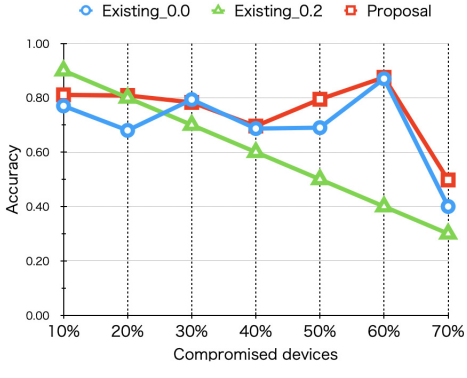
The detection rate of the poisonous data on attack parameter set to 0.5 is shown in Figs. 10(b) and 10(c). These results show that the proposed method is slightly better than the existing method. Regarding the $f1$-score, it can be seen that the detection rate increases while the poisonous data increase up to 60%. This may be because the degree of influence of false positives decreases with an increase in the number of poisonous data. Also, the detection rate dropped sharply when the poisonous data reached 70%. When the poisonous data was 70%, the ratio of poisonous data to total data was more than half for both the existing and proposed methods. Therefore, it is considered that the reason is that it is difficult to judge that the



(a) Accuracy of the model after each defenses



(b) Detection rate of poisonous data (F1-score)



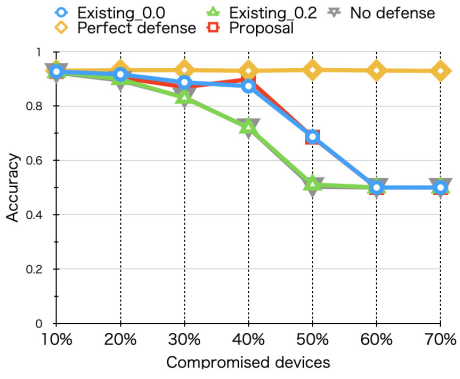
(c) Detection rate of poisonous data (Accuracy)

Fig. 10. Results of model accuracy and poisonous data detection rate on synthetic data (attack parameter = 0.5).

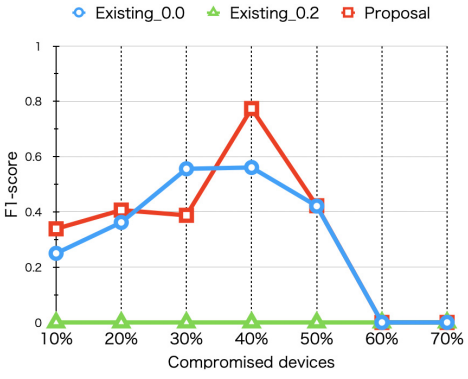
poisonous data has an adverse effect on the model. Figure 12 shows the value of negative impact for a given ratio of poisonous data. As can be seen from this figure, the difference in the negative impact between normal and poisonous devices is small. This indicates that the reason for the lower detection rate may be due to the reason described above.

Similar to the accuracy results for each model, the detection rate of poisonous data decreases as the attack parameter increases. This result is likely due to the following two factors:

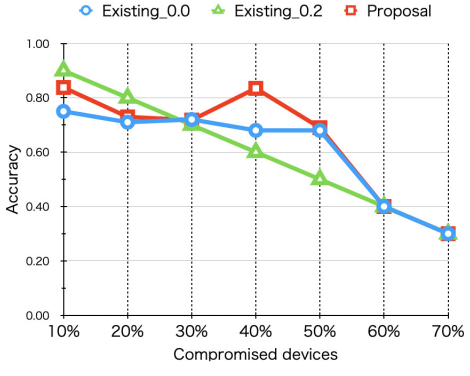
- The stronger effect of poisonous data made it easier to detect the percentage of poisonous data up to a certain value.
- The stronger effect of poisonous data made it more difficult to detect less poisonous data than the weaker parameters.



(a) Accuracy of the model after each defenses



(b) Detection rate of poisonous data (F1-score)



(c) Detection rate of poisonous data (Accuracy)

Fig. 11. Results of model accuracy and poisonous data detection rate on synthetic data (attack parameter = 0.7).

