Security and Communication Networks

# Attack and Defence of Smart Systems

Lead Guest Editor: Ting Chen
Guest Editors: Xiaodong Lin, Xiapu Luo, and Jiang Ming

# Attack and Defence of Smart Systems

Security and Communication Networks

# Attack and Defence of Smart Systems

Lead Guest Editor: Ting Chen
Guest Editors: Xiaodong Lin, Xiapu Luo, and Jiang Ming

# Contents

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9924684, 13 pages
https://doi.org/10.1155/2021/9924684

*Research Article*

# A Defense Framework for Privacy Risks in Remote Machine Learning Service

**Yang Bai**,[1,2] **Yu Li**,[1] **Mingchuang Xie**,[1] **and Mingyu Fan**[1]

[1]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China*
[2]*No. 30, Institute of CETC, Chengdu, China*

Correspondence should be addressed to Mingyu Fan; ff98@163.com

Received 5 March 2021; Revised 22 May 2021; Accepted 5 June 2021; Published 18 June 2021

Academic Editor: Jiang Ming



In recent years, machine learning approaches have been widely adopted for many applications, including classification. Machine learning models deal with collective sensitive data usually trained in a remote public cloud server, for instance, machine learning as a service (MLaaS) system. In this scene, users upload their local data and utilize the computation capability to train models, or users directly access models trained by MLaaS. Unfortunately, recent works reveal that the curious server (that trains the model with users' sensitive local data and is curious to know the information about individuals) and the malicious MLaaS user (who abused to query from the MLaaS system) will cause privacy risks. The adversarial method as one of typical mitigation has been studied by several recent works. However, most of them focus on the privacy-preserving against the malicious user; in other words, they commonly consider the data owner and the model provider as one role. Under this assumption, the privacy leakage risks from the curious server are neglected. Differential privacy methods can defend against privacy threats from both the curious sever and the malicious MLaaS user by directly adding noise to the training data. Nonetheless, the differential privacy method will decrease the classification accuracy of the target model heavily. In this work, we propose a generic privacy-preserving framework based on the adversarial method to defend both the curious server and the malicious MLaaS user. The framework can adapt with several adversarial algorithms to generate adversarial examples directly with data owners' original data. By doing so, sensitive information about the original data is hidden. Then, we explore the constraint conditions of this framework which help us to find the balance between privacy protection and the model utility. The experiments' results show that our defense framework with the AdvGAN method is effective against MIA and our defense framework with the FGSM method can protect the sensitive data from direct content exposed attacks. In addition, our method can achieve better privacy and utility balance compared to the existing method.

## 1. Introduction

In recent years, machine learning technology has rapidly gained popularity, as its model can be improved automatically through learning from the training dataset. Benefit from the development enables more efficient storage, processing, and computation, and more and more machine learning models are widely applied in the real world for classification or regression tasks for many domains, such as image classification [1], speech recognition [2], healthcare data management [3], and financial analysis [4]. Samples are used to train the machine learning model to represent many individuals' sensitive information, such as patients' healthcare information, personal preference, and personal photos. For instance, image classification, especially facial recognition technologies, is applied in many scenes including activity monitoring and identification. Healthcare record management requires clinical features and medical data from patients to learn a model for diagnosis and prognosis. In finance, an individual's trade record, current price records, and so on are used to study the financial prediction machine learning and financial risk analysis systems.

Machine learning models deal with collective sensitive data usually trained in a remote public cloud server, for instance, machine learning as a service (MLaaS) system. In

this scene, users upload their local data and utilize the computation capability to train models, or users directly access models trained by MLaaS. Unfortunately, recent works reveal that the curious server (that trains the model with users' sensitive local data and is curious to know the information about individuals) and the malicious MLaaS user (who abused to query from the MLaaS system) will cause privacy risks. For example, Shrokri et al. [5] came up with a membership inference attack (MIA) against machine learning models. The MIA adversary uses machine learning to train an attack model to speculate if a given data record was a member of the target model's training dataset. These privacy risks not only can directly violate the privacy of the training set but also can be a gateway to further attacks [6]. That is, some adversaries use the inference information to construct other security threats. For example, if the individual's biometric features were inferred out, the attacker can utilize this information to make a personating attack or obtain illegal authorities. These further attacks might pose additional severe information leakage or other heavy security risks [5].

The adversarial method is one of the typical mitigations to address the machine learning privacy risks. Nasr et al. [7] formalized the privacy risk preserving as a min-max game optimization problem that minimizes the classification error of the target model and minimizes the inference adversary's maximum gain. They use the stochastic gradient descent algorithm to optimize the min-max problem. Jia et al. [8] proposed a MemGuard to defend against black-box MIAs by adding a carefully crafted noise vector to the target model's output confidence score vector. Wang et al. [9] jointly formulate model compression and MIA as MCMIA, utilizing model compression against MIAs in deep neural networks. However, most of them focus on the privacy-preserving against the malicious user; in other words, they commonly considered the data owner and the model provider as one role and the adversarial perturbation objective always focuses on the target model or the confidence score of the target model. Under this assumption, the privacy leakage risks from the curious server are neglected. It is not suitable for the data user to upload their local data to the MLaaS system for training a machine learning model remotely. That is, the privacy leakage risks from the model provider are commonly neglected, and the owner's data privacy cannot be preserved entirely. Differential privacy methods can defend against privacy threats from both the curious server and the malicious MLaaS user by directly adding noise to the training data. Nonetheless, the differential privacy method will decrease the classification accuracy of the target model heavily.

In this paper, we investigate existing adversarial perturbation-based defenses to categorize them. Then, we propose a generic privacy-preserving framework based on an adversarial method to defend both the curious server and the malicious MLaaS user. The framework can adapt with several adversarial algorithms to generate adversarial examples directly with data owners' original data. By doing so, sensitive information about the original data is hidden. In addition, we explore the constraint conditions of this framework which help us to find the balance between privacy protection and the model utility. The framework consists of one adversarial perturbation generator, shadow models, and privacy evaluator. The generator learns to perturb the original data of the local user. We leverage the adversarial method to diminish the effect of the membership inferences by generating adversarial examples of the original training data. By doing so, sensitive information about the training data is hidden. The privacy evaluator then detects whether there exist privacy risks. The feedback scheme and the constraint are used to find out the balance between privacy-preserving and the performance of the target model. The main contributions of this paper are summarized as follows:

(i) Taxonomy of existing adversarial method-based privacy risk mitigation. We propose a pioneering study to categorize adversarial method-based defenses by different perturbation objects; and we analyze the limitation of this existing mitigation. These works help us to build general recognition in this field.

(ii) The generic defense framework for adversarial method against privacy risk. We utilize the feedback mechanism and some constraint conditions to develop a common adversarial defense framework for machine learning privacy-preserving. We also explore the constraint conditions that make our method more effective. This framework can protect the original training data's privacy under data sharing and MLaaS scenarios; and we implement three different kinds of adversarial example algorithms based on privacy-preserving mitigation with our generic framework. We exploit the MIA as the classical privacy leakage evaluator to verify the effectiveness of these three defenses.

(iii) Comprehensive analysis of the influence factors for the proposed method. We investigate defense factors, including adversarial algorithms, perturbation rates, adversarial distance, and data type, showing that, under appropriate factors strategies, our defense framework with the AdvGAN method is effective against MIA and our defense framework with the FGSM method can protect the sensitive original data from direct content exposed attacks. In addition, our method can achieve more privacy and utility compared to the existing method, for instance, the min-max method and the differential privacy. Furthermore, the experiment results show that we can control the defense performance with parameters such as distortion rate and distance threshold.

## 2. Background and Related Work

In this section, we present background and related work on privacy threats in data sharing and MLaaS scenes and briefly review the membership inference attacks and machine learning privacy-preserving proposed in previous works.

*2.1. Privacy Threats in Data Sharing and MLaaS Scenes.* Fatemehsadat et al. [10] divided existing threats to privacy in machine learning systems into two main categories of direct and indirect information exposure hazards. They define in the direct threats that the adversary can gain direct access to sensitive datasets. Private data can be exposed through data sharing scenes or the cloud service that receives it to run a process on it [10]. For example, in the MLaaS platform, training data must be revealed to the service. If the MLaaS service operators are malicious, they can directly access the sensitive training data. However, users may not want to reveal their private information. The indirect information exposure means the attacker attempts to infer or guess the information and does not have access to the actual information [10]. A number of works have focused on the indirect information exposure, such as model extraction attacks [11], model inversion [12], and membership inference attacks [5]. Tramer et al. [11] explore model extraction attacks that exploit the tension between query access and confidentiality in machine learning models and aim to copy functions from victim models in the MLaaS setting. Fredrikson et al. [13] devise the model inversion attack in which the adversary access to a machine learning model abused to learn sensitive genomic information about individuals. Fredrikson made a further exploration to develop a new class of model inversion attacks based on prediction results from the MLaaS APIs. Shokri et al. [5] proposed the membership inference attack performed entirely through the MLaaS services. The model extraction attacks refer to the confidentiality of the victim model. The model inversion and the membership inference attack threaten the privacy of training data. The privacy risks in data sharing and MLaaS are not only from adversaries but may also come from the curious server and other malicious users.

*2.2. Membership Inference Attacks (MIAs).* In an MIA, the adversary queries the victim model to determine whether a particular sample is included in the training set of the victim model. Previous research demonstrates that overfitting is one of the reasons to cause the MIAs [14]. Shokri et al. [5] introduced the first MIA against machine learning system, and they turned MIA into a binary classification problem. In Shokri et al.'s method, the process of MIA is as follows. In the first stage, the attacker prepares a set of candidate data records. Next, they query access to the victim model and gain a classification probability vector from the victim model. After that, the adversary trains shadow models to craft training data for the attack model, followed by training the attack model and using the attack model to determine whether the candidate samples are members of the target model's training dataset. For each candidate record, there are two possible classes: class "member" means that the candidate data is a member of the target model's training dataset, and the class "nonmember" means that the candidate data is not in the training dataset. Hayes et al. [6] present the first MIAs against generative adversarial networks (GANs). ML-leaks [15] relaxes some assumptions of Shokri et al.'s work [5], such as the number of shadow models, the knowledge of the target model structure, and the target model's dataset information.

Despite the fact that the previous research has focused on MIA as a method of attack, recent works [7, 8, 16, 17] have used MIA as a privacy leak assessment tool. Jayaraman et al. [16] used membership inference (MI) to quantify the impact of differential privacy with the privacy loss of different ML models. The MI-based evaluator helps to find the range of differential privacy parameters to achieve a balance between utility and privacy and also to evaluate the privacy leakage of training data at risk of exposure. Song et al. [17] exploited MI to evaluate the defense approach of adversarial regularization [7] and MemGuard [8] with a new metric called the privacy risk score. Jia et al. [8] and Nasr et al. [7] used neural network classifiers to evaluate the mitigation performance. Thus, we choose MI as the classical privacy leakage evaluator to verify the effectiveness of our defense framework.

*2.3. Machine Learning Privacy-Preserving.* Existing ML privacy-preserving can be classified into three categories, that is, generalization method [5, 15], differential privacy [16, 18, 19], and adversarial methods [7–9].

*2.3.1. Generalization Method.* Previous works show that the overfitting model can memorize the information about the training data; furthermore, it can cause the privacy leakage risks, such as MIA. Thus, generalization is one of the most popular mitigations against ML privacy risks. Salimans et al. [20] presented the weight normalization, a simple reparameterization of the weight vectors in a neural network, by speeding up the convergence of stochastic gradient descent. Srivastava et al. [21] used dropout to fit the overfitting problem in deep neural network with a large number of parameters. Shokri et al. [22] and Salem et al. [15] found that dropout only was effective to degrade overfitting and strengthen privacy-preserving in neural networks. Salem et al. [15] exploited model stack as the generalization method which is suitable for all ML models. Shokri et al. [5] utilized standard regularization to mitigate privacy risk caused by overfitting. However, Long et al. [14] discovered that overfitting is a sufficient but not necessary condition for MIA to succeed. They found that existing generalization techniques are less effective in MIA protecting.

*2.3.2. Differential Privacy.* Differential privacy has been regarded as a strong privacy standard [23–27]. Differential privacy is one of the classical defenses against privacy risks. According to the ML processes, the differential privacy mechanism can be applied as the input perturbation, the gradient perturbation, the objective perturbation, and the output perturbation [28]. Reference [29] presented a differentially private GANs model which includes a Gaussian noise layer in the discriminator in case of a generative adversarial network to make the output and the gradients differential privacy with respect to the training data. Papernot et al. [30] used Private Aggregation of Teacher Ensembles (PATE) to construct an output perturbation.

Reference [31] used the differentially private stochastic gradient descent algorithm (DP-SGD) to prevent memorization. Although differential privacy has a significant effect on privacy protection, the introduction of a differential privacy mechanisms will greatly reduce the classification performance of the target model. This disadvantage hampers the application of differential privacy in real-world scenarios [16].

*2.3.3. Adversarial Methods.* Previous works showed that the adversarial method-related defenses contain two branches of research, that is, adversarial training and adversarial examples. Nasr et al. [7] put forward a min-max game which designs an adversarial training algorithm that minimizes the prediction loss of the model as well as the maximum gain of the inference attacks. This strategy can guarantee that membership privacy acts also helped to generalize the target model. Jia et al. [8] proposed a method based on adversarial examples, which adds noise to each confidence score vector by victim classifier. Its aim is to mislead the classification ability of attack models through carefully crafted adversarial samples. MemGuard [8] can effectively defend against MIAs and achieve better privacy-utility tradeoffs compared to previous works. However, Nasr et al. and Jia et al. considered the data owner and the model provider as one role. Therefore, these mitigations are not suitable for data sharing or MLaaS scenes in which the data owner should reveal their sensitive data to the services. In addition, there are many methods and algorithms for adversarial training and adversarial examples [33–39]. Therefore, it is necessary to construct a general defense framework to adapt the different sample adversarial algorithms, to find a balance between the defense of privacy and the performance of the target model, and to cover a wider range of privacy risk scenarios.

## 3. Taxonomy of Adversarial Method-Based Defense

In general, adversarial method defense can be categorized into three types by different perturbation objects: the input training data, the model, and the output prediction vector. We categorize the existing and the proposed adversarial method-based defenses in Table 1. We describe all the possible types in the following paragraphs.

*3.1. Output Perturbation-Based Defenses.* This kind of defense aims at adding crafted noise to the confidence score vector or labels to synthesize adversarial examples to mislead the attack classifiers. For example, MemGuard [8] adds noise vector to the confidence score with a certain probability to defend against black-box membership inference attacks. Yang et al. [32] proposed a framework to purify the confidence score vectors by reducing their dispersion. Yang's purification framework can defend against the model inversion attack and the membership inference attack. Both Jia et al. [8] and Yang et al. [32] considered the data owner and the model provider as one role. This assumption is not suitable for the scenario where users upload their local data to train a model remotely. This defense is the effective mitigation for privacy threats that the adversary infers the reconstruction and the membership of a training data from the probability predicted by the victim classifier.

Table 1: Taxonomy of adversarial method-based defenses.

| Defenses | Perturbation object | | |
|---|---|---|---|
| | Input | Model | Output |
| MemGuard [8] | | | ✓ |
| Yang et al. [32] | | | ✓ |
| MCMIA [9] | | ✓ | |
| Min-max [7] | | ✓ | |
| Our method | ✓ | | |

*3.2. Model Perturbation-Based Defenses.* In this defense type, defenders exploit model compression or model adversarial training to reduce the privacy risks. For instance, Wang et al. [9] jointly formulated model compression and MIA as MCMIA to reduce the information leakage from MIA. Min-max method [7] designed an adversarial training algorithm that minimizes the classification loss of victim model and maximum gain of attack model. These defenses can mitigate the privacy risk caused by the model overfitting, whereas these defenses are hard to be deployed to a public MLaaS platform.

*3.3. Input Perturbation-Based Defenses.* A number of proposed adversarial method-based defenses almost focus on the output perturbation and the model regularization to reduce the privacy leakage. As Figure 1 shows, both output perturbation and model perturbation schemes cannot defend against the privacy risks from the curious model provider. Input perturbation-based defenses which add noise directly to the training data can solve this problem. However, it is a challenge to find the balance between privacy-preserving and the utility of the classification model.

## 4. Generic Framework for Adversarial Method-Based Defenses

In this section, we begin by describing the privacy leakage in remote model training scenario (MLaaS) to formulate the problem of defending against these threats. Then, the generic defense framework is designed for adversarial method-based defense. In addition, we analyze the constraint conditions of the framework. Finally, we introduce three adversarial algorithms to implement the framework-based mitigation.

*4.1. Problem Formulation.* In a typical MLaaS application, there are four parties: user, model provider, attacker, and defender. We discuss each party as follows.

*4.1.1. User.* There are two kinds of users in the MLaaS scenario. The first user has some sensitive training data, such as facial images, healthcare records, financial information, and individual performance. As the user does not have

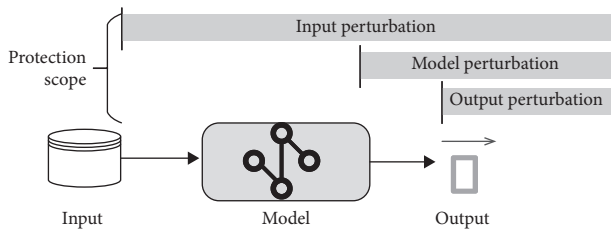



Figure 1: The protection scope of different kinds of adversarial method defenses.

enough computation resources, they want to input these local data to train a machine learning classifier by a remote MLaaS platform. Then the user can gain a trained classifier for their machine learning applications. The second user can directly exploit the classifier deployed in the MLaaS platform to query for some examples and obtains the classification result from the remote services. The local datasets $X$ are uploaded to the remote service platform.

*4.1.2. Model Provider.* The model provider commonly is the public MLaaS platform. The model provider has two abilities: First of all, it can supply the initial model and the computation resource to users for training a model suitable for their characters. The second capability is that the provider can offer a trained model to users for querying and return the classification result (the confidence score vector of classification) for users. We call the machine model as target model (victim model).