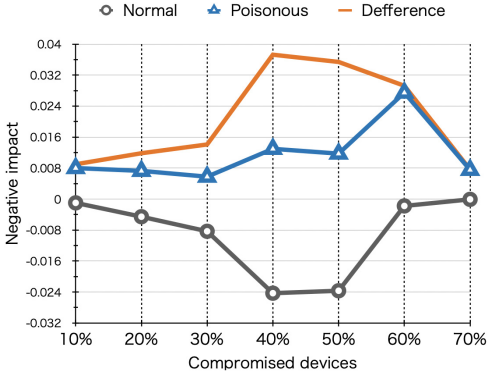


Fig. 12. I_{neg} for the ratio of poisonous data on synthetic data.

5.3. *Environmental sensor telemetry data*

5.3.1. *Setting*

The dataset used in this experiment was obtained from "kaggle" [36], a platform for data analysis. These data ware generated by sensing with an environmental sensor array [37]. Each of the breadboard-based sensor arrays is connected to a Raspberry Pi single board computer. These IoT devices are intentionally placed in physical locations with changing environmental conditions such as temperature and humidity to acquire data. Table 2 presents the attributes of these data.

Table 2. Environmental sensor telemetry data.

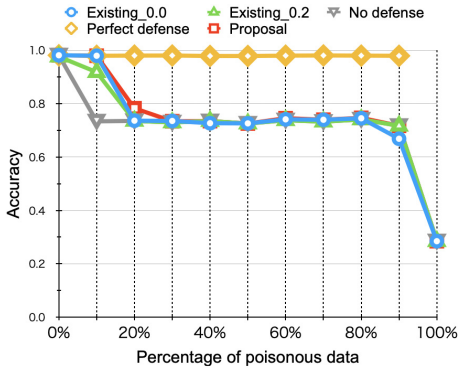
Notation	Description	Units
ts	Timestamp of event	epoch
Device	Unique device name	string
Co	Carbon monoxide	ppm (%)
Humidity	Humidity	percentage
Light	Light detected?	boolean
LPG	Liquid petroleum gas	ppm (%)
Motion	Motion detected?	boolean
Smoke	Smoke	ppm (%)
Temp	Temperature	fahrenheit

As a learning task in this experiment, we assume that the attribute value of "light" is predicted by other sensed data other than "motion". The reason for excluding "motion" from the explanatory variables is to avoid the extreme ease of the task due to its boolean value. In addition, we used a slightly refined attack method from Experiment 5.1.

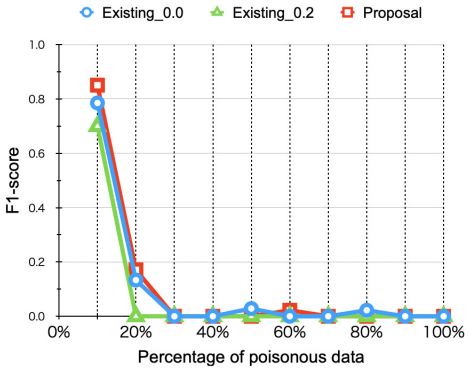
The experimental setup is the same as in the experiments in Secs. 5.1 and 5.2, with the total number of devices set to 10. The number of data for each device is set to 100, and the experiments are conducted.

5.3.2. *Results*

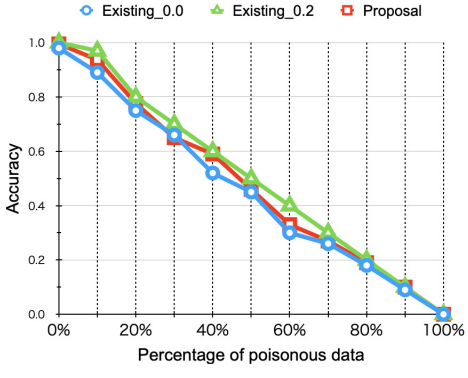
Figure 13 shows the experimental results. As can be seen from the results, there was no significant difference between the existing and proposed methods. Comparing with the model without defense, we can see that the accuracy degradation is suppressed when the percentage of poisonous data is small. The detection accuracy of the poisonous data shows that it is almost not detected when the number of poisonous devices reaches three. This result indicates that the detection of poisonous data is becoming more difficult compared to MNIST and synthetic data.



(a) Accuracy of the model after each defenses



(b) Detection rate of poisonous data (F1-score)



(c) Detection rate of poisonous data (Accuracy)

Fig. 13. Results of model accuracy and poisonous data detection rate on environmental data.

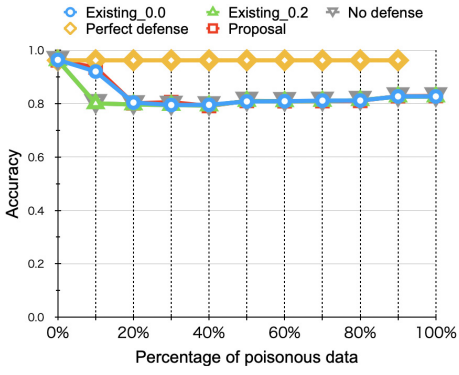
5.4. *Air quality data*

5.4.1. *Setting*

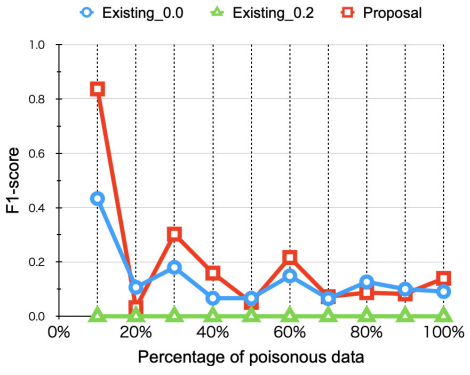
As in Experiment 5.3, this data set was obtained from kaggle. The dataset consists of 2017 air quality data (concentrations of particulate matter PM1, PM2.5 and PM10, temperature, pressure and humidity) generated by a network of 56 low-cost sensors in Krakow, Poland [38]. We used the same setup as in the previous experiments. The attack method is the same as in Experiment 5.3.

5.4.2. *Results*

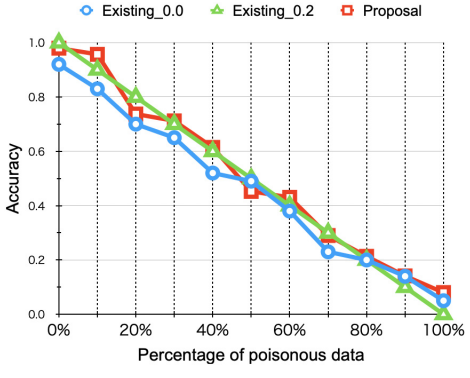
Similar to the previous results, Fig. 14(a) shows the accuracy of each model. Compared to the experiments on the other data sets, the changes in the models other than perfect defense are the smallest. However, (b) and (c) show that the detection accuracy of the proposed method is slightly higher. Regarding the detection accuracy, the detection rate decreases significantly when there are multiple poisonous devices.



(a) Accuracy of the model after each defenses



(b) Detection rate of poisonous data (F1-score)



(c) Detection rate of poisonous data (Accuracy)

Fig. 14. Results of model accuracy and poisonous data detection rate on air quality data.

Conversely, when there is a single harmful device, the proposed method removes the poisonous data with high accuracy.

6. Discussion

6.1. *Attacks by defenders*

The proposed method is a defense method against poisoning attacks, however, it uses poisonous data generated by poisoning attacks. In the evaluation experiments shown in Sec. 5, both attackers and defenders use the same poisoning attack methods. However, it is unlikely to use the same methods as those used by attackers in the real world. Our proposed method takes advantage of the feature that the poisonous effects are not lost even if the attack algorithm is executed on the poisonous data again. Thus, it is necessary to verify whether the assumption is overturned by a combination of attack algorithms in the future.

6.2. *Adversarial impact of each devices*

In the existing methods, the threshold to judge whether a device is normal or poisonous is uniformly set for each device. Conversely, in the proposed method, an adversarial impact calculated for each device is used. Table 3 shows the adversarial impact I_{adv} of normal devices and poisonous devices in the proposed method. The reason why the accuracy of the proposed method is higher than those of the existing methods in the results presented in Sec. 5 is due to the following two factors:

- The negative impact of normal devices is less than or equal to zero and is judged to be normal regardless of the adversarial impact.
- The negative impact of the poisonous devices is close to the adversarial impact and the removal rate becomes high (see Table 4).

Table 3. I_{adv} of proposal in experiment I.

Compromised devices (%)	Normal devices	Poisonous devices
8	0.064	0.115
17	0.041	0.078
25	0.027	0.064
33	0.004	0.024

Table 4. I_{neg} of proposal in experiment I.