*4.1.3. Attacker.* An attacker aims to obtain the information about user's sensitive data. The adversary probably is malicious users; they aim to infer the information about the training data. For instance, the adversary utilizes the black-box MIA [5, 6] to infer the members of the training dataset by querying from the target model. These attackers construct a binary classifier that can speculate out whether a query data record is one of the target model's training datasets or not by its confidence vector of the target model. Besides, the attacker may be a curious server of MLaaS that wants to know the details about user's uploading data. During the training and querying process, the user's data record is directly exposed to the server. So, attackers can easily achieve their goals.

*4.1.4. Defender.* The defender could be the user (data owner) or a trusted third party that has access to the user's uploading data $X$. For any training tasks or query tasks from the user to the model provider, the defender directly perturbs the uploading data to hide sensitive information to prevent the adversaries from obtaining exposed data or launching inference attacks. The defender aims to achieve two goals: The first goal is that the defender protects user's sensitive data from directly exposed risks or data inference attacks, such as MIAs. The second goal is that the defender tries to find the balance between privacy-preserving and target model's classification utility.

*4.2. Framework Architecture.* To defend against privacy threats in remote ML service scenarios, we propose a defense framework as depicted in Figure 2. Our defense framework consists of three components: (1) an adversarial perturbation generator that crafts adversarial examples to hide the sensitive information of uploading data, (2) a simulator to mimic the classification probability of the target model, and (3) a privacy leakage evaluator that detects the privacy leakage extent. The evaluator gives feedback to the generator to optimize the performance and the balance between defending against privacy threats and the target model's performance.

*4.2.1. Adversarial Perturbation Generator.* Although the adversarial examples are treated as harmful in many poisoning scenarios, they can be used to achieve our privacy protection goal. The fundamental idea is to generate an adversarial dataset, in which the sensitive privacy of original data is masked. As shown in Figure 2, the adversarial perturbation generator's goal is to find a set of adversarial examples $X^{\text{adv}}$ which can conceal the sensitive information of the original uploading dataset $X$. In consideration of the target model's classification performance, we do the adversarial distortion as slightly as possible. That is, the Euclidean distance $L_2(x^{\text{adv}}, x)$ between each adversarial record and the original data should be small. We introduce the distance threshold $\epsilon$, and we aim to achieve $L_2(x^{\text{adv}}, x) \leq \epsilon$ to help the generator to find out the appropriate adversarial examples which will be uploaded to train the simulator. There are many adversarial algorithms, for instance, the AdvGAN [40, 41], the Fast Gradient Sign Method (FGSM) [34], and the OPTMARGIN [42]. The adversarial algorithm is regarded as the basic support of the adversarial perturbation generator. The details of adversarial algorithms exploited in this paper are introduced in Section 5.1. We bring a parameter $\alpha$ named as the perturbation rate, which can control the ratio of adversarial examples in uploading training data.

*4.2.2. Simulator.* The simulator is constituted by one or several ML models that mimic the probability distribution of the target model. Shokri et al. [5] and Salem et al. [15] explored how to construct shadow models to mimic the target model effectively. We use the same method to construct the simulator. In our framework, we aim to generate adversarial perturbed data which has two attributes: (1) the perturbed data $X^{\text{adv}}$ has the same classification label as the original record's label. (2) The generated data can be used to train and minimize the quadratic loss function $L(Y, f(X^{\text{adv}}))$ of the simulator.

*4.2.3. Privacy Leakage Evaluator.* We exploit the privacy leakage evaluator to detect the privacy risk level in order to evaluate the privacy-preserving effectiveness of the adversarial example generated module. The privacy leakage evaluator can adapt to multiple evaluation plugins. In this paper, we utilize the membership inference method as the

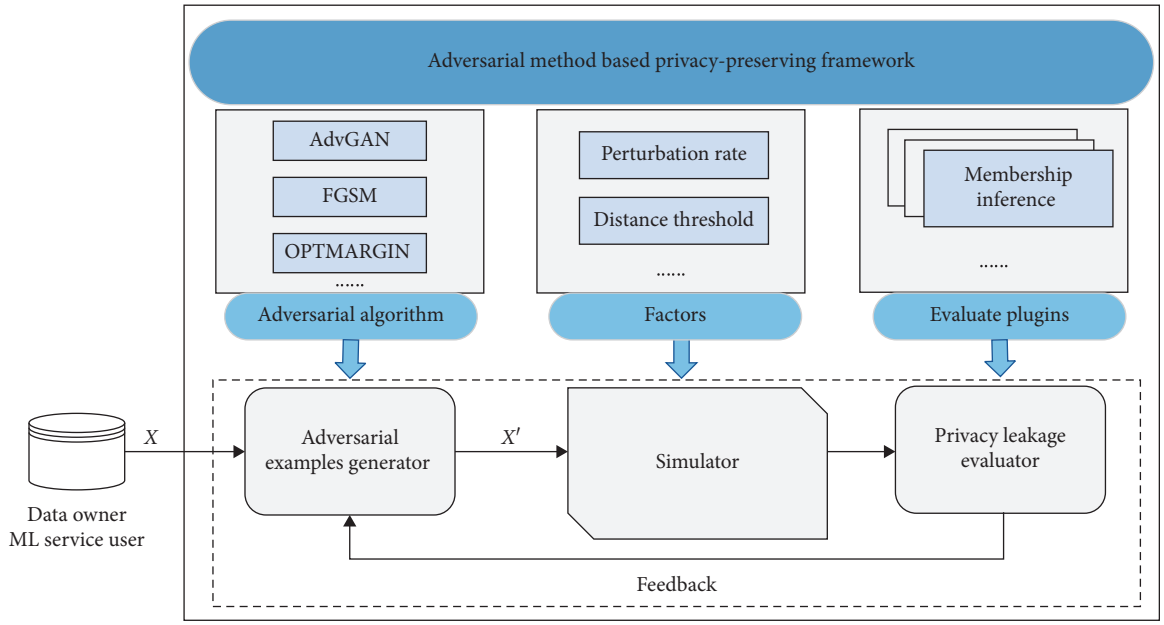



Figure 2: Adversarial method-based privacy-preserving framework.

typical evaluation plugin to verify the performance of our defense framework. The evaluator is an MI classifier that speculates whether an example is one of the simulator's members. In addition, this module sends the evaluation result to adversarial perturbation generator for helping the generated module to find out the proper adversarial examples. The assessment function considers several constraint conditions, for example, the generalization gap, the decreasing level of MI accuracy, and the loss function of the MI classifier. The loss function of the MI classifier is formulated as $R_{\text{emp}}(f) = (1/N)\sum_{N}^{i=1} L(y_i, f(x_{x_i}))$. In this module, the constraint conditions will be sent to the adversarial perturbation generator module for noise production parameter adjustment direction. If the MI inference accuracy is too large, we need to adjust the direction of greater perturbation disturbance. On the contrary, we need to adjust in the direction of less perturbation disturbance.

*4.2.4. Defense for MIA.* The defense framework can be detailedly specialized in the specific scenario of mitigating the MIA when the user uploads their local data to the remote machine learning service. In this paper, we explore the MIA proposed by Shokri et al. [5] in which the user supplies training data as inputs to the MLaaS service (model provider) to construct a model from the uploading data. Then, the user can use the model in other applications. In this scene, the adversary can access querying the model and obtain the classification result.

*Algorithm.* 1 shows the work method in this case. The adversarial perturbation generator is trained to synthesize a set of training data, which can mask their data information to the MI adversary (Goal I). At the same time, the performance of the simulator trained by these syntheses data does not degrade heavily (Goal II). The simulator and the evaluator are the assistant tools to help the generator to produce adversarial examples which satisfy the needs of Goal I and Goal II.

We formally present this MIA defense problem by the following optimization problem.

Let $x$ be the user's original data. $x^{\text{adv}}$ is the adversarial example based on $x$. The Euclidean distance $d(x^{\text{adv}}, x)$ is used to measure the adversarial distortion between the adversarial example and the original data. The distance threshold $\epsilon$ is used as the upper bound of the adversarial distortion as inequation (1). We chose the adversarial example, whose adversarial distortion is smaller than the threshold $\epsilon$, to construct the uploading training dataset. We introduce the perturbation rate $\alpha$ to control the degree of adversarial perturbation. $N$ denotes the size of the uploading dataset. The size of adversarial examples which the generator needs to synthesize is $\alpha \times N$. When $\alpha = 1$, it means that all the upload training data is the generated adversarial examples. The value of $\epsilon$ can be set by experiences.

$$d\left(x^{\text{adv}}, x\right) = L_2\left(x^{\text{adv}}, x\right) \leq \epsilon. \tag{1}$$

We aim to maintain the performance of the simulator. Thus, we exploit the method to maintain the same classification label between the original data $x$ and the perturbed data $x^{\text{adv}}$ proposed by Jia et al. [8]. The formulation can be described as in equation (2). Let $f: X \longrightarrow Y$ be the classification function of the simulator. Each output $f(x)$ is a confidence vector. The classification label of original data and adversarial data can be represented as $\arg\max f(x)$ and $\arg\max f(x^{\text{adv}})$, respectively.

$$\arg\max f\left(x^{\text{adv}}\right) = \arg\max f(x). \tag{2}$$

Let $X^{\text{adv}}$ be perturbed uploading dataset. $Y$ is the label set of original examples $X$. Let $x\left\{\_i\hat{a}\text{dv}\right\}$ indicate the $i$th training

examples. The cross-entropy loss function of simulator is given in equation (3). We aim to minimize the loss function of the simulator, as equation (4) shows, to guarantee that the mitigation causes the minimum influence to the simulator.

$$L\left(Y, f\left(X^{\text{adv}}\right)\right) = \sum_{i=1}^{N} y_i \log f\left(x_i^{\text{adv}}\right) + (1 - y_i)\log\left(1 - f\left(x_i^{\text{adv}}\right)\right), \tag{3}$$

$$\min\left(L\left(Y, f\left(X^{\text{adv}}\right)\right)\right). \tag{4}$$

Let $g$ be the MI evaluator's model. The optimization problem for the MIA is maximizing the loss function of the evaluator, as in the following equation:

$$\max\left(\frac{1}{N}\sum_{i=1}^{N} L\left(y_i^{\text{label}}, g\left(f\left(x^{\text{adv}}\right)\right)\right)\right). \tag{5}$$

## 5. Implementation and Evaluation

In this section, we present details of implementation, including the adversarial algorithms, models, and datasets. Then, we give out experiment results and analysis of our defense framework.

### 5.1. Adversarial Algorithms

*5.1.1. Generative Adversarial Networks- (GANs-) Based Method.* GANs-based method generates adversarial examples with generative adversarial networks (GANs) [41]. As the GANs can learn and approximate the distribution of original records, the GANs-based method can generate perturbation efficiently. $G$ is the generator of AdvGAN. $f$ denotes the target model. Feed the original instance into generator $G$, and produce perturbation $G(x)$. Then, input $x + G(x)$ to discriminator $D$. $D$ aims to encourage generator $G$ to synthesize instance which is indistinguishable from the original data's classification result.

$$f(x) = f(x + G(x)). \tag{6}$$

*5.1.2. Fast Gradient Sign Method (FGSM).* FGSM was first proposed by Goodfellow et al. [34]. $\theta$ denotes the model parameters. $x$ is the original data. $y$ is the real label of the original data $x$. $J(\theta; x; y)$ is the loss function used to train the model with respect to the inputs. $\varepsilon$ is the distortion parameter that controls the perturbation level.

$$\eta = \varepsilon \text{sign}\left(\nabla_x J(\theta; x; y)\right). \tag{7}$$

The adversarial example $x_{\text{adv}}$ can be expressed as

$$x_{\text{adv}} = \eta + x. \tag{8}$$

FGSM is a fast and reliable method to generate adversarial examples which cause more noticeable perturbation compared to other adversarial algorithms. This attribute can be used to protect the original data from the direct content exposed attack.

*5.1.3. OPTMARGIN-Based Defense.* He et al. [42] proposed the OPTMARGIN adversarial method, which can generate low-distortion adversarial examples that are robust to small perturbations. The OPTMARGIN method creates a surrogate model of the region classifier, which classifies a smaller number of perturbations. $f$ is the point classifier; $v_i$ is perturbation applied to the original data $x$. $f_i(x) = f(x + v_i)$; they define a loss term for each model:

$$l_i(x') = l(x' + v_i). \tag{9}$$

The minimization problem of one $l(x')$ is

$$\text{minimize} \left\|x' - x\right\|_2^2 + c \cdot (l_1(x') + \cdots + c \cdot (l_n(x'). \tag{10}$$

### 5.2. Datasets

MNIST (http://yann.lecun.com/exdb/mnist/): we use MNIST [43] for our experiments, which consists of 60000 training images and 10000 test images, all drawn from the same distribution. All these black and white digits are size-normalized and centered in a fixed-size image, where the center of gravity of the intensity lies at the center of the image with $28 \times 28$ pixels.

CIFAR10 (https://www.cs.toronto.edu/kriz/cifar.html): the CIFAR10 dataset consists of 60000 $32 \times 32$ color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images.

Data separation: in our experiments, we randomly select 5000 records $X$ as the original instances used to generate adversarial examples$X_{\text{adv}}$ and 5000 records $X_{\text{test}}$ as test data for training the simulator. We randomly select the other 5000 $X_{\text{infer}}$ for the inference evaluator. The evaluated data is expressed as $X \cup X_{\text{infer}}$.

*5.3. Performance of Our Defense Framework.* Table 2 shows the performance of our defense framework against MIA. We analyze the training accuracy and test accuracy of the target model and the attack accuracy of IM model under two kinds of public datasets (CIFAR10 and MNIST). In this experiment, we exploit the AdvGAN algorithm. We find the following: (1) our defense framework can restrain the inference accuracy. Under the CIFAR10, our mitigation reduces the attack accuracy from 93.71% to 51.4%, while with the MNIST, our defense decreases the attack accuracy from 89% to 52.5%. As these attacks nearly randomly guess probability (50%), the mitigation was proven to be effective. (2) Our defense can improve the test accuracy of the target classifier. It is interesting to see that even though the training accuracy is declined after applying our defense, the test accuracy of the target is promoted. Specifically speaking, the

**Input**: user's original data $X$ for remote training; the test dataset $X_{\text{test}}$ for evaluation; $\alpha$ is the perturbation rate;
$X_{\text{adv}}$ is the dataset constructed by the adversarial examples which synthesized by the generator;
$N$ is the size of $X$;
? is the Euclidean distance threshold between the adversarial example and the original data. $m$ is the maximize number of adversarial perturbation rounds.
**Output**: $h$, $X_{\text{adv}}$
init the adversarial model h, the simulator's model $f$, the evaluation model $g$;
$S = \alpha \times N$;
random choose $(S, X) \longrightarrow X_{\text{choosed}}$;
$X = X_{\text{choosed}} \cup X_{\text{rest}}$;
$X_{\text{adv}} = \varnothing$
for($i = 0$; $i < m$; $i$++){//to generate the adversarial examples
  Train($h$);
  for($j = 0$; $j < S$; $j$++){
    $h(x^j_{\text{gen}}) \longrightarrow X^j_{\text{gen}}$;
    for($k = 0$; $k <$ size($X^j_{\text{gen}}$); $k$++) {
      $L_k = \infty$;
      $d_k = d(x^{jk}_{\text{gen}}, x^j_{\text{choosed}})$//Euclidean distance
      if($d_k \leq$ ?){
        $l_k = \text{loss}(y^j_{\text{choosed}}, f(x^{jk}_{\text{gen}}))$
        if($l_k < L_k$){
          $L_k = l_k$
          $x_{\text{return}} = x^{jk}_{\text{gen}}$
        }
      }
    }
    $x_{\text{return}} \longrightarrow X_{\text{gen}}$
    if(cross entropy($Y^j_{\text{gen}}$, $fX^j_{\text{gen}}$) > cross entropy $(Y_{\text{adv}}, f(X_{\text{adv}}))$
    $x_{\text{adv}} \longrightarrow X^j_{\text{gen}}$
  }
Return $h$, $X_{\text{adv}}$

Algorithm 1: Synthesis of the uploading data.

Table 2: Performance of our defense framework.

| Without defense | Train acc. (%) | Test acc. (%) | Attack acc. (%) | With defense | Train acc. (%) | Test acc. (%) | Attack acc. (%) |
|---|---|---|---|---|---|---|---|
| CIFAR | 10 96.1 | 62.24 | 93.71 | Defense framework (AdvGAN) | 78.3 | 69.7 | 51.4 |
| MNIST | 100 | 57.9 | 89 | Defense framework (AdvGAN) | 91.66 | 91.12 | 52.5 |

test accuracy with CIFAR10 increases from 62.24% to 69.7%, while the test accuracy with MNIST increases from 57.9% to 91.12%.

*5.4. Comparison with Existing Methods.* We illustrate the comparison of our method with existing methods including min-max [7] and differential privacy [16]. We use CIFAR 10 as the dataset; and we use the AdvGAN algorithm and set $\epsilon = 50$ in differential privacy mitigation; the distance of our method is 23. Table 3 demonstrates that all the three defenses can restrain the MIA accuracy close to 50%, after 50 epochs. Among these three mitigations, the inference accuracy under differential privacy is the lowest. However, the training accuracy and the test accuracy are just at 1.2% and 1%, respectively. Under the min-max method, the inference accuracy, the training accuracy, and the test accuracy are 52.9%, 68.6%, and 62.7%, respectively. Under our method, the training accuracy and the test accuracy are 51.94%, 78.3%, and 69.7%. Our method achieves MI privacy with the minimum utility cost of the target classifier.

### *5.5. Different Adversarial Algorithm-Based Privacy-Preserving Capability Evaluation*

*5.5.1. Protect Content Disclosed Attack.* We visualize MNIST outputs with different adversarial algorithms with our defense framework. As demonstrated in Figure 3, the AdvGAN-based method produces the clearest records compared to the FGSM and OPTMARGIN-based ones. Examples synthesized by the FGSM-based method are more noticeable, causing the content of these outputs to be unrecognizable. This attribute can be used to protect the sensitive

Table 3: Comparing our method with existing mitigation.

| Defenses | Train acc. (epoch = 50) (%) | Test acc. (epoch = 50) (%) | Inference acc. (epoch = 50) (%) |
|---|---|---|---|
| Min-max (CIFAR 10) | 68.6 | 62.7 | 52.9 |
| Differential privacy (epsilon = 50 CIFAR 10) | 1.2 | 1 | 50.00 |
| Framework-AdvGAN (distance = 23 CIFAR 10) | 78.3 | 69.7 | 51.94 |

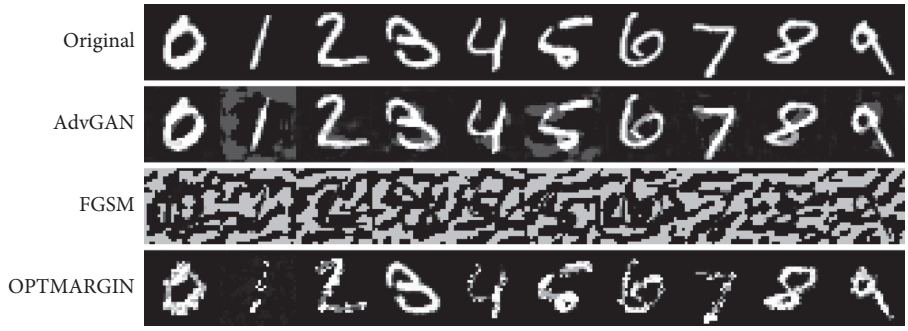



Figure 3: Visualization of adversarial outputs. We visualize outputs with different adversarial algorithms with our defense framework.

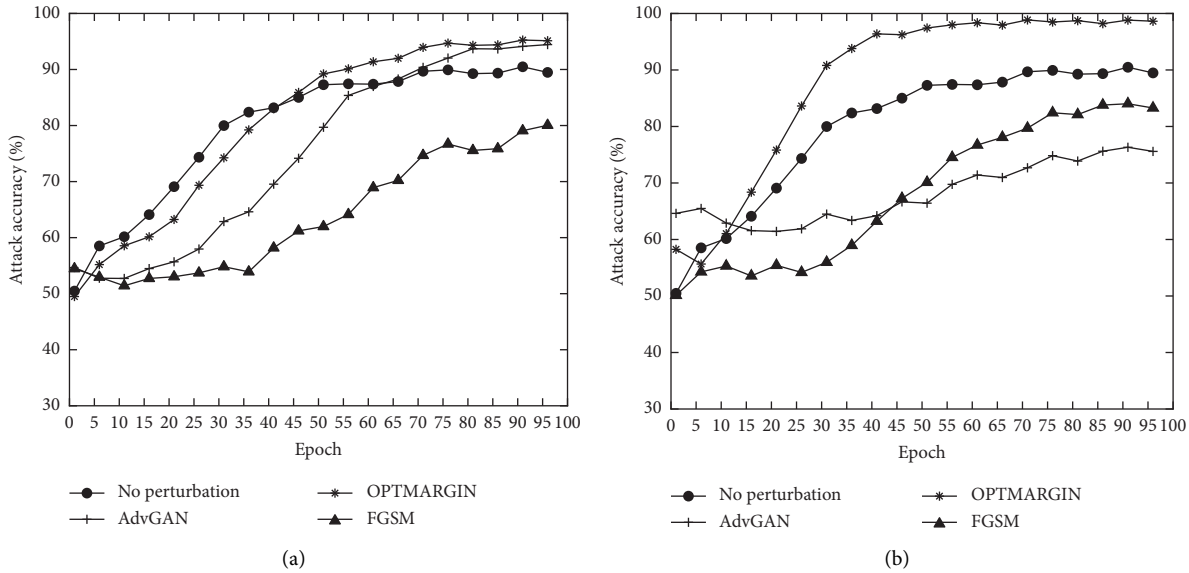



Figure 4: The defense performance with the different adversarial methods. (a) Without constraint conditions. (b) With constraint conditions.

information from the direct content disclosed risks coming from the curious model provider platform.

*5.5.2. Privacy-Preserving.* Figure 4 shows the MIA accuracy of various adversarial construction methods: AdvGAN, OPTMARGIN, and FGSM. In Figure 4(a), the adversarial instances are generated without constraint conditions introduced in 4.2. We run this experiment with MNIST and set the distance threshold $\epsilon$ as 100. We let the distortion rate $\alpha$ be 100%; in other words, we train the target model with 100% adversarial instances. In Figure 4(b), the adversarial examples are synthesized by our defense framework (with constraint conditions introduced in 4.2). The result indicates that our constrained conditions can optimize the defense performance of the AdvGAN- and FGSM-based mitigation. However, the defense effectiveness of OPTMARGIN-based defense is not satisfying. The reason might be that the parameter of our defense framework should be adjusted with a different algorithm. The relationship between parameters and the defense performance will be further discussed in 5.6.

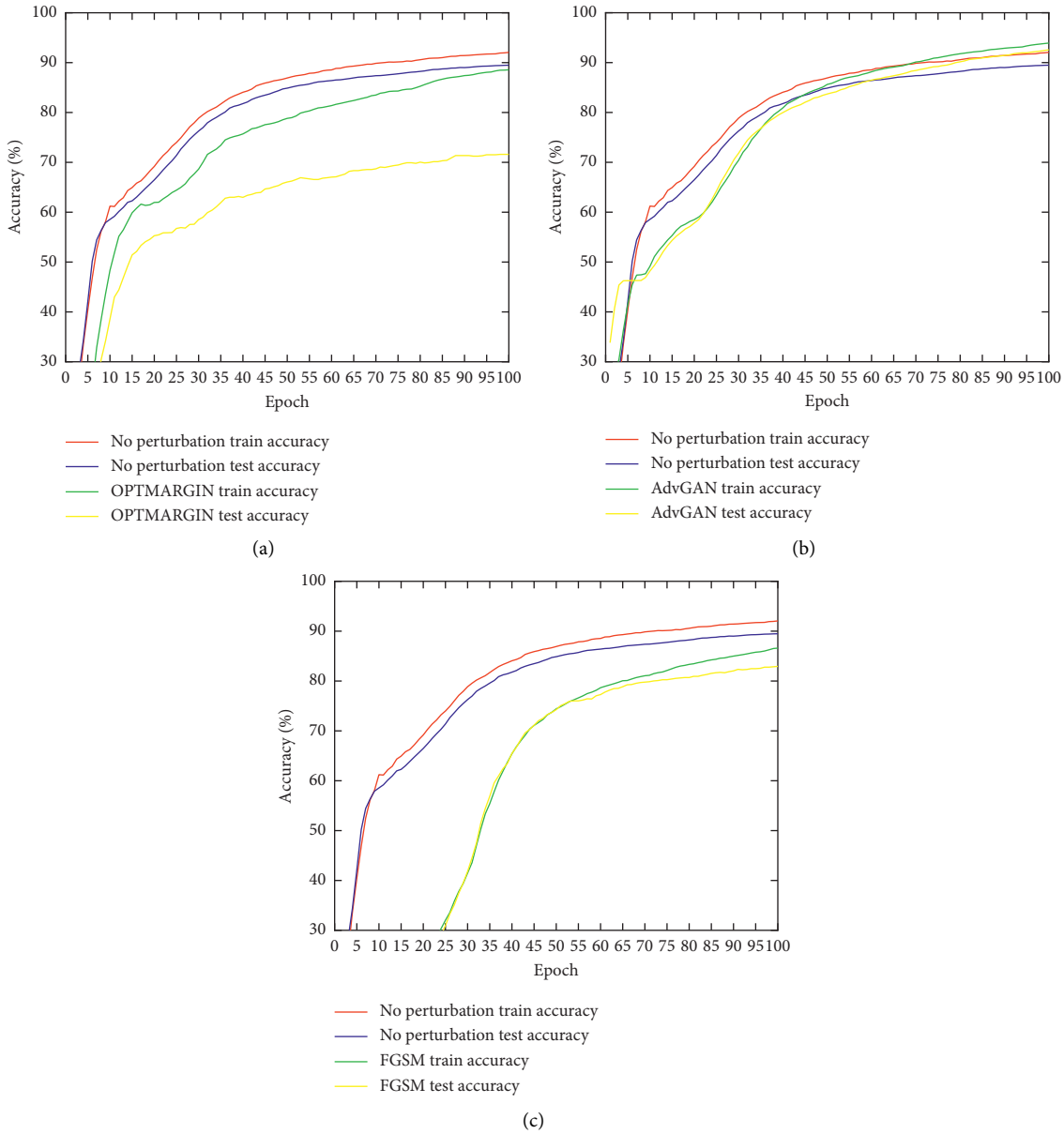



Figure 5: Comparing the training/test accuracy of original training data (no perturbation) with adversarial perturbation training data (OPTMARGIN, AdvGAN, and FGSM). (a) OPTIMARGIN perturbation with constraint condition. (b) AdvGAN perturbation with constraint condition. (c) FGSM perturbation with constraint condition.

*5.5.3. Evaluation of Privacy and Utility.* In Figure 5, we compare the training accuracy and test accuracy of the target classification model with different adversarial algorithms. We run this experiment with MNIST and set the distance threshold $\epsilon$ as 100. We let the distortion rate $\alpha$ be 100%; in other words, we train the target model with 100% adversarial instances. As we can see, the OPTIMARGIN method has the lowest test accuracy, but its training accuracy is higher than that of FGSM. The AdvGAN algorithm has the highest training accuracy and test accuracy after 100 epochs; these values are even bigger than the value without defense. The FGSM algorithm has the coincident training accuracy and test accuracy, and, after 100 epochs, values are close to the without defense ones.

*5.6. Analyze the Influence Factors.* We analyze the influence factors of our generic defense framework, including Euclidean distance and perturbation rate. We exploit the AdvGAN as the adversarial algorithm, and the original training data is MNIST. We set the training size to 5000 and the shadow model number is 5.

*5.6.1. Distance Threshold.* Figure 6 shows that, under different Euclidean distance threshold, privacy risks vary. The perturbation rate = 100%. We output one MIA accuracy for every five epochs in the 100 epochs. As shown in Figure 6, when the distance = 50, after 45 epochs, the attack accuracy area converges and is between 52% and 54%. When the

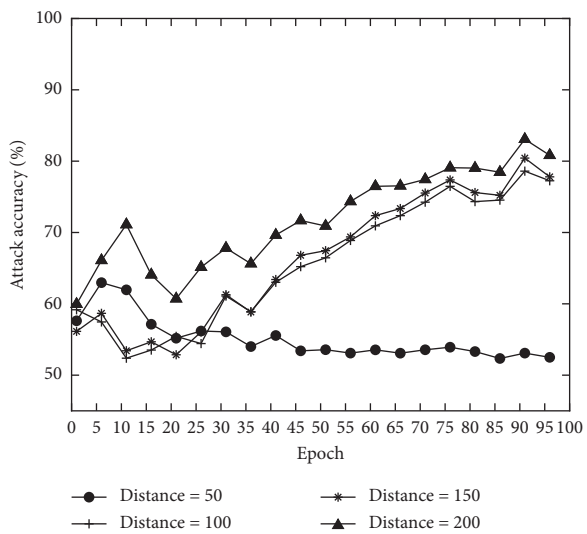



Figure 6: MIA accuracy under different Euclidean distance.

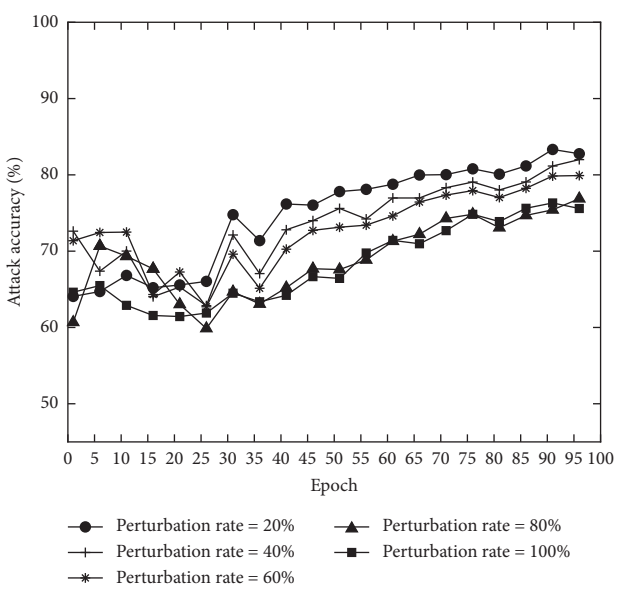



Figure 7: MIA accuracy under different adversarial perturbation rate.

distance = 200, 150, and 100, attack accuracy increases gradually between the 1st and 10th epochs, decreases between the 10th and 20th epochs, and increases slowly thereafter. According to Figure 6, the greater the Euclidean distance, the higher the MIA accuracy, the greater the risk of a privacy breach. In Euclidean distance, the smaller the distance, the better the defense. In particular, a value of 50 is the best defense against Mia, accuracy close to the random selection result (50%).

*5.6.2. Distortion Rate.* The Euclidean distance = 100. Figure 7 shows that we output one MIA accuracy for every five epochs in the 100 epochs. As shown in Figure 7, MIA accuracy is lower when perturbation rate = 100% compared to when distortion rate = 80%, 60%, 40%, and 20%. That is, the least amount of private information is disclosed. MIA accuracy is the highest when the perturbation rate is 20%. It means that the most private information is leaked. As you can see from Figure 7, the greater the perturbation rate is, the less privacy is exposed. The smaller the distortion rate is, the more privacy is exposed.

## 6. Conclusion and Future Work

We have designed and implemented a defense framework for privacy-preserving in remote machine learning services. We aim at devising a framework that not only can mitigate the privacy risk, that is, MIAs, but also can protect the training data from direct content exposed attacks, with the different adversarial algorithm. The evaluation result shows that our defense framework with the AdvGAN method is effective against MIA and our defense framework with the FGSM method can protect the sensitive original data from direct content exposed attacks. In addition, our method can achieve more privacy and utility compared to the existing method, for instance, the min-max method and the differential privacy. Furthermore, the experiment results show that we can control the defense performance with parameters such as distortion rate and distance threshold.

In this paper, we just evaluate the performance of our defense framework against MIA. Thus, exploring our defense framework against other privacy threats, such as model inversion, is left as an open question. We should do more comprehensive investigating and refining our countermeasures for future work.

## Data Availability

No data were used to support this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[2] A. Hannun, C. Case, J. Casper et al., "Deep speech: scaling up end-to-end speech recognition," 2014, https://arxiv.org/abs/1412.5567.

[3] A. Esteva, A. Robicquet, B. Ramsundar et al., "A guide to deep learning in healthcare," *Nature Medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[4] B. Krollner, B. J. Vanstone, and G. R. Finnie, "Financial time series forecasting with machine learning techniques: a survey," in *Proceedings of the 18th European Symposium on Artificial Neural Networks (ESANN 2010)*, Bruges, Belgium, April 2010.

[5] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models,"

in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, IEEE, San Jose, CA, USA, May 2017.

[6] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: evaluating information leakage of generative models using generative adversarial networks," 2018, https://arxiv.org/abs/1705.07663.

[7] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 634–646, Toronto Canada, October 2018.

[8] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: defending against black-box membership inference attacks via adversarial examples," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 259–274, London, UK, November 2019.

[9] Y. Wang, C. Wang, Z. Wang et al., "Mcmia: model compression against membership inference attack in deep neural networks," 2020, https://ui.adsabs.harvard.edu/link_gateway/2020arXiv200813578W/arxiv:2008.13578.

[10] F. Mirshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, and H. Esmaeilzadeh, "Privacy in deep learning: a survey," 2020, https://arxiv.org/abs/2004.12254.

[11] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, pp. 601–618, Austin, TX, USA, August 2016.

[12] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, Denver, CO, USA, October 2015.

[13] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: an end-to-end case study of personalized warfarin dosing," in *Proceedings of the 23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pp. 17–32, San Diego, CA, USA, August 2014.

[14] Y. Long, V. Bindschaedler, L. Wang et al., "Understanding membership inferences on well-generalized learning models," 2018, https://arxiv.org/abs/1802.04889.

[15] A. Salem, Y. Zhang, M. Humbert et al., "Model and data independent membership inference attacks and defenses on machine learning models," 2018, https://arxiv.org/pdf/1806.01246.pdf.

[16] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *Proceedings of the 28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 1895–1912, Santa Clara, CA, USA, August 2019.

[17] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *Proceedings of the 30th {USENIX} Security Symposium ({USENIX} Security 21)*, Vancouver, Canada, August 2021.

[18] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "On the connection between differential privacy and adversarial robustness in machine learning," *Stat*, vol. 1050, p. 9, 2018.

[19] Y. Long, V. Bindschaedler, and C. A. Gunter, "Towards measuring membership privacy," 2017, https://arxiv.org/abs/1712.09136.

[20] T. Salimans and D. P. Kingma, "Weight normalization: a simple reparameterization to accelerate training of deep neural networks," 2016, https://arxiv.org/abs/1602.07868.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1310–1321, Denver, CO, USA, October 2015.

[23] C. Dwork, "Differential privacy: a survey of results," in *Proceedings of the International Conference on Theory and Applications of Models of Computation*, pp. 1–19, Springer, Changsha, China, May 2008.

[24] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pp. 371–380, Bethesda, MD, USA, May 2009.

[25] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: privacy via distributed noise generation," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology–EUROCRYPT 2006*, pp. 486–503, Springer, St. Petersburg, Russia, May-June 2006.

[26] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 51–60, IEEE, Vancouver, Canada, December 2010.

[27] C. D, work and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[28] S. Rahimian, T. Orekondy, and M. Fritz, "Differential privacy defenses and sampling attacks for membership inference," in *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), PriML Workshop (PriML)*, vol. 13, Vancouver, Canada, December 2019.

[29] A. Triastcyn and B. Faltings, "Generating differentially private datasets using gans," 2018, https://www.arxiv-vanity.com/papers/1803.03148/.

[30] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," 2016, https://arxiv.org/abs/1610.05755.

[31] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The secret sharer: measuring unintended neural network memorization & extracting secrets," 2018, https://arxiv.org/abs/1802.08232.

[32] Z. Yang, B. Shao, B. Xuan, E.-C. Chang, and F. Zhang, "Defending model inversion and membership inference attacks via prediction purification," 2020, https://arxiv.org/abs/2005.03915.

[33] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, IEEE, San Jose, CA, USA, May 2017.

[34] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, https://arxiv.org/abs/1412.6572.

[35] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, https://arxiv.org/abs/1607.02533.

[36] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, https://arxiv.org/abs/1706.06083.

[37] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1765–1773, Honolulu, HI, USA, July 2017.

[38] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, Las Vegas, NV, USA, June 2016.

[39] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, IEEE, Saarbrucken, Germany, March 2016.

[40] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," 2017, https://arxiv.org/abs/1710.11342.

[41] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," 2018, https://arxiv.org/abs/1801.02610.

[42] W. He, B. Li, and D. Song, "Decision boundary analysis of adversarial examples," in *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada, April-May 2018.

[43] Y. LeCun, "The MNIST database of handwritten digits," 1998, http://yann.lecun.com/exdb/mnist/.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9930608, 9 pages
https://doi.org/10.1155/2021/9930608

*Research Article*

# Augmented Data Selector to Initiate Text-Based CAPTCHA Attack

**Aolin Che,[1] Yalin Liu,[1] Hong Xiao,[2] Hao Wang,[3] Ke Zhang,[4] and Hong-Ning Dai[1]**

[1]*Macau University of Science and Technology, Macau, China*
[2]*Guangdong University of Technology, Guangzhou 510006, China*
[3]*Department of Computer Science in Norwegian University of Science and Technology, Gjøvik, Norway*
[4]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China*

Correspondence should be addressed to Hong Xiao; wh_red@gdut.edu.cn

Received 4 March 2021; Accepted 31 May 2021; Published 16 June 2021

Academic Editor: Ting Chen



In the past decades, due to the low design cost and easy maintenance, text-based CAPTCHAs have been extensively used in constructing security mechanisms for user authentications. With the recent advances in machine/deep learning in recognizing CAPTCHA images, growing attack methods are presented to break text-based CAPTCHAs. These machine learning/deep learning-based attacks often rely on training models on massive volumes of training data. The poorly constructed CAPTCHA data also leads to low accuracy of attacks. To investigate this issue, we propose a simple, generic, and effective preprocessing approach to filter and enhance the original CAPTCHA data set so as to improve the accuracy of the previous attack methods. In particular, the proposed preprocessing approach consists of a data selector and a data augmentor. The data selector can automatically filter out a training data set with training significance. Meanwhile, the data augmentor uses four different image noises to generate different CAPTCHA images. The well-constructed CAPTCHA data set can better train deep learning models to further improve the accuracy rate. Extensive experiments demonstrate that the accuracy rates of five commonly used attack methods after combining our preprocessing approach are 2.62% to 8.31% higher than those without preprocessing approach. Moreover, we also discuss potential research directions for future work.

## 1. Introduction

Completely Automated Public Turing tests to tell Computers and Humans Apart, abbreviated as CAPTCHA, is a kind of test that automatically distinguishes human and robot operations. In 2003, Von Ahn et al. [1] firstly discuss the use of the dedicated-designed CAPTCHA schemes to prevent the attack from robots and thereby enhance network security. After that, researchers have invented a variety of CAPTCHA schemes, such as text-based CAPTCHAs, image-based CAPTCHAs, slider-type CAPTCHAs, and SMS CAPTCHAs. These CAPTCHA schemes are widely used in websites and mobile applications, to protect the corporations and users' passwords and data [2–6]. As shown in Figure 1, we present a statistical pie graph to count the types of the commonly used CAPTCHAs from 100 popular websites. According to our survey, nearly 55% of websites use text-based CAPTCHAs as security and identification mechanisms, far exceeding other types, due to the low development and maintenance costs. For this reason, we mainly focus on the study of text-based CAPTCHAs in this paper.

A critical design point of the text-based CAPTCHA is to prevent different computer-based attacks from recognizing the text information in CAPTCHAs, because these attacks may cause serious data leakages and economic losses for the enterprises and users [7]. To achieve this goal, researchers design many encryption methods to hinder computer recognition according to the structure of the text-based CAPTCHA. Generally, a text-based CAPTCHA image consists of three layers [8], as illustrated in Figure 2. The foreground layer ordinarily has some occluding lines, noisy spots, and other interference elements. The character layer contains characters that human users can recognize. Designers usually add some extra antisegmentation and anti-recognition characteristics, such as varied typeface, overlapping, rotation, distortion, and waving on text-based CAPTCHAs. The background layer is usually a

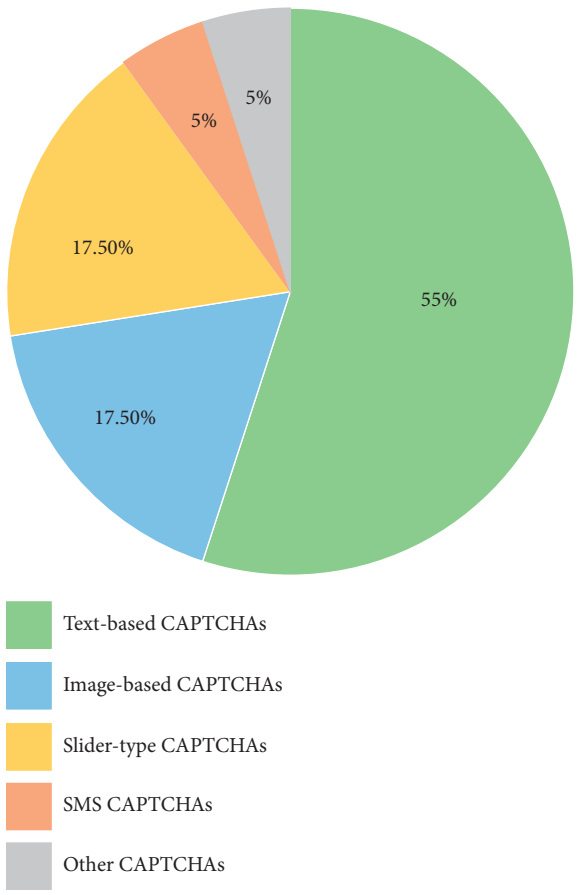



Figure 1: A statistical pie graph of the CAPTCHA types from 100 popular websites with each of them possessing more than 1 million daily visits. The chosen websites are very comprehensive, including varieties of practically applied fields (e.g., search engines, corporate websites, game websites, media-sharing sites, brand-building sites, and school websites).

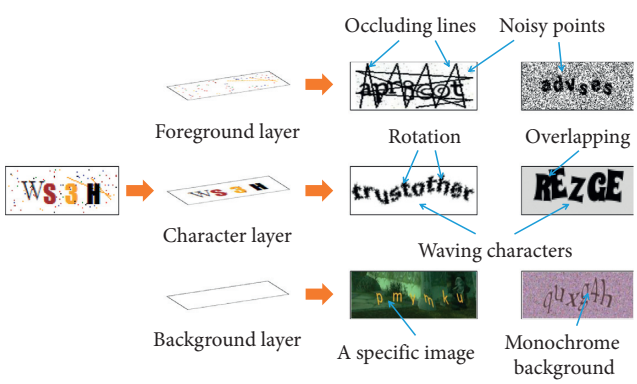



Figure 2: An example of text-based CAPTCHAs (reproduced from [8]). Generally, a text-based CAPTCHA comprises three layers: foreground layer, character layer, and background layer.

monochrome background board or a specific image that can interfere with the recognition of characters by computers.

A good attack method can identify the text-based CAPTCHA characters after adding multiple complex security features. In recent years, researchers have proposed different attack methods to solve text-based CAPTCHAs [9, 10]. One preferred attack method is using machine learning to conduct a computer-based recognition of text-based CAPTCHAs [11]. In addition, as an evolution of machine learning, deep learning is capable of making accurate decisions in the field of image recognition [12]. For this reason, recent research efforts began to explore deep learning-based attack methods to solve text-based CAPTCHAs with a high-accurate accuracy rate [7, 8]. However, the previous studies are often based on the massive volumes of unlabeled training data (e.g., millions of samples [13]), thereby bringing extra costs which are labor-intensive and time-consuming to label all training data. In fact, not all data is meaningful for training because the training on some data does not significantly improve the accuracy of the attack [14, 15]. It is not worth wasting a lot of time for us to manually label these types of data.

In this paper, we propose a preprocessing approach to effectively select CAPTCHA images so as to improve the recognition rate of machine/deep learning-based CAPTCHA attack methods. Our preprocessing approach can significantly improve the accuracy rate of commonly used attack methods. In particular, our proposed preprocessing approach contains two preprocessing modules: data selector and data augmentor. In detail, the data selector module selects some specific text-based CAPTCHAs from the original CAPTCHA data set as the training set according to the given regulations. Then, the data augmentor module randomly adds noises for each training data to obtain a higher target accuracy rate. We evaluate the proposed preprocessing approach by integrating it into five commonly used machine learning-based attack methods. These commonly used attack methods include convolutional neural network (CNN), support vector machine (SVM), decision tree (DT), random forest (RF), and logistic regression (LR). The experimental results show that, in the case of 3,319 original text-based CAPTCHAs, the accuracy rates of the five attack methods after adding our preprocessing approach are from 2.62% to 8.31% higher than the accuracy rates without adding the preprocessing module. Our experiment proves that the accuracy of text-based CAPTCHA attacks can be significantly improved by strengthening preprocessing, which however is ignored in existing studies.

In summary, this paper makes the following contributions:

(1) We present a simple and generic data-preprocessing approach to enhance the accuracy and efficiency of the text-based CAPTCHAs attacking process that utilizes machine/deep learning methods. Our preprocessing approach is composed of two modules: (i) a data selector to efficiently filter out CAPTCHAs with training significance and (ii) a data augmentor to improve the accuracy of the attack method.

(2) We perform comprehensive ablation experiments by combining our approach in five commonly used machine/deep learning methods, including CNN, SVM, DT, RF, and LR. The experiment results confirm that our approach can significantly improve

the high accuracy of the general attack methods, indicating the wide applicability of our approach.

The remainder of this paper is organized as follows: Section 2 overviews the commonly used attack methods based on machine/deep learning to break text-based CAPTCHAs. Section 3 introduces the general attacking process and the detailed design of our preprocessing approach. In Section 4, we perform a set of ablation experiments on five commonly used training methods to demonstrate that our preprocessing approach can effectively improve the accuracy rate of existing attack methods. Moreover, Section 5 discusses the future research related to man-machine verification schemes based on the current text-based CAPTCHAs. Section 6 concludes this paper.

## 2. Previous Attacking Methods

In recent years, relying on the huge magnitude of data, many training model-based attack methods are presented to recognize text-based CAPTCHAs. We can roughly divide them into two categories: machine learning and deep learning attack methods.

Computer security has received a growing interest recently in diverse computer-based systems [16–25]. Traditional machine learning methods include SVM, DT, and RF, which are widely used to break text-based CAPTCHAs. Mori and Malik first attempt to attack the text-based CAPTCHA in 2003 [26]. They obtained a 33% accuracy rate on a text-based CAPTCHA called Gimpy by exploring sophisticated object recognition algorithms. Since then, researchers have proposed various attack algorithms for text-based CAPTCHAs [27]. The most representative work is that Bostik and Klecka compared five traditional machine learning methods to attack the character bubble CAPTCHA [28]. Their experimental results show that SVM had the highest accuracy rate but DT had the lowest computational cost. It is worth noting that these machine learning-based attack methods rely heavily on character segmentation. Once a character cannot be completely segmented, the accuracy rate of the machine learning-based attack method will be greatly reduced. Therefore, we summarize the limitations of using machine learning-based attack methods: (1) it is necessary to design a complex character segmentation algorithm so that the preprocessing is extremely cumbersome; (2) it needs to manually find the characteristics of each character, consequently resulting in a relatively low accuracy rate.

Deep learning-based attack methods are mainly attributed to convolutional neural networks. In contrast to traditional machine learning methods, convolutional neural networks can automatically obtain the underlying feature information of the picture through continuous convolution and pooling, and the extracted features are far more than traditional machine learning methods. Substantial studies have proved that convolutional neural networks are more suitable for image recognition than the traditional machine learning methods [9–13, 29–33]. Consequently, Gao et al. [10] took the lead in using CNN to train the segmented Hollow CAPTCHAs. Their model obtained an 89% accuracy rate on the test set, and the fastest success recognition speed was 3 seconds per text-based CAPTCHA. Among incumbent deep learning-based attacks, the accuracy rate still depends heavily on the quality of the character segmentation. As text-based CAPTCHAs designed by researchers become more complex, traditional segmentation algorithms have been unable to completely separate each character. To tackle this deficiency, Ye et al. utilized a generative adversarial network (GAN) to automatically generate a large number of labeled text-based CAPTCHAs as a training set [7, 14, 15]. For some types of text-based CAPTCHAs, this method only needs to mark about 500 original text-based CAPTCHAs to obtain a recognition rate of more than 90%.

## 3. Our Approach

*3.1. The General Attacking Process.* As illustrated in Figure 3, to break text-based CAPTCHAs by using machine/deep learning methods, the general attacking process usually requires the following steps.

*3.1.1. Data Acquisition.* This step is used to acquire the original data set, which is practically achieved by crawling a large number of text-based CAPTCHAs from real-world websites. To ensure comprehensive data samples, the acquired text-based CAPTCHAs need to have characteristics including distortion, rotation, waving, and overlapping.

*3.1.2. Data Preprocessing.* Generally speaking, data preprocessing is used to filter the original data according to a dedicated filtering operation, thus improving the efficiency of the later training process. In this paper, we present a new preprocessing approach that can also achieve a high accuracy rate of the attacking process. The detailed method design is shown in Section 3.2.

*3.1.3. Model Training.* This step is using the machine/learning training methods to generate a prediction model for recognizing text-based CAPTCHAs. Herein, we choose five commonly used machine/deep learning methods: CNN, SVM, DT, RF, and LR. Notably, CNN is a popular deep learning algorithm that can provide a relatively high accuracy rate for identifying CAPTCHA images. Particularly, in our experiment, we put every 200 images as a training batch into CNN for training after binarization. Table 1 shows the architecture and parameters of the CNN used in our experiment. In addition, we also use four machine learning methods including SVM, DT, RF, and LR in the experiment. For these methods, researchers need to artificially search some character features for computers to learn according to the characteristics of different characters.

*3.1.4. Prediction Outputting.* A prediction model is generated after the model training process, which can be used to identify the unlabeled test set. Thereby, the prediction model can output a recognition result for each test CAPTCHA.

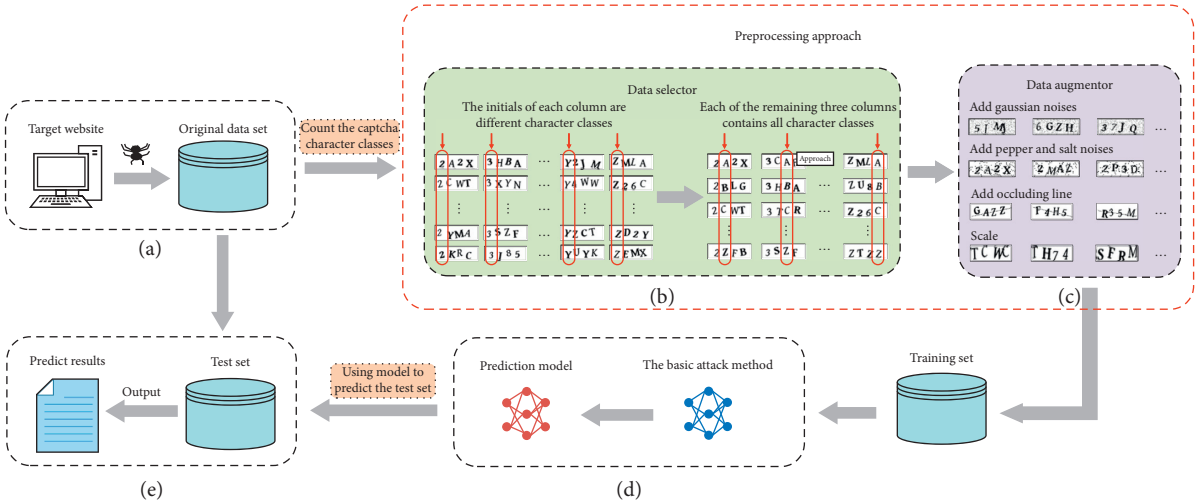



Figure 3: The general attacking process based on machine/deep learning methods to break text-based CAPTCHAs.

Table 1: Architecture of the training model.

| Type/stride | Filter shape | Input size | Output size |
|---|---|---|---|
| Conv2D_1/S1 | $3 \times 3 \times 32$ | $200 \times 60 \times 160 \times 1$ | $200 \times 58 \times 158 \times 32$ |
| MaxPool2D_1/S2 | $2 \times 2$ | $200 \times 58 \times 158 \times 32$ | $200 \times 29 \times 79 \times 32$ |
| Conv2D_2/S1 | $5 \times 5 \times 64$ | $200 \times 29 \times 79 \times 32$ | $200 \times 25 \times 75 \times 64$ |
| MaxPool2D_2/S2 | $2 \times 2$ | $200 \times 25 \times 75 \times 64$ | $200 \times 12 \times 37 \times 64$ |
| Conv2D_3/S1 | $5 \times 5 \times 128$ | $200 \times 12 \times 37 \times 64$ | $200 \times 8 \times 33 \times 128$ |
| MaxPool2D_3/S2 | $2 \times 2$ | $200 \times 8 \times 33 \times 128$ | $200 \times 4 \times 16 \times 128$ |
| Flatte | — | $200 \times 4 \times 16 \times 128$ | $200 \times 8192$ |
| Dense | — | $200 \times 8192$ | $200 \times 128$ |
| Reshape | — | $200 \times 128$ | $200 \times 4 \times 32$ |

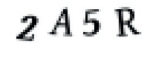
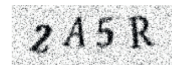
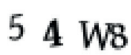
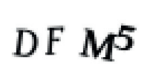
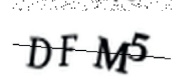
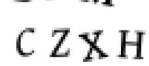
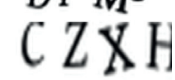
Then, we count whether the prediction result given by the prediction model is consistent with the characters on the corresponding CAPTCHA. Finally, we give the accuracy rate of the prediction model on the test set.

*3.2. Our Preprocessing Approach.* We can clearly see that, in a general attacking process, not all text-based CAPTCHAs in the original data samples can affect the final accuracy rate. This motivates us to choose only a small number of training data that have a significant influence on prediction results and can also lead to a high accuracy rate. To achieve this goal, we design a new preprocessing approach composed of two modules: (i) a data selector that invokes a set of regulations to filter out a high-quality text-based CAPTCHA training set and (ii) a data augmentor that is used to expand the data set and enhance the robustness of the prediction model.

*3.2.1. Data Selector.* We aim to design a data selector that can generate a clean and effective training set by filtering the original data set, thereby improving the prediction accuracy of the whole model. For the detailed design of the data selector, we formulate a set of regulations that can filter out a high-training quality training set of text-based CAPTCHAs. The filtering regulations of the data selector are also shown in Part B of Figure 3. First, we count the character classes on the original CAPTCHA image set. There are a total of 32 character classes: 2 ~ 9, A ~ Z except for "O" and "I". That is to say, each CAPTCHA on the website is composed of four characters (repeatable) that are randomly selected from the 32 character classes. In this case, each class of characters can appear with the probability of 1/32 in one bit of the text-based CAPTCHA string. Secondly, we divide the training set pictures into 32 columns in alphabetical order and arrange the images with the same first CAPTCHA character into one column. For the remaining three columns, each column must contain all character classes. This will ensure that 32 different characters in each position will be trained by the neural network.

*3.2.2. Data Augmentor.* Data augmentation is used to expand the data set, thus enhancing the robustness of the prediction model. It can also prevent the neural network from learning irrelevant features, thereby fundamentally improving the overall performance of the model. We propose a data augmentor using four different image noises such as Gaussian noise, pepper and salt noise, scale, and occluding line. Table 2 shows the comparison of the four types of noise images with the original image. In our experiment, the data augmentor randomly allocates an augmentation method for each picture or decides not to perform any processing. Then, the data augmentor automatically puts the noise-added images into CNN for

Table 2: Comparison after adding different noises.

| Scheme | Original example | Noised example |
|---|---|---|
| Gaussian noise |  |  |
| Pepper and salt noise |  |  |
| Occluding line |  |  |
| Scale |  |  |

training. The workflow of the data augmentor is illustrated in part C of Figure 3. Each image noise is described in detail as follows:

(i) Gaussian noise: Gaussian noise is named after Carl Friedrich Gauss because its probability density function (PDF) is equal to that of the normal distribution. In our experiment, we first use the `random.randn()` function in Python to randomly return an array with a standard normal distribution. Then, we multiply the array by the Gaussian noise probability to generate a noise matrix. Finally, we add the noise matrix to the matrix of the normal text-based CAPTCHA to generate a Gaussian noise image. The function of Gaussian noise can be expressed as

$$G = I + N_g, \tag{1}$$

where $G$ denotes the noisy image, $I$ denotes the unprocessed image, and $N_g$ represents the Gaussian noise with a variance $\sigma = 50$ and a mean value of 0.

(ii) Pepper and salt noise: in essence, both pepper and salt noises are some black and white pixels that randomly appear on images. Salt noise is also called white noise and pepper noise is also called black noise. Concretely, in our method, we randomly select some pixels on the image and change the pixels' RGB values to the $(255, 255, 255)$ or $(0, 0, 0)$ colour on the standard RGB colour comparison table. The function of pepper (or salt) noise can be expressed as

$$P = I + N_p, \tag{2}$$

where $P$ denotes the noisy image, $I$ means the unprocessed image, and $N_p$ represents the pepper (or salt) noise with a noise probability of 0.5.

(iii) Occluding line: we implement a method to generate an occluding line. The principle of this method is shown in Figure 4. The formula of this method is shown in equation (3). First, we assume that the coordinates of the left and right endpoints $A$ and $B$ of the occluding line are $A(x_1, x_2)$ and $B(x_2, y_2)$, respectively. Furthermore, we found that when $A_x \in (0, 10)$, $A_y \in (10, 50)$ and $B_x \in (150, 160)$, $B_y \in (10, 50)$, the occluding line can cross the four characters on the CAPTCHA image. Therefore, we use the function to generate the endpoints of the occluding line. Then, we call the function `draw.line()` to generate a line segment linking the two endpoints $A$ and $B$. The function of generating an occluding line is defined as

$$L = I + N_l, \tag{3}$$

where $L$ means the observed image of occluding line, $I$ denotes the unprocessed image, and $N_l$ represents the occluding line.

(iv) Scale: the fourth data augmentation method used in data augmentor is scale. We carefully analyze the characteristics of the original CAPTCHA images then proposed an image scale method. First, we translate a $60 \times 160$ size image to the left by 10 pixels and then upwards by 10 pixels. Secondly, we call the pan and zoom functions to adjust the image to $50 \times 140$. Finally, we call the resize function to restore the image to $60 \times 160$. It can be seen from the example in Table 2 that, compared with the original CAPTCHA image, the characters after enlarged still retain their basic character features. This method can effectively prevent the neural network from learning irrelevant or nonkey features of characters, fundamentally improving the accuracy of the model. In particular, we have

$$S = \Phi(I), \tag{4}$$

where $S$ denotes the observed image of occluding line, $\Phi$ means the scale operator, and $I$ represents the unprocessed image.

## 4. Experiments

In this section, we perform the complete attacking experiment by using our approach presented in Section 3. We particularly conduct a set of ablation experiments by applying our data-preprocessing approach in different training methods, including CNN and four commonly used machine learning methods.

### 4.1. Experiment Settings

*4.1.1. Data Preparation.* We used the crawled 9,955 unlabeled original text-based CAPTCHAs from E-ZPass New York website as the attack target of this experiment. The

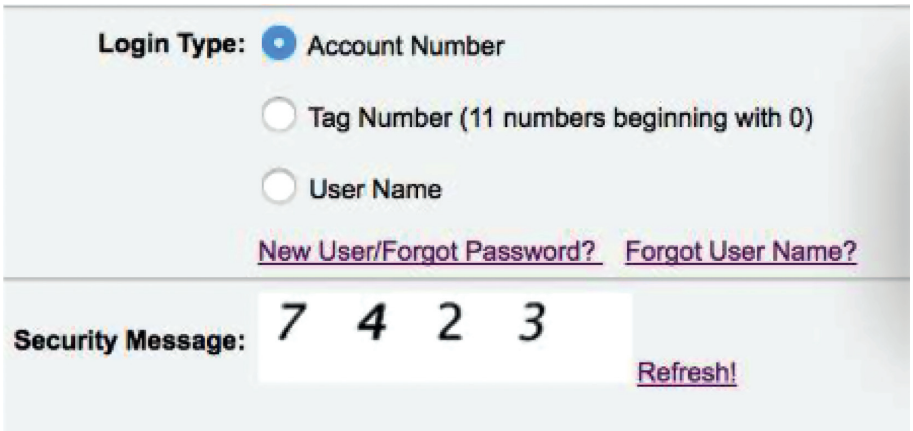



Figure 4: The text-based CAPTCHA of the E-ZPass New York login interface.

principle of this method is shown in Figure 5. The characteristics of the CAPTCHA of this website are as follows: no extra interference noise, the characters are distorted, and only two visual colours of white and black. These features significantly reduce our workload because it saves us the tedious preprocessing step of denoising. It is worth noting that similar characters may cause a negative impact on the success rate of recognition, such as "0" and "O", "I" and "1". To avoid this predicament, we manually delete the pictures containing these four characters in the web crawler data set. Therefore, the original CAPTCHA image set contains a total of 32 character classes: 2–9, A–Z except for "O" and "I".

*4.1.2. Implementation Details.* Our implementation environment is PyCharm 2020.2.2, the programming language is Python 3.7, and we test it on a laptop with 1.6 GHz Internet core i5-8250CPU, 4 GB RAM, and Windows $10 \times 64$. Meanwhile, we uniformly use `findContours()` function to segment training images, which is built-in OpenCV 4.1. For CNN, we use TensorFlow 2.0 framework to write the training code.

*4.1.3. Evaluation Metric.* To evaluate the quality of text-based CAPTCHA attacks, we use a metric of the prediction accuracy, that is, the accuracy rate ACC. A prediction model with a higher accuracy rate will have an accuracy value closer to 100%. We can use the following formula to calculate the accuracy rate:

$$\mathrm{ACC} = \frac{K}{T} \times 100\%, \tag{5}$$

where $K$ represents the number of text-based CAPTCHAs successfully identified by the prediction model on the test set and $T$ represents the size of the test set.

*4.2. Ablation Experiments.* In our experiment, we conduct a set of ablation experiments to verify the two preprocessing approaches we propose. Figure 6 reports the accuracy rate of each basic attack method in different preprocessing


Figure 5: Principle of adding the occluding line. The CAPTCHA image size is $60 \times 160$. The abscissa represents the length, the ordinate represents the width, and the shading area represents the area where the left and right endpoints A and B of the occluding line are generated by the method.

environments. In the benchmark experiment, we input a set of unprocessed data sets into five commonly used attack methods for training. Moreover, we prepare the same amount of labeled training images and the same basic attack method for each experiment to ensure the fairness of the experiment. The performance improvement shows that our preprocessing approach is advantageous in character recognition, especially when using two preprocessing modules simultaneously.

We first add the data selector and data augmentor modules separately. The accuracy rate of the CNN-based attack method increases by 2.61% and 0.66% compared to the benchmark experiment after adding data selector and data augmentor, respectively. This shows that even adding one preprocessing module can also be advantageous in recognition. When we add these two preprocessing modules to the CNN-based attack at the same time, the accuracy rate of this model is 95.5%, which is 4.51% higher than the case without any preprocessing module. Overall, the preprocessing approach can further improve the accuracy rate of the CNN-based attack method.

To verify the applicability of the preprocessing approach, we conduct a set of ablation experiments on four machine learning-based attack methods. We choose the four most commonly used traditional machine learning methods, including SVM, DT, RF, and LR as basic training models. Then,

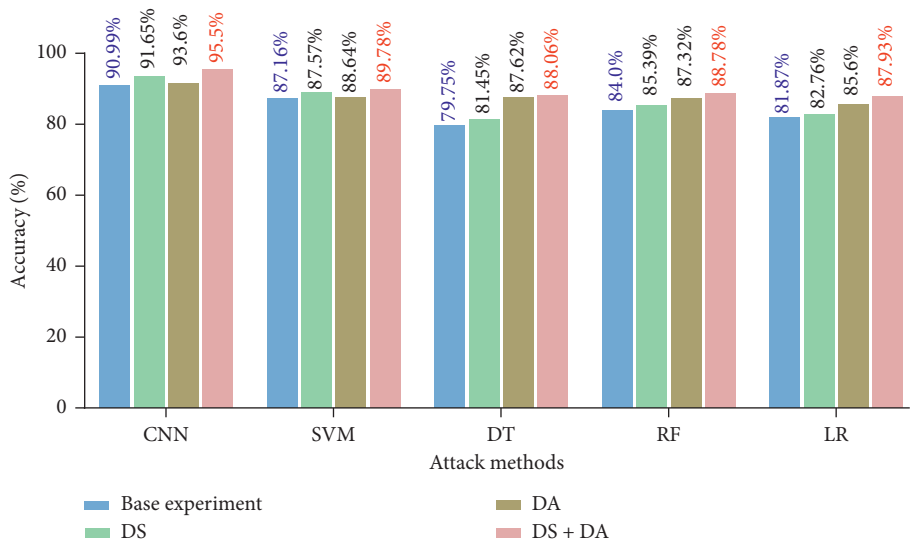



Figure 6: Combining our preprocessing approach with five commonly used attack methods, it shows that even adding a single preprocessing module can improve the accuracy of the basic attack method. Adding two attack modules at the same time can significantly improve the accuracy of each method. Furthermore, compared with machine learning methods, CNN has more advantages in text-based CAPTCHA recognition.

we do a set of ablation experiments on each of the machine learning methods to demonstrate the wide applicability of our proposed preprocessing modules. Each set of ablation experiments consists of a basic control group, an experiment adding data selector, an experiment adding data augmentor, and an experiment adding two preprocessing modules. The training sets in the basic control group contain 3,319 randomly marked original CAPTCHA images. For the remaining two experiments with adding a data selector, the module selects 3,319 CAPTCHAs according to the actual situation.

From Figure 6, we can see that after adding data selector (DS) and data augmentor (DA) modules to different attack methods, the accuracy rate has improved in varied degrees. In detail, after adding two preprocessing modules simultaneously, the accuracy rate for SVM increases from 87.16% to 89.78%, DT from 79.75% to 88.06%, RF from 84.0% to 88.78%, and LR from 81.87% to 87.93%. The extensive contrast experiments show that our preprocessing approach has strong applicability, which can be easily deployed to other attack methods.

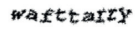
Notably, in the five commonly used attack methods, the CNN attack method has the highest accuracy rate under the same amount of training images, even in the case without adding any preprocessing modules. The fundamental reason is that the advantage of CNN lies in feature extraction. The neural network can even extract some underlying features after multiple convolutions. In the case of large-scale training data, the deep neural network can extract more features than traditional machine learning methods. Since people usually use ImageNet or other large-scale data sets in current image recognition algorithms, many traditional methods such as SVM and DT are difficult to meet practical requirements in terms of computational complexity. Therefore, when the training set is large and complex, the effect of using a neural network with low computational complexity and high parallelism is better than traditional machine learning methods. The outstanding performance of CNN shows that using CNN to identify text-based CAPTCHA is more accurate than traditional machine learning methods.

## 5. Discussion

Reviewing the current research studies of CAPTCHAs, lots of efforts are made to improve the accuracy rate of attack methods, including varieties of machine/deep learning-based training models as well as our data-preprocessing approach. With such an increasing evolution of attack methods, the more sophisticated design of CAPTCHAs is required to enhance security. The sophisticated design, however, may also cause recognition difficulties of CAPTCHAs for humans and thus leads to a poor user experience. Toward this end, how to balance the design complexity of CAPTCHAs and the user experience is a significant research topic in the future [18, 34, 35].

Taking text-based CAPTCHAs as an example, we investigate a real user experience in terms of different security strengths of CAPTCHA designs. In particular, we perform a questionnaire survey for 20 participants who are required to identify the 500 text-based CAPTCHA images that are crawled from real-world websites. These CAPTCHA images have different security strengths according to different combinations of several basic image features including noise, overlapping, rotation, distortion, and waving. Then, participants evaluate the user experience of the CAPTCHAs by giving the usability score from 1 to 5. The statistical results of our questionnaire survey are shown in Table 3, in which we consider five combinations of image features to compare five different security strengths of CAPTCHA designs. For each combination of image

Table 3: User experience against five different security strengths of text-based CAPTCHAs.

| Image examples | Image features | | | | | Humans | Usability score |
|---|---|---|---|---|---|---|---|
| | Noises | Overlapping | Rotation | Distortion | Waving | Accuracy rate (%) | |
|  | | | √ | | √ | 99 | 4.8 |
|  | | | √ | √ | √ | 93 | 3.7 |
|  | | √ | √ | √ | √ | 92 | 3.4 |
|  | √ | √ | √ | | √ | 85 | 3 |
|  | √ | √ | √ | √ | √ | 33 | 2 |

features, we count the average accuracy rate and the average usability rating given by all participants.

In Table 3, we can clearly observe that the difficulty of human identification is increasing with the increasing combination of the image features. For the CAPTCHA including only two image features (i.e., rotation and waving), it has the lowest security strength, thereby the average accuracy rate of humans can reach 99%. However, the average accuracy rate of humans is only 66% when the CAPTCHA has five image features, meaning that around one-third of users would fail identification. In addition, as the used image features increase, the usability score given by the participants gets lower and lower. This is because although complex CAPTCHA images can better prevent malicious bot attacks, it also increases the difficulty of human identification and deteriorates the user experience. All in all, to address the unbalanced problem between the security strengths and user experience, the future work is required to design a new type of text-based CAPTCHA that can effectively resist robot attacks while also being user-friendly.

## 6. Conclusion

As the most widely used CAPTCHA scheme, text-based CAPTCHA is an important security mechanism to distinguish between humans and computers. However, our attack poses a threat to real-world text-based CAPTCHA. We present a new preprocessing approach for text-based CAPTCHA attacks. The preprocessing approach includes two modules: data selector and data augmentor. First of all, the data selector module automatically selects some data with training significance as the training set in the original data set according to the rules set by us. Secondly, the data augmentor module increases the amount of training data to increase the robustness of the model. We successfully deployed this preprocessing approach on five different commonly used attack methods, and the accuracy rate was significantly improved. The key advantage of our preprocessing approach is that it requires less human involvement and for any basic attack method, by adding this preprocessing step, a higher accuracy rate can be achieved with relatively less training data for any basic attack method. Finally, we further discussed the existing problems in the current text-based CAPTCHAs design. Our work can also be applied to many practical scenarios [20, 22, 24], such as ID card character recognition, bank card number recognition, and handwritten numeral recognition. How to expand to the application field is also a part of our ongoing work.

## Data Availability

Data are available on request to the corresponding author.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: using hard AI problems for security," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 294–311, Springer, Wersaw, Poland, May 2003.

[2] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "Recaptcha: human-based character recognition via web security measures," *Science*, vol. 321, no. 5895, pp. 1465–1468, 2008.

[3] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, "Re: captchas-understanding captcha-solving services in an economic context," in *Proceedings of the USENIX Security Symposium*, vol. 10, p. 3, Washington, DC, USA, August 2010.

[4] M. Shirali-Shahreza and S. Shirali-Shahreza, "Captcha for blind people," in *Proceedings of the 2007 IEEE International Symposium on Signal Processing and Information Technology*, pp. 995–998, IEEE, Cairo, Egypt, December 2007.

[5] J. Lazar, J. Feng, T. Brooks et al., "The soundsright captcha: an improved approach to audio human interaction proofs for blind users," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2267–2276, Austin, TX, USA, May 2012.

[6] V. P. Singh and P. Pal, "Survey of different types of captcha," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 2, pp. 2242–2245, 2014.

[7] G. Ye, Z. Tang, D. Fang et al., "Yet another text captcha solver: a generative adversarial network based approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 332–348, Toronto, Canada, October 2018.

[8] S. Tian and T. Xiong, "A generic solver combining unsupervised learning and representation learning for breaking text-based captchas," in *Proceedings of the Web Conference*, pp. 860–871, Taipei, Taiwan, 2020.

[9] S. Sachdev, "Breaking captcha characters using multi-task learning cnn and svm," in *Proceedings of the 2020 4th International Conference on Computational Intelligence and Networks (CINE)*, pp. 1–6, IEEE, Kolkata, India, February 2020.

[10] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, "The robustness of hollow captchas," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 1075–1086, Berlin, Germany, November 2013.

[11] J. Yan and A. S. El Ahmad, "A low-cost attack on a microsoft captcha," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, pp. 543–554, Virginia, VA, USA, October 2008.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[13] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," 2013, https://arxiv.org/abs/1312.6082.

[14] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[15] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, p. 1, 2015.

[16] T. Chen, X. Li, X. Luo, and X. Zhang, "Under-optimized smart contracts devour your money," in *Proccedings of the 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 442–446, IEEE, Austria, TX, USA, February 2017.

[17] T. Chen, Z. Li, Y. Zhu et al., "Understanding ethereum via graph analysis," *ACM Transactions on Internet Technology*, vol. 20, no. 2, pp. 1–32, 2020.

[18] E. Bursztein, M. Martin, and J. Mitchell, "Text-based captcha strengths and weaknesses," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 125–138, Illinois, IL, USA, October 2011.

[19] T. Chen, Y. Zhang, Z. Li et al., "Tokenscope: automatically detecting inconsistent behaviors of cryptocurrency tokens in ethereum," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1503–1520, London, UK, November 2019.

[20] A. K. Sangaiah, D. V. Medhane, T. Han, M. S. Hossain, and G. Muhammad, "Enforcing position-based confidentiality with machine learning paradigm through mobile edge computing in real-time industrial informatics," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4189–4196, 2019.

[21] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5790–5798, 2020.

[22] A. K. Sangaiah, D. V. Medhane, G.-B. Bian, A. Ghoneim, M. Alrashoud, and M. S. Hossain, "Energy-aware green adversary model for cyberphysical security in industrial system," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3322–3329, 2019.

[23] W. Liang, L. Xiao, K. Zhang, M. Tang, D. He, and K.-C. Li, "Data fusion approach for collaborative anomaly intrusion detection in blockchain-based systems," *IEEE Internet of Things Journal*, 2021.

[24] A. K. Sangaiah, A. S. Rostami, A. A. R. Hosseinabadi et al., "Energy-aware geographic routing for real time workforce monitoring in industrial informatics," *IEEE Internet of Things Journal*, 2021.

[25] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational lstm enhanced anomaly detection for industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2020.

[26] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: breaking a visual captcha," in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, IEEE, Madison, WI, USA, June 2003.

[27] Y. Zi, H. Gao, Z. Cheng, and Y. Liu, "An end-to-end attack on text captchas," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 753–766, 2019.

[28] O. Bostik and J. Klecka, "Recognition of captcha characters by supervised machine learning algorithms," *IFAC-PapersOnLine*, vol. 51, no. 6, pp. 208–213, 2018.

[29] M. Tang, H. Gao, Y. Zhang, Y. Liu, P. Zhang, and P. Wang, "Research on deep learning techniques in breaking text-based captchas and designing image-based captcha," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2522–2537, 2018.

[30] A. Dionysiou and E. Athanasopoulos, "Sok: Machine vs. machine–a systematic classification of automated machine learning-based captcha solvers," *Computers & Security*, vol. 97, Article ID 101947, 2020.

[31] J. Yan, "A simple generic attack on text captchas," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2016.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[34] J. Yan and A. S. El Ahmad, "Usability of captchas or usability issues in captcha design," in *Proceedings of the 4th Symposium on Usable Privacy and Security*, pp. 44–52, Pittsburgh, PA, USA, August 2008.

[35] C. A. Fidas, A. G. Voyiatzis, and N. M. Avouris, "On the necessity of user-friendly captcha," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2623–2626, Vancouver, BC, Canada, May 2011.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9983309, 9 pages
https://doi.org/10.1155/2021/9983309

*Research Article*

# Boosting Adversarial Attacks on Neural Networks with Better Optimizer

**Heng Yin, Hengwei Zhang, Jindong Wang, and Ruiyu Dou**

*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, Henan, China*

Correspondence should be addressed to Hengwei Zhang; wlby_zzmy_henan@163.com

Received 5 March 2021; Revised 19 April 2021; Accepted 26 May 2021; Published 7 June 2021

Academic Editor: Ting Chen



Convolutional neural networks have outperformed humans in image recognition tasks, but they remain vulnerable to attacks from adversarial examples. Since these data are crafted by adding imperceptible noise to normal images, their existence poses potential security threats to deep learning systems. Sophisticated adversarial examples with strong attack performance can also be used as a tool to evaluate the robustness of a model. However, the success rate of adversarial attacks can be further improved in black-box environments. Therefore, this study combines a modified Adam gradient descent algorithm with the iterative gradient-based attack method. The proposed Adam iterative fast gradient method is then used to improve the transferability of adversarial examples. Extensive experiments on ImageNet showed that the proposed method offers a higher attack success rate than existing iterative methods. By extending our method, we achieved a state-of-the-art attack success rate of 95.0% on defense models.

## 1. Introduction

In image recognition tasks, convolutional neural networks are able to classify images with an accuracy approaching that of humans [1–4]. However, researchers have found that neural networks are also vulnerable to adversarial examples. Szegedy et al. [5] first proposed the concept of adversarial examples: images added with small perturbations, which cause neural network models to output incorrect classifications with high confidence. These adversarial perturbations are often indistinguishable to the human eyes (in other words, there is no obvious visual difference between the adversarial examples and the original images).

Adversarial attacks can be categorized into white- and black-box attacks. A variety of techniques can be used to generate adversarial examples and perform white-box attacks, depending on the model structure and corresponding parameters [6–10]. In addition, adversarial examples are generally transferable as data generated for one model may be able to fool other models. This facilitates black-box attacks, in which the structure and parameters of the model are not available, for various neural networks [11]. Goodfellow et al. [6] suggested that different models learn similar decision boundaries during the same image classification tasks and obtain similar parameters. These properties make it easier to generalize adversarial examples to different models.

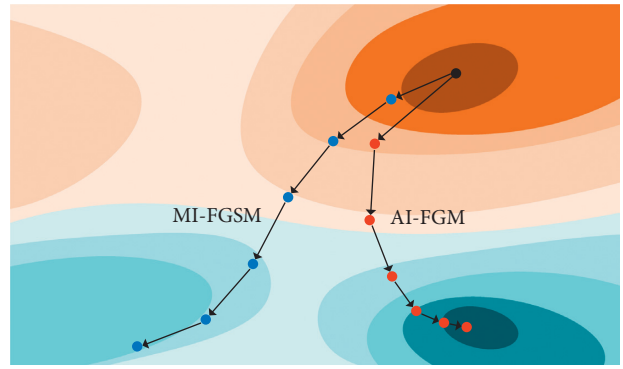
Although adversarial examples are generally transferable, the optimal approach of improving this transferability needs to be examined further. Due to a balance between attack performance and transferability, basic iterative attacks are often more effective than single-step attacks in white-box environments and weaker in black-box environments. In white-box attacks, iterative methods excessively fit specific network parameters, achieving a high success rate but preventing generalization to other models [9]. We consider that this is the result of overfitting since attack performance for adversarial examples in white- and black-box environments is similar to the neural network performance on training and test sets.

Unlike white-box attacks, black-box attacks are more consistent with actual attack-defense environments and are primarily performed in one of the following three ways. (1) Decision-based attacks are conducted using information collected from the network. Although access to model structure and parameters is unavailable, the attacker can

generate adversarial examples by executing multiple queries against the model [12–15]. (2) In substitute-model techniques, the attacker can input images into the model to obtain an output label. Then, the substitute model can be trained to imitate the target model and generate adversarial examples [16]. (3) Transferability attacks are conducted by improving the transferability of adversarial examples generated in a white-box setting [8, 9]. The first two black-box attacks require large quantities of model queries, which is impractical in some cases (e.g., online platforms often limit the number of queries). As such, this study investigates black-box attacks using adversarial data with strong transferability. We investigate a state-of-the-art optimizer Adam for improving black-box adversarial attacks.

In this paper, we propose the Adam iterative fast gradient method (AI-FGM) to improve the transferability of adversarial examples among different models [17]. Inspired by the fact that Adam is better than momentum in optimization, we adopt Adam optimizer into the iterative gradient-based attack, so as to accelerate data update in dimensions with small gradients and achieve better convergence, by applying the second momentum term and a decreasing step size. As shown in Figure 1, being different from the momentum method that just accumulates the gradients of data points along the optimization path, our Adam method can also accumulate the square of the gradients. Such accumulation might help obtain an adaptive update direction, which leads to a better local minimum. Besides, a variable step size is also used to avoid oscillating.

Compared with existing gradient-based methods, the proposed method offers improved attack performance in black-box settings. This approach was tested on multiple networks, including adversarially trained networks, achieving higher attack success rates on black-box models. In addition, we combined our approach with advanced methods and attacked an ensemble of multiple networks, which further improved the transferability of adversarial examples.




- Start point
- Data point for gradient calculation by MI-FGSM
- Data point for gradient calculation by AI-FGM

Figure 1: Schematic optimization path of MI-FGSM and the proposed AI-FGM. MI-FGSM accumulates the gradients of data points along the optimization path, while Adam accumulates the gradients and the square of the gradients. MI-FGSM adopts an invariable step size, while AI-FGM adopts a decayed step size that leads to better convergence.

## 2. Related Works

*2.1. Adversarial Attack Methods.* A variety of techniques have been proposed to generate transferable adversarial examples, which can be divided into two categories: optimizer-based methods and data-augmentation-based methods. Specifically, Goodfellow et al. proposed the fast gradient sign method (FGSM) to generate adversarial examples with very low calculation costs [6]. By extending FGSM, Alexey et al. developed the iterative fast gradient sign method (I-FGSM) [7]. Dong et al. proposed the momentum iterative fast gradient sign method (MI-FGSM) to improve the transferability of adversarial examples in black-box environments [8]. Lin et al. proposed Nesterov iterative fast gradient sign method (NI-FGSM) and adopted Nesterov accelerated gradient into the iterative attacks [18]. Xie first applied data augmentation in the generation of adversarial examples and proposed the diverse inputs method (DIM) [9]. Dong et al. proposed a translation-invariant attack (TI-FGSM) and improved the transferability of adversarial examples on defense models by a large margin [19]. Lin et al. proposed the scale-invariant attack method (SIM) and combined it with existing methods, resulting in SI-NI-TI-DIM (the currently strongest black-box attack method) [18].

*2.2. Adversarial Defense Methods.* Multiple defense mechanisms have been proposed to protect deep learning models from the threat of adversarial examples [20–29]. Among these, adversarial training is the most effective way to improve model robustness [6, 30]. In this process, adversarial examples are generated and added to the training set to participate in the model training procedure. Since normal adversarial training remains vulnerable to adversarial examples, Tramèr et al. proposed ensemble adversarial training to improve robustness further [31]. In this process, adversarial data generated by multiple models are included in the training set for a single model, thereby producing a more robust classifier.

## 3. Methodology

This section provides a detailed introduction to our proposed methodology. Let $x$ denote an input image, and $y$ denote the corresponding ground-truth label. The term $\theta$ represents network parameters, and $J(\theta, x, y)$ describes a loss function, typically the cross-entropy loss. The primary objective is to generate an adversarial example $x^*$ to fool the model by maximizing $J(\theta, x, y)$, such that the prediction label $y^{\text{pre}} \neq y$. In this paper, we used an $L_\infty$ norm bound to limit adversarial perturbations, such that $\|x^* - x\|_\infty \leq \varepsilon$, where $\varepsilon$ refers to the size of the perturbation. The generation of adversarial examples can, therefore, be converted into the following optimization problem:

$$\begin{aligned} &\arg\max_{x^*} J(\theta, x^*, y), \\ &\text{s.t. } \|x^* - x\|_\infty \le \varepsilon. \end{aligned} \tag{1}$$

*3.1. Attack Methods Based on Gradient.* In this section, several techniques that are used to solve the above optimization problem are introduced briefly:

(i) Fast gradient sign method (FGSM) [6]: As one of the simplest techniques, it seeks adversarial examples in the direction of the gradient of the loss function with respect to the input image $x$. The method can be expressed as

$$x^* = x + \varepsilon \cdot \operatorname{sign}(\nabla_x J(\theta, x, y)). \tag{2}$$

(ii) Iterative fast gradient sign method (I-FGSM) [7]: This algorithm is an iterative version of FGSM. The approach involves dividing the FGSM gradient operation into multiple steps that can be expressed as follows:

$$\begin{aligned} x_0^* &= x, \\ x_{i+1}^* &= x_i^* + \alpha \cdot \operatorname{sign}(\nabla_x J(\theta, x_i^*, y)), \\ x_{i+1}^* &= \operatorname{Clip}(x_{i+1}^*, x - \varepsilon, x + \varepsilon). \end{aligned} \tag{3}$$

where $\alpha$ denotes the step size of each iteration and $\alpha = (\varepsilon/T)$, in which $T$ denotes the number of iterations. The Clip function was included to limit the adversarial example $x^*$ within the $\varepsilon$ neighborhood of the original image $x$ and satisfy the $L_\infty$ norm constraint. I-FGSM is more effective than FGSM in white-box environments but less effective in black-box environments. In other words, I-FGSM exhibits poor transferability. This iterative attack method is also known as projected gradient descent (PGD) if the algorithm is added by a random initialization on $x$ [20].

(iii) Momentum iterative fast gradient sign method (MI-FGSM) [8]: In this method, a momentum term is applied to the iterative process to escape from poor local maxima. This produces adversarial examples with more transferability, which can be expressed as [8]

$$\begin{aligned} x_0^* &= x, \\ g_0 &= 0, \\ g_{i+1} &= \mu \cdot g_i + \frac{\nabla_x J(\theta, x_i^*, y)}{\|\nabla_x J(\theta, x_i^*, y)\|_1}, \\ x_{i+1}^* &= x_i^* + \alpha \cdot \operatorname{sign}(g_{i+1}), \\ x_{i+1}^* &= \operatorname{Clip}(x_{i+1}^*, x - \varepsilon, x + \varepsilon), \end{aligned} \tag{4}$$

where $\mu$ denotes the decay factor of the momentum term, and $g_i$ denotes the weighted accumulation of gradients in the first $i$ rounds of iterations.

*3.2. Adam Iterative Fast Gradient Method.* The generation of adversarial examples is similar to the training of neural networks; both processes can be viewed as an optimization problem. Specifically, the adversarial example can be viewed as the training parameter, while the white-box model can be viewed as the training set, and the black-box model can be viewed as a testing set. From this point of view, the transferability of the adversarial examples is similar to the generalization of the models. Therefore, we can apply the methods used to improve the generalization of the models to the generation of adversarial examples. Many methods are proposed to improve the generalization of neural network models, which can be divided into two categories: better optimization and data augmentation. Correspondingly, these two kinds of methods can be used in the adversarial attacks, and there have been attempts to do so, for example, MI-FGSM and DIM [8, 9]. Based on the analysis above, we aim to improve the transferability of adversarial examples with Adam optimizer since it performs well in the training of neural networks.

Adam [17] is an optimization algorithm combining momentum [32] and RMSProp [33]. In the iteration process, Adam accumulates not only the gradients of the loss function with respect to the input image but also the square of the gradients. This is done to accelerate loss function ascent in dimensions with small gradients. In addition, Adam uses a decreasing step size in each iteration to achieve better convergence.

In contrast, both I-FGSM and MI-FGSM adopt a constant step size. In the later stages of the algorithm, oscillations occur near the local maximum, which do not converge well. To solve this problem, we have improved the Adam algorithm by normalizing the step size sequence (defined in Algorithm 1) and have applied it to the iterative gradient method. The proposed Adam iterative fast gradient method (AI-FGM) is summarized in Algorithm 1.

Specifically, the gradient $\nabla_x J(\theta, x_t^*, y)$ in each iteration is normalized by its own $L_1$ distance, defined in Algorithm 1, because the scale of these gradients differs widely in each iteration [8]. Similar to MI-FGSM, $m_t$ accumulates gradients of the first $t$ iterations with a decay factor $\beta_1$, defined in Algorithm 1. The result can be considered the first momentum. The term $v_t$ represents the second momentum, which accumulates the squares of gradients for the first $t$ iterations with a decay factor $\beta_2$, defined in Algorithm 1. It is noteworthy that $g_t^2$ denotes an elementwise square $g_t \odot g_t$, where $\odot$ represents the Hadamard product [32]. The terms $\beta_1$ and $\beta_2$ are typically defined in the range $(0, 1)$. The update direction of input $x$ is defined in Algorithm 1, where the stability coefficient $\delta$ is set to avoid a zero in the denominator. The $s_t$ term is advantageous as it prompts $x$ to escape from local maxima and accelerates the updating of $x$ in dimensions with small gradients.

Learning rate decay is often used in the training of neural networks. In this study, the step size used in the Adam optimizer is continually reduced to help improve the convergence of the algorithm. If $\beta_1$ and $\beta_2$ are set appropriately, the value of $\sqrt{1-\beta_2^t}/(1-\beta_1^t)$ will decrease with the increase of $t$. Thus, a decreasing sequence can be generated when $t$ ranges from 1 to $T$. The sequence can then be normalized to obtain the weight of step sizes in each iteration, relative to the total step size $\alpha$. This can be used to acquire a set of exponential decay step sizes controlled by $\beta_1$ and $\beta_2$, as defined in Algorithm 1.

Existing techniques typically use the sign function to satisfy the $L_\infty$ norm limitation. However, if the sign function was applied in Algorithm 1, the update direction in our method would be equivalent to that of MI-FGSM. Hence, we constrain adversarial examples within the $L_\infty$ norm bound by the Clip function and apply the step size and update direction within the corresponding $L_2$ norm bound, defined in Algorithm 1. The relationship between the $L_\infty$ norm bound ($\varepsilon$) and the $L_2$ norm bound ($\alpha$) is defined in Algorithm 1, where $N$ represents the dimension of the input image $x$.

*3.3. Attacking an Ensemble of Networks.* The proposed method was also used to attack an ensemble of networks. If an adversarial example poses a threat to multiple networks, it is far more likely to transfer to other models [11]. We followed the ensemble strategy proposed by Dong et al., in which multiple models are attacked by fusing network logits [8]. Specifically, in order to attack an ensemble of $K$ models, logits were fused as follows:

$$l(x) = \sum_{k=1}^{K} \omega_k l_k(x), \tag{5}$$

where $l_k(x)$ refers to the logit output of the $k$-th model, $\omega_k$ refers to the ensemble weight ($\omega_k \geq 0$), and $\sum_{k=1}^{K} \omega_k = 1$.

## 4. Experiment

Extensive validation experiments were conducted to demonstrate the effectiveness of the proposed methodology. Experimental settings are provided in Section 4.1. The results of attacking a single network and some parameters that influence the results are discussed in Section 4.2. Thus, the results of attacking an ensemble of models are shown in Section 4.3. Finally, Section 4.4 exhibits the results of attacks by the combination of AI-FGM and existing methods (source code is available at https://github.com/YinHeng121/Adam_attack).

### *4.1. Experimental Setup*

(i) Data set: if the original images cannot be classified correctly by the network, it is meaningless to generate adversarial examples based on these data. Therefore, we selected 1,000 images belonging to the 1,000 categories from the ImageNet validation set, all of which could be classified correctly by the tested networks. All images were scaled to $299 \times 299 \times 3$. As such, $N = 299 \times 299 \times 3$ in Section 3.2.

(ii) Networks: seven different models were studied; four of which were normally trained networks (i.e., Inception-v3 (Inc-v3) [34], Inception-v4 (Inc-v4) [35], Inception-Resnet-v2 (IncRes-v2) [35], and Resnet-v2-101 (Res-101) [36]). The other three models were adversarially trained networks (i.e., ens3-adv-Inception-v3 (Inc-v3$_{ens3}$), ens4-adv-Inception-v3 (Inc-v3$_{ens4}$), and ens-adv-Inception-ResNet-v2 (IncRes-v2$_{ens}$) [31]).

(iii) Other details: the proposed method was compared with the other methods discussed in Section 3.1. All the experiments described in this paper were based on the $L_\infty$ norm. Thus, the momentum decay factor $\mu$ in Algorithm 1 was 1.0, and the stability coefficient $\delta$ in Algorithm 1 was $10^{-8}$, as suggested in [17].

*4.2. Attacking a Single Network.* We first performed adversarial attacks on a single network with FGSM, I-FGSM, MI-FGSM, and AI-FGM. Moreover, the adversarial examples were generated on four normally trained networks and tested on all seven networks. The results are shown in Table 1, where the success rates are misclassification rates for the corresponding models, with adversarial examples used as input. The decay factors $\beta_1$ and $\beta_2$ in AI-FGM were set to 0.99 and 0.999, respectively. The maximum perturbation $\varepsilon$ was 16. The number of iterations $T$ for I-FGSM, PGD, MI-FGSM, and AI-FGM was 10. These methods are hereafter referred to as "iterative methods" without ambiguity. The effects of these parameter choices are discussed further in this section.

As shown in Table 1, all four iterative methods attacked a white-box model with a near 100% success rate. AI-FGM performed better than the other three methods on all black-box models. For example, for adversarial examples generated on Inc-v3, AI-FGM had a success rate of 60.7% on Inc-v4, while MI-FGSM, PGD, I-FGSM, and FGSM reached 54.3%, 18.9%, 27.5%, and 32.1%, demonstrating the effectiveness of the proposed method. An original image and the corresponding adversarial example generated for Inc-v3 by AI-FGM are shown in Figure 2.

*4.2.1. Decay Factors.* The terms $\beta_1$ and $\beta_2$ control not only the decay amplitude of the step sizes but also the accumulation intensity of gradients for $m_t$ and $v_t$. As such, they have a direct impact on attack success rates. We applied a grid-search method to identify an optimal set of $\beta_1$ and $\beta_2$ values. In the experiments, the maximum perturbation $\varepsilon$ was set to 16, and the number of iterations $T$ was set to 10. The values of $\beta_1$ and $\beta_2$ ranged from 0 to 1. Notably, we not only chose the values of $\beta_1$ and $\beta_2$ uniformly from 0.1 to 0.9 but also selected values close to 0 and 1, as shown in Figure 3. This was done to provide a more comprehensive study on the effects of decay factors, as we propose that the

Input: A convolutional neural network $f$ and the corresponding cross-entropy loss function $J(\theta, x, y)$; an original image $x$ and the corresponding ground-truth label $y$; the number of iterations $T$; the iteration time step $t$; the dimension of the input image $N$; the size of the perturbation $\varepsilon$; Adam decay factors $\beta_1$ and $\beta_2$; and a denominator stability factor $\delta$.
Output: An adversarial example $x^*$, s.t. $\|x - x^*\|_\infty \leq \varepsilon$.

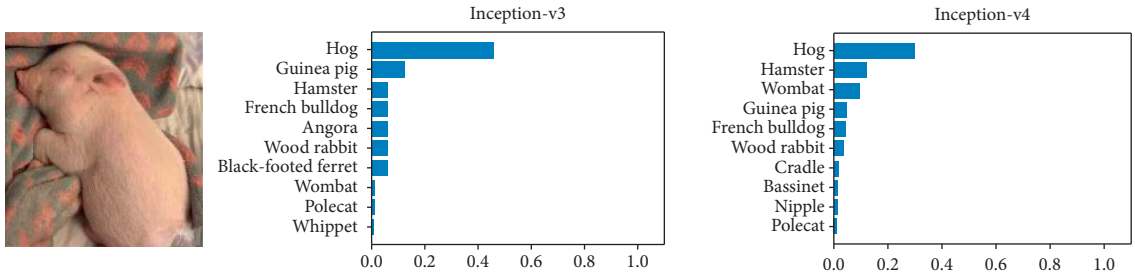
$m_0 = 0$, $v_0 = 0$, $x_0^* = x$, and $t = 0$;
$\alpha = \varepsilon \cdot \sqrt{N}$
while $t < T$ do:
  $g_t = (\nabla_x J(\theta, x_t^*, y)/\|\nabla_x J(\theta, x_t^*, y)\|_1)$
  $m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_t$;
  $v_{t+1} = \beta_2 \cdot v_t + (1 - \beta_2) \cdot g_t^2$;
  $s_{t+1} = (m_{t+1}/\delta + \sqrt{v_{t+1}})$;
  $\alpha_t = \alpha \cdot (\sqrt{1 - \beta_2^{t+1}}/1 - \beta_1^{t+1})/\sum_{i=0}^{T-1} (\sqrt{1 - \beta_2^{i+1}}/1 - \beta_1^{i+1})$;
  $x_{t+1}^* = x_t^* + \alpha_t \cdot (s_{t+1}/\|s_{t+1}\|_2)$;
  $x_{t+1}^* = \mathrm{Clip}(x_{t+1}^*, x - \varepsilon, x + \varepsilon)$;
  $t = t + 1$;
end while
return $x^* = x_T^*$.

Algorithm 1: Adam iterative fast gradient method.

Table 1: Attack success rates (%) for all seven networks included in the study.

| | Attack | Inc-v3 | Inc-v4 | Incres-v2 | Res-101 | Inc-v3$_{ens3}$ | Inc-v3$_{ens4}$ | Incres-v2$_{ens}$ |
|---|---|---|---|---|---|---|---|---|
| Inc-v3 | FGSM | 72.2 | 32.1 | 31.7 | 32.3 | 10.3 | 10.6 | 4.2 |
| | I-FGSM | **100.0** | 27.5 | 23.0 | 20.9 | 6.2 | 4.7 | 1.8 |
| | PGD | 99.8 | 18.9 | 14.7 | 14.4 | 6.1 | 6.2 | 3.2 |
| | MI-FGSM | **100.0** | 54.3 | 50.6 | 44.0 | 13.9 | 13.3 | 6.5 |
| | AI-FGM | **100.0** | **60.7** | **55.8** | **50.2** | **17.0** | **16.8** | **8.5** |
| Inc-v4 | FGSM | 39.4 | 64.9 | 29.5 | 33.0 | 12.2 | 11.1 | 5.0 |
| | I-FGSM | 43.8 | **100.0** | 27.4 | 23.4 | 6.1 | 6.3 | 2.4 |
| | PGD | 32.0 | 99.8 | 17.7 | 16.6 | 6.4 | 5.8 | 3.1 |
| | MI-FGSM | 69.9 | **100.0** | 57.9 | 54.1 | 19.6 | 17.7 | 8.7 |
| | AI-FGM | **72.9** | **100.0** | **60.1** | **57.1** | **21.7** | **19.5** | **10.3** |
| Incres-v2 | FGSM | 37.6 | 31.8 | 57.9 | 31.4 | 13.7 | 12.0 | 6.8 |
| | I-FGSM | 46.1 | 35.0 | 99.4 | 30.4 | 7.3 | 6.7 | 4.4 |
| | PGD | 30.1 | 23.0 | 97.3 | 18.3 | 6.1 | 5.7 | 2.7 |
| | MI-FGSM | 73.5 | 69.3 | **99.5** | 60.0 | 27.0 | 23.0 | 16.7 |
| | AI-FGM | **74.6** | **71.1** | **99.5** | **61.8** | **31.3** | **25.7** | **20.5** |
| Res-101 | FGSM | 38.3 | 33.0 | 30.2 | 79.3 | 14.6 | 13.3 | 6.4 |
| | I-FGSM | 35.1 | 28.3 | 25.1 | 99.5 | 8.4 | 6.7 | 3.7 |
| | PGD | 30.9 | 22.4 | 20.9 | **99.6** | 7.3 | 7.2 | 3.4 |
| | MI-FGSM | 60.0 | 55.3 | 50.6 | 99.5 | 22.9 | 19.8 | 11.3 |
| | AI-FGM | **64.0** | **57.7** | **54.0** | **99.5** | **27.2** | **24.1** | **15.4** |

The diagonal blocks indicate white-box attacks, while the off-diagonal blocks indicate black-box attacks.




(a)

Figure 2: Continued.

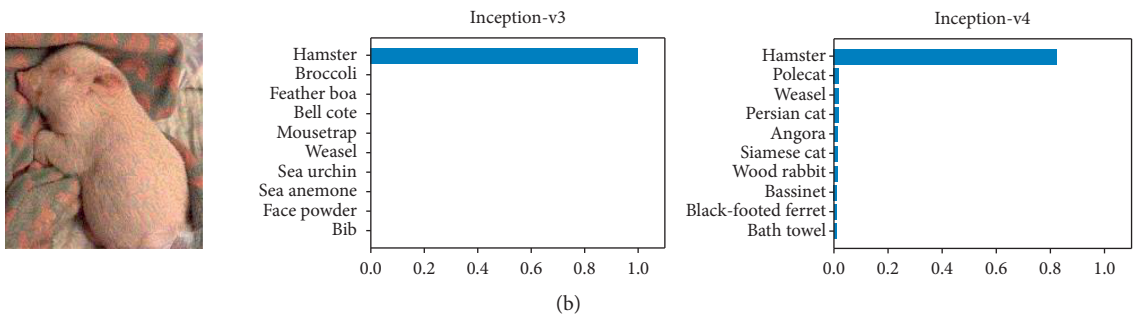



Figure 2: Classification of a normal image and corresponding adversarial example by Inc-v3 and Inc-v4. The first row shows the top 10 confidence distributions for a clean image, for which both models provided a correct prediction. The second row shows the top 10 confidence distributions for the adversarial example generated for Inc-v3 by AI-FGM. It is evident that the adversarial example successfully attacked Inc-v3 (white-box) and Inc-v4 (black-box) with high confidence. (a) Clean. (b) Adversarial.

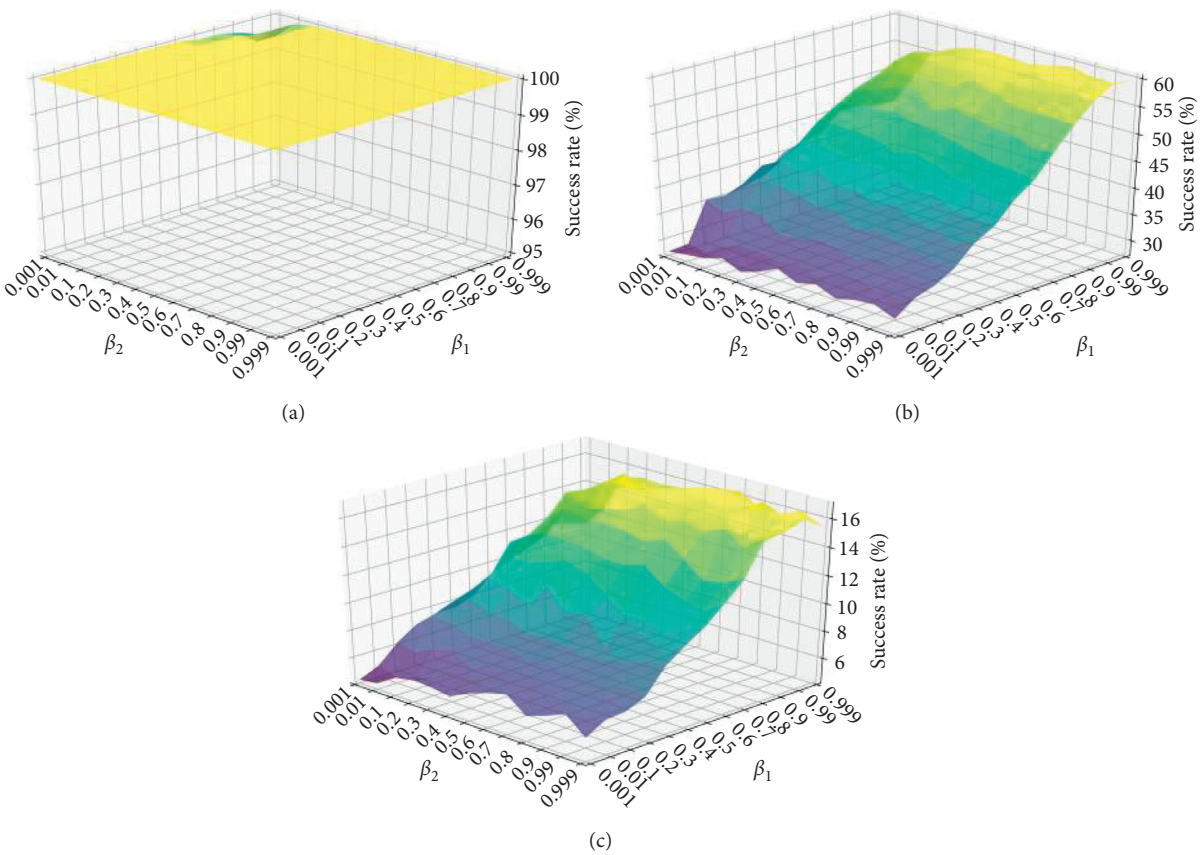



Figure 3: Attack success rates (%) of adversarial examples generated for Inc-v3 with AI-FGM, applied to Inc-v3 (white-box), Inc-v4 (black-box and normally trained), and Inc-v3ens4 (black-box and adversarially trained) with $\beta_1$ and $\beta_2$ in the range of $(0, 1)$. (a) Inc-v3. (b) Inc-v4. (c) Inc-v3$_{ens4}$.

relationship between the success rate and the decay factors may be nonlinear. Adversarial examples were then generated on Inc-v3 with AI-FGM and used to perform attacks on Inc-v3, Inc-v4, and Inc-v3$_{ens4}$. It is evident from the figure that regardless of the values of $\beta_1$ and $\beta_2$, attack success rates were always near 100% in white-box environments. Beyond that, attack success rates were more sensitive to $\beta_1$ for black-box models, regardless of whether networks were normally or adversarially trained. Success rates were maximized when both $\beta_1$ and $\beta_2$ were close to 1.

*4.2.2. The Number of Iterations.* The effect of iteration quantities on success rates was studied by performing attacks using iterative methods. The maximum perturbation $\varepsilon$ was set to 16, and the decay factors $\beta_1$ and $\beta_2$ were set to 0.99 and

0.999, respectively. The number of iterations $T$ ranged from 1 to 20. Adversarial examples generated on Inc-v3 with I-FGSM, MI-FGSM, and AI-FGM were then used to attack Inc-v3 and Inc-v4, as shown in Figure 4.

As shown in the figure, the success rates of black-box attacks for several iterative methods decreased with the increase of the iteration quantities. However, AI-FGM always outperformed I-FGSM and MI-FGSM for a given value of $T$.

*4.2.3. Perturbation Size.* The effect of adversarial perturbation size on attack success rates was studied by setting the number of iterations $T$ to 10, with decay factors of 0.99 and 0.999. The size of perturbations $\varepsilon$ ranged from 1 to 30. Adversarial examples generated on Inc-v3 with I-FGSM, MI-FGSM, and AI-FGM were used to attack Inc-v3 and Inc-v4, as shown in Figure 5.

It is evident from the figure that attack success rates can reach 100% in white-box environments. Success rates for black-box models increased steadily with increasing values of $\varepsilon$. AI-FGM always outperformed I-FGSM and MI-FGSM for a given value of $\varepsilon$. In other words, AI-FGM can achieve comparable black-box attack success rates with smaller perturbations.

*4.3. Attacking an Ensemble of Networks.* The experimental results presented above suggest that AI-FGM can increase the transferability of adversarial examples. In addition, the success rate of black-box attacks can be further improved by attacking an ensemble of networks. As discussed in Section 3.3, multiple models were attacked by fusing network logits. In the experiment, all seven networks described in Section 4.1 were used, and we generated adversarial examples on the ensemble of Inc-v3, Inc-v4, IncRes-v2, and Res-101, with FGSM, I-FGSM, MI-FGSM, NI-FGSM, and AI-FGM. Attacks were then performed on the other three defense models. Decay factors were set to 0.99 and 0.999; the number of iterations was 10; the maximum perturbation was 16; and each network had an equal ensemble weight of $\omega_k = (1/4)$. The corresponding results are shown in Table 2, and it is evident that AI-FGM was more effective than the other four methods on adversarially trained models.

*4.4. Combination with Advanced Methods.* In this subsection, we combined our AI-FGM with SI-NI-TI, SI-NI-DI, and SI-NI-TI-DI [18] and compared the black-box attack success rates of our extensions with the original methods under a single-model setting. The adversarial examples were generated on Inc-v3, with the number of iterations set to 10 and the maximum perturbation to 16. It is noteworthy that our combination with other methods is more like an improvement. For example, the combination of AI-FGM and SI-NI-TI was done by replacing the Nesterov optimizer in SI-NI-TI with Adam, thus resulting in SI-AI-TI. As shown in Table 3, our method SI-AI-TI-DI achieved an average attack success rate of 65.1%, surpassing the state-of-the-art attack by 10.7%.

Furthermore, we generated adversarial examples on the ensemble models using our SI-AI-TI-DI. As shown in Table 4, we achieved an average attack success rate of 95.0% on adversarially trained models under the black-box setting, which raised a new security issue for the robust deep neural networks.

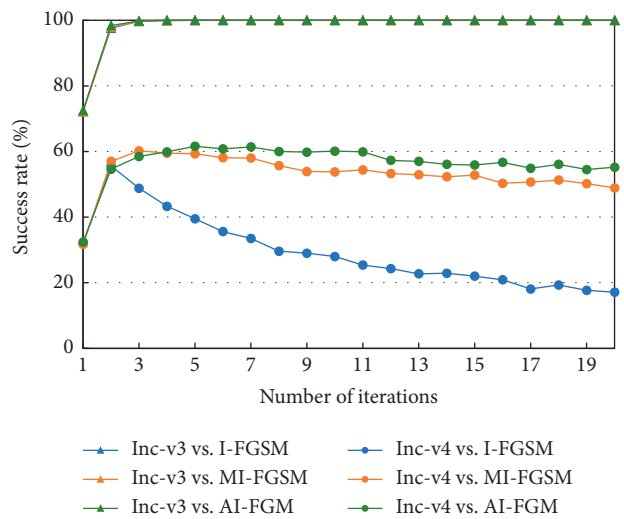



Figure 4: Attack success rates (%) of adversarial examples generated for Inc-v3 with I-FGSM, MI-FGSM, and AI-FGM, applied to Inc-v3 (white-box) and Inc-v4 (black-box) with $T$ ranging from 1 to 20. It is noteworthy that the curves of Inc-v3 vs. I-FGSM, Inc-v3 vs. MI-FGSM, and Inc-v3 vs. AI-FGM overlap.

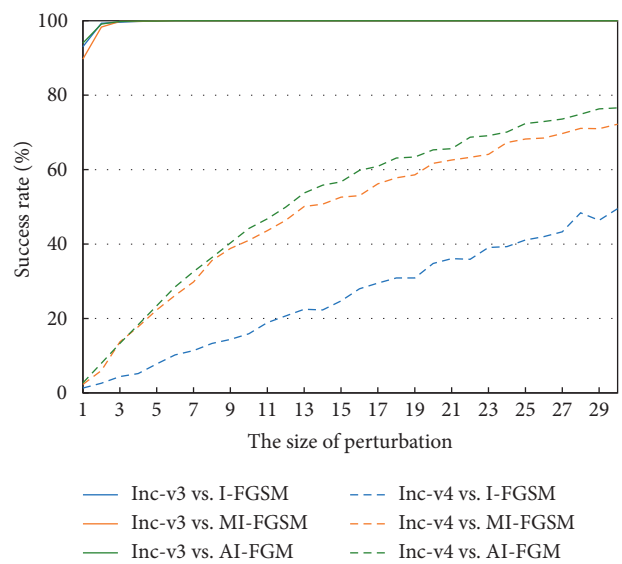



Figure 5: Attack success rates (%) of adversarial examples generated for Inc-v3 with I-FGSM, MI-FGSM, and AI-FGM, applied to Inc-v3 (white-box) and Inc-v4 (black-box) with $\varepsilon$ ranging from 1 to 30. It is noteworthy that the curves of Inc-v3 vs. I-FGSM, Inc-v3 vs. MI-FGSM, and Inc-v3 vs. AI-FGM overlaps.

## 5. Discussion

It is commonly acknowledged that the training of neural network models is similar to the generation of adversarial

Table 2: Attack success rates (%) on adversarially trained networks under the ensemble-model setting.

| Attack | Inc-v3$_{ens3}$ | Inc-v3$_{ens4}$ | Incres-v2$_{ens}$ | Average |
|---|---|---|---|---|
| FGSM | 18.6 | 17.3 | 9.5 | 15.1 |
| I-FGSM | 20.1 | 16.9 | 11.4 | 16.1 |
| MI-FGSM | 49.4 | 43.2 | 25.9 | 39.5 |
| NI-FGSM | 46.7 | 41.6 | 23.0 | 37.1 |
| AI-FGM (ours) | 56.2 | 50.6 | 33.0 | 46.6 |

The adversarial examples are generated on the ensemble of Inc-v3, Inc-v4, IncRes-v2, and Res-101.

Table 3: Comparison of combined methods and original methods on attack success rates (%) against adversarially trained networks under the single-model setting.

| Attack | Inc-v3$_{ens3}$ | Inc-v3$_{ens4}$ | Incres-v2$_{ens}$ | Average |
|---|---|---|---|---|
| SI-NI-TI | 54.7 | 52.3 | 34.4 | 47.1 |
| SI-AI-TI (ours) | 56.3 | 54.4 | 39.4 | 50.0 |
| SI-NI-DI | 39.5 | 37.2 | 19.3 | 32.0 |
| SI-AI-DI (ours) | 50.8 | 51.5 | 27.7 | 43.3 |
| SI-NI-TI-DI | 62.3 | 59.3 | 41.7 | 54.4 |
| SI-AI-TI-DI (ours) | 72.6 | 69.8 | 53.0 | 65.1 |

The adversarial examples are generated on the ensemble of Inc-v3, Inc-v4, Incres-v2, and Res-101.

Table 4: Comparison of SI-AI-TI-DI and SI-NI-TI-DI under the ensemble-model setting.

| Attack | Inc-v3$_{ens3}$ | Inc-v3$_{ens4}$ | Incres-v2$_{ens}$ | Average |
|---|---|---|---|---|
| SI-NI-TI-DI | 95.4 | 93.7 | 89.5 | 92.9 |
| SI-AI-TI-DI (ours) | 96.2 | 96.0 | 92.7 | 95.0 |

The adversarial examples are generated on the ensemble of Inc-v3, Inc-v4, Incres-v2, and Res-101.

examples, especially for gradient-based generation methods. Hence, techniques used in the training of neural networks, to improve model generalizability, can also be adopted to improve the transferability of adversarial examples. Since the Adam optimizer is often used in the training of neural networks, to improve convergence and achieve better performance on test sets, a second momentum term and decay step size in Adam were included in the AI-FGM algorithm. This was done to improve the transferability of adversarial examples. Furthermore, we suggest that other techniques (such as data augmentation) could be used to improve the performance of adversarial examples further in black-box environments.

## 6. Conclusions

In this study, we proposed the Adam iterative fast gradient method to improve the transferability of adversarial examples. Specifically, the Adam algorithm was modified to increase its suitability for the generation of adversarial examples. The proposed method improved both the iteration update direction and step size. The effectiveness of the proposed method was verified by an extensive series of experiments with ImageNet. Compared with previous gradient-based adversarial example generation techniques, our method improved attack success rates in black-box environments. In addition, we further improved the transferability of adversarial examples by combining our approach with existing methods and attacking ensemble models, which achieved a state-of-the-art attack success rate against adversarially trained networks. We suggest the proposed method could be used as a reference for other iterative gradient-based methods. For example, data augmentation could be combined with AI-FGM to achieve better attack performance.

## Data Availability

The public data (data set and models) can be downloaded from https://github.com/tensorflow/cleverhans/tree/master/examples/nips17_adverarial_competition/dataset, https://github.com/tensorflow/models/tree/master/research/slim, and https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models. The source code of the proposed method in this paper is available at https://github.com/YinHeng121/Adam_attack.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Authors' Contributions

Heng Yin and Hengwei Zhang contributed equally to this work.

## Acknowledgments

This work was supported by the National Key Research and Development Program of China, under grant no. 2017YFB0801900.

## References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, http://arxiv.org/abs/1409.1556.

[3] C. Szegedy, "Going deeper with convolutions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, Boston, MA, USA, June 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[5] C. Szegedy, "Intriguing properties of neural networks," in *Proceedings of the 2013 International Conference on Learning Representations*, Banff, Canada, April 2013.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, http://arxiv.org/abs/1412.6572.

[7] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, http://arxiv.org/abs/1607.02533.

[8] Y. Dong, "Boosting adversarial attacks with momentum," in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193, Salt Lake City, UT, USA, June 2018.

[9] C. Xie, "Improving transferability of adversarial examples with input diversity," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2725–2734, Long Beach, CA, USA, June 2019.

[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy*, pp. 39–57, San Jose, CA, USA, May 2017.

[11] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," 2016, http://arxiv.org/abs/1611.02770.

[12] A. I. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," 2018, http://arxiv.org/abs/1804.08598.

[13] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *Journal of Machine Learning and Research*, vol. 15, pp. 949–980, 2014.

[14] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C. J. Hsieh, and M. B. Srivastava, "GenAttack: practical black-box attacks with gradient-free optimization," in *Proceedings of the 2019 Genetic Evolution Computer Conference*, pp. 1111–1119, New York, NY, USA, July 2019.

[15] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: reliable attacks against black-box machine learning models," 2017, http://arxiv.org/abs/1712.04248.

[16] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 Asia Conference on Computer and Communications Security,*, pp. 506–519, Abu Dhabi, UAE, April 2017.

[17] D. P. Kingma and J. Ba, "A method for stochastic optimization," 2014, http://arxiv.org/abs/1412.6980.

[18] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, "Nesterov accelerated gradient and scale invariance for adversarial attacks," in *Proceedings of the ICLR 2020*, Addis Ababa, Ethiopia, April 2020.

[19] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4307–4316, Long Beach, CA, USA, June 2019.

[20] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2018, http://arxiv.org/abs/1706.06083.

[21] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy*, pp. 582–597, San Jose, CA, USA, May 2016.

[22] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: one hot way to resist adversarial examples," in *Proceedings of the 2018 International Conference Learning Representations*, Vancouver, BC, Canada, May 2018.

[23] C. Guo, M. Rana, M. Cissé, and L. Maaten, "Countering adversarial images using input transforma-tions," 2017, http://arxiv.org/abs/1711.00117.

[24] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: protecting classifiers against adversarial attacks using generative models," 2018, http://arxiv.org/abs/1805.06605.

[25] A. Levine and S. Feizi, "Wasserstein smoothing: Certified robustness against wasserstein adversarial attacks," in *Proceedings of the 2020 Internaltional Conference Artificial Intellgent Statistic Palermo*, Sicily, Italy, August 2020.

[26] C. Mao, "Multitask learning strengthens adversarial robustness," 2020, http://arxiv.org/abs/2007.07236.

[27] F. Wu, W. Yang, L. Xiao, and J. Zhu, "Adaptive wiener filter and natural noise to eliminate adversarial perturbation," *Electronics*, vol. 9, no. 10, 1634 pages, 2020.

[28] J. M. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," 2019, http://arxiv.org/abs/1902.02918.

[29] Y. Shi, C. Fan, L. Zou, C. Sun, and Y. Liu, "Unsupervised adversarial defense through tandem deep image priors," *Electronics*, vol. 9, no. 11, 1957 pages, 2020.

[30] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, http://arxiv.org/abs/1611.01236.

[31] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel, "Ensemble adversarial training: attacks and defenses," 2017, http://arxiv.org/abs/1705.07204.

[32] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 2013 International Conference Machnisim Learning*, pp. 1139–1147, Atlanta, GA, USA, June 2013.

[33] T. Tieleman and G. Hinton, "Lecture 6.5-RMSProp, coursera: neural networks for machine learning," 2012.

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, Las Vegas, NV, USA, June 2016.

[35] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proceedings of the 2017 AAAI Conference Artificial Intellgence*, San Francisco, California, USA, February 2017.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proceedings of the 2016 Computer Vision*, pp. 630–645, Cham, Switzerland, September 2016.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 6664578, 10 pages
https://doi.org/10.1155/2021/6664578

*Research Article*

# Spoofing Speaker Verification System by Adversarial Examples Leveraging the Generalized Speaker Difference

**Hongwei Luo**,[1] **Yijie Shen**,[2] **Feng Lin**,[2] **and Guoai Xu**[1]

[1]*National Engineering Laboratory of Mobile Network Security, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*Institute of Cyberspace Research, College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*

Correspondence should be addressed to Feng Lin; flin@zju.edu.cn

Received 1 December 2020; Revised 25 December 2020; Accepted 29 January 2021; Published 9 February 2021

Academic Editor: Ting Chen



Speaker verification system has gained great popularity in recent years, especially with the development of deep neural networks and Internet of Things. However, the security of speaker verification system based on deep neural networks has not been well investigated. In this paper, we propose an attack to spoof the state-of-the-art speaker verification system based on generalized end-to-end (GE2E) loss function for misclassifying illegal users into the authentic user. Specifically, we design a novel loss function to deploy a generator for generating effective adversarial examples with slight perturbation and then spoof the system with these adversarial examples to achieve our goals. The success rate of our attack can reach 82% when cosine similarity is adopted to deploy the deep-learning-based speaker verification system. Beyond that, our experiments also reported the signal-to-noise ratio at 76 dB, which proves that our attack has higher imperceptibility than previous works. In summary, the results show that our attack not only can spoof the state-of-the-art neural-network-based speaker verification system but also more importantly has the ability to hide from human hearing or machine discrimination.

## 1. Introduction

In recent years, the verification system has been employed in many scenarios including entrance guard, online payment, and smart home management. Speaker verification system that offers a convenient and reliable verification is a popular biometrics system. It verifies the person by an utterance that contains the unique biometrics feature called voiceprint. Voiceprint has the advantages of noncontact, high privacy, and low cost compared with other biometrics features used in verification system (e.g., fingerprint [1], face ID [2], and iris features [3]). Therefore, speaker verification has become a promising biometrics technique and received high social acceptance, especially in the area of smart Internet of Things (IoT) such as voice assistant.

One concern of applying speaker verification systems is whether they are secure enough. To explore this latent risk, we first reviewed the speaker verification system to understand how the speaker verification system works. We found that current speaker verification systems can be divided into two types: one is text-dependent speaker verification (TD-SV) and the other is text-independent speaker verification (TI-SV). They have different requirements for inputs. TD-SV requires users to say the same utterance with the one used to enroll, but users can say any utterance for verification in TI-SV. Obviously, TI-SV offers a more convenient speaker verification system than TD-SV. However, we raised a question about the security of the speaker verification system based on TI-SV: Can the speaker verification system based on TI-SV be spoofed by the adversary?

Firstly, we need to find the state-of-the-art speaker verification system based on TI-SV. We found that deep-neural-networks-based speaker verification shows a better performance than conventional solutions through the review. Because it shows a more promising prospect, we set it as our target speaker verification system to explore the latent risk. With the further survey, we raised another question:

Can the speaker verification system based on deep neural networks be spoofed by adversarial examples? To find out whether the possibility of spoofing exists, we firstly perform a comprehensive review of voiceprint identification technology using deep neural networks [4–6]. Among these works, a state-of-the-art embedding vector model with generalized end-to-end (GE2E) loss function [5] proposed by Google performs best and has been applied in many domains (e.g., education and speaker verification). Our work tries to explore the vulnerability in TI-SV based on the GE2E loss function due to the above knowledge. We rebuild this speaker verification system, therefore, with the TIMIT dataset and try to spoof it by adversarial examples. To achieve the spoofing attack, we firstly give two requirements.

*1.1. Imperceptible.* The adversarial examples need to be similar to the original utterance. In other words, the injected perturbation to the original utterance should be slight enough. Otherwise, the spoofing attack will be discovered with ease by humans or machines, which will cause the spoofing attack to fail.

*1.2. Purposeful.* The victim should be verified to the special target set by the adversary. Then the spoofing attack is more powerful and destructive.

We consider that the speaker verification system verifies the user by detail waveform rather than the macroscopical waveform. The trait gives us the possibility to spoof the speaker verification system with a slight perturbation. Based on this possibility, we found several technical challenges that still need to be addressed to realize the spoofing attack: (i) How to generate the perturbation for the victim? (ii) How to limit the perturbation slightly enough? (iii) How to evaluate the perturbation, which can measure the influence in utterance? We structure a novel adversarial examples generator for producing effective slight perturbation to spoof the state-of-the-art speaker verification system, which is shown in Figure 1. The adversary utilizes the generator to generate a tiny perturbation to inject into an utterance of the unregistered user; the synthetic utterance will be verified as the targeted registered legal user. In addition, we designed a novel loss function to achieve the imperceptibility that tries to find the slightest perturbation on the premise of the spoofing attack. We proposed two different methods that can evaluate perturbation in maximum noise ratio (MNR) and signal-to-noise ratio (SNR). Finally, we evaluated our spoofing attack on the TIMIT dataset, which contains 630 speakers and 6300 sentences. Based on the dataset, we rebuild the state-of-the-art speaker verification system with GE2E loss function and two identification models, linear discriminate analysis (LDA) and cosine similarity threshold, respectively. In our experiments, the spoofing attack achieved up a high success rate of 82% and a slight distortion of −77 dB in MNR (usually negative) and 76 dB in SNR (usually positive).

To summarize, contributions in our work can be listed as follows:

(1) We proposed a novel multifactor-based attack to spoof the state-of-the-art TI-SV system based on deep learning. Our spoofing attack transforms an illegal utterance's verification result into a legal target with a slight perturbation. Meanwhile, we do not have any utterance from the target. To the best of our knowledge, this is the first exploratory work in spoofing the state-of-the-art TI-SV system based on deep learning.

(2) We consider imperceptibility as a key metric in the spoofing attack. Hence, our spoofing attack improves imperceptibility by a novel loss function. The result shows that the imperceptibility of our spoofing attack is much better than previous works.

(3) We evaluated our spoofing attack on the state-of-the-art TI-SV system based on deep learning with the TIMIT dataset including 630 speakers. The result shows that our spoofing attack achieves up a high success rate and high imperceptibility.

## 2. Related Work

This section will introduce previous works about spoofing attacks in speaker verification systems and adversarial examples in the audio domain.

*2.1. Attack on Speaker Verification Services.* Many conventional methods were used to realize speaker verification systems before deep learning, including two models that work best: *i*-vector [7] and Gaussian mixture model-universal background model (GMM-UBM) [8]. The two models acquired effective results in realizing speaker verification systems, but there is still some vulnerability that was found by adversaries. For instance, two genetic algorithms were used to produce a new utterance, which