Compromised devices (%)	Normal devices	Poisonous devices
8	−0.011	0.091
17	−0.008	0.076
25	−0.022	0.055
33	−0.014	0.008

6.3. *The case of majority of poisonous data*

As described in Sec. 4.2, the proposed method judges that all the devices are normal if I_{neg} is less than 0 and does not remove the corresponding data. However, when more than half of the datasets to be protected become poisonous data, it is possible to consider that "normal data have a negative impact on poisonous data". In other words, the situation is similar to the reversal of the positions of normal and poisonous data. In such a case, the defense algorithm of the proposed method in this paper can detect normal devices rather than poisonous devices by taking a relatively high value of I_{neg} for normal data.

6.4. *Against an attacker who knows the internal structure of the defense method*

As described in Sec. 2.1, this study assumes an environment where data are acquired from multiple devices in an IoT environment. However, the method proposed in this

paper is not limited to such an environment and can be applied to several data sources. For example, consider the case where the annotation of training data is outsourced. In this case, we can divide the data sources by company or by employee. If there are attackers lurking among these sources, they can launch a poisoning attack. In this case, however, it is necessary to assume that the poisoning attacker cannot detect the modification of the data before and after the annotation by comparing hash values.

6.5. *Future work*

Although the experimental setup of the poisoning attack treated in this paper is based on previous studies, several other poisoning attacks have been studied in the past. Thus, it is necessary to verify the effectiveness of these methods against other poisoning attacks through experiments. In this study, we used SVM for evaluation, but our proposed method does not depend on the machine learning methods. Thus, it is necessary to use a different machine learning method for future evaluation.

7. Conclusion

In this paper, we proposed a defense method against poisoning attacks to improve the accuracy of the model after the defense in an IoT environment where data are aggregated from multiple devices. The proposed method uses a poisoning attack and defines an index I_{adv} for each device to determine whether the data are normal or poisonous. We evaluate the performance of the proposed method by comparing it with existing methods. The experimental results show that the proposed method improves the accuracy of detecting poisonous data and the stability of the detection accuracy. Through the discussion, we showed that it is necessary to verify the effectiveness of the correspondence between the attack methods used by the attackers and those used in the proposed method.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers JP17H04705, JP18H03229, JP18H03340, JP18K19835, JP19H04113 JP19K12107 and JST, PRESTO Grant Number JPMJPR1934.

References

[1] T. D. Duan, T. L. H. Du, T. V. Phuoc and N. V. Hoang, Building an automatic vehicle license plate recognition system, in *Int. Conf. Comput. Sci.*, 2005, pp. 59–63.

[2] R. Carbonneau, K. Laframboise and R. Vahidov, Application of machine learning techniques for supply chain demand forecasting, *Eur. J. Oper. Res.* **184**(3) (2008) 1140–1154.

[3] B. Battista, F. Giorgio and R. Fabio, Multiple classifier systems for robust classifier design in adversarial environments, *Int. J. Mach. Learn. Cybern.* **1**(1) (2010) 27–41.

[4] M. Tony and W. Brendon, Spambayes: Effective open-source, bayesian based, email classification system, in *Conf. Email and Anti-Spam*, 2004.

[5] A. L. Buczak and E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, *IEEE Commun. Surveys Tutorials* **18**(2) (2016) 1153–1176.

[6] M. Bagaa, T. Taleb, J. B. Bernabe and A. Skarmeta, A machine learning security framework for iot systems, *IEEE Access* **8**(12) (2020) 114066–114077.

[7] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi and A. P. Sheth, Machine learning for internet of things data analysis: A survey, *Digital Commun. Netw.* **4**(3) (2018) 161–175.

[8] Y. Tsukagoshi, S. Egami, Y. Sei, Y. Tahara and A. Ohsuga, Ontology-based correlation detection among heterogeneous data sets: A case study of university campus issues, in *2020 IEEE Third Int. Conf. Artificial Intelligence and Knowledge Engineering*, 2020, pp. 33–40.

[9] J. P. McCrae for the Insight Centre for Data Analytics, The linked open data cloud, https://lod-cloud.net/.

[10] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis and G. Loukas, A taxonomy and survey of attacks against machine learning, *Comput. Sci. Rev.* **34** (2019) 100199.

[11] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein and J. D. Tygar, Adversarial machine learning, in *Proc. 4th ACM Workshop on Security and Artificial Intelligence*, 2011, pp. 43–58.

[12] M. Barreno, B. Nelson, A. D. Joseph and J. D. Tygar, The security of machine learning, *Mach. Learn.* **81**(2) (2010) 121–148.

[13] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay and D. Mukhopadhyay, Adversarial attacks and defences: A survey, CoRR, abs/1810.00069 (2018).

[14] M. Barreno, B. Nelson, R. Sears, A. D. Joseph and J. D. Tygar, Can machine learning be secure?, in *Proc. 2006 ACM Symp. Information, Computer and Communications Security*, 2006, pp. 16–25.

[15] B. Biggio, B. Nelson and P. Laskov, Poisoning attacks against support vector machines, in *Proc. 29th Int. Conf. Machine Learning*, 2012, pp. 1467–1474.

[16] T. Chiba, Y. Sei, Y. Tahara and A. Ohsuga, A defense method against poisoning attacks on iot machine learning using poisonous data, in *2020 IEEE Third Int. Conf. Artificial Intelligence and Knowledge Engineering*, 2020, pp. 100–107.

[17] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar and K. Xia, Exploiting machine learning to subvert your spam filter, in *Proc. 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2008, pp. 1–9.

[18] B. Biggio and F. Roli, Wild patterns: Ten years after the rise of adversarial machine learning, *Pattern Recogn.* **84** (2018) 317–331.

[19] M. Li, Y. Sun, H. Lu, S. Maharjan and Z. Tian, Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems, *IEEE Internet Things J.* **7**(7) (2020) 6266–6278.

[20] Y. Yao, H. Li, H. Zheng and B. Y. Zhao, Latent backdoor attacks on deep neural networks, in *Proc. 2019 ACM SIGSAC Conf. Computer and Communications Security*, 2019, pp. 2041–2055.

[21] 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 *2018 IEEE Symp. Security and Privacy*, 2018, pp. 19–35.

[22] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru and F. Roli, Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks, in *28th USENIX Security Symp.*, 2019, pp. 321–338.

[23] 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 *Advances in Neural Information Processing Systems*, Vol. 31, 2018, pp. 6103–6113.

[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu, Towards deep learning models resistant to adversarial attacks, arXiv preprint arXiv:1706.06083 (2017).

[25] J. Steinhardt, P. W. W. Koh and P. S. Liang, Certified defenses for data poisoning attacks, in *Advances in Neural Information Processing Systems*, Vol. 30, 2017, pp. 3517–3529.

[26] C. Dunn, N. Moustafa and B. Turnbull, Robustness evaluations of sustainable machine learning models against data poisoning attacks in the internet of things, *Sustainability* **12**(16) (2020) 1–17.

[27] N. Baracaldo, B. Chen, H. Ludwig, A. Safavi and R. Zhang, Detecting poisoning attacks on machine learning in iot environments, in *2018 IEEE Int. Congress on Internet of Things*, 2018, pp. 57–64.

[28] N. Baracaldo, L. A. D. Bathen, R. O. Ozugha, R. Engel, S. Tata and H. Ludwig, Securing data provenance in internet of things (IOT) systems, in *Service-Oriented Computing — ICSOC 2016 Workshops*, 2017, pp. 92–98.

[29] B. A. Nelson, Behavior of machine learning algorithms in adversarial environments, Ph.D. thesis, University of California Berkeley (2010).

[30] 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 *Machine Learning and Knowledge Discovery in Databases*, 2013, pp. 387–402.

[31] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, Intriguing properties of neural networks, arXiv preprint arXiv:1312.6199, 2014.

[32] B. Wang and N. Z. Gong, Stealing hyperparameters in machine learning, in *2018 IEEE Symp. Security and Privacy*, 2018, pp. 36–52.

[33] R. Shokri, M. Stronati, C. Song and V. Shmatikov, Membership inference attacks against machine learning models, in *2017 IEEE Symp. Security and Privacy*, 2017, pp. 3–18.

[34] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy and B. Edwards, Adversarial robustness toolbox v1.2.0 (2018).

[35] Y. Zhou, M. Kantarcioglu, B. Thuraisingham and B. Xi, Adversarial support vector machine learning, in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 1059–1067.

[36] Kaggle, https://www.kaggle.com/.

[37] Towards Data Science, Getting started with IOT analytics on AWS, https://towards-datascience.com/getting-started-with-iot-analytics-on-aws-5f2093bcf704.

[38] Air quality data from extensive network of sensors, https://www.kaggle.com/datascienceairly/air-quality-data-from-extensive-network-of-sensors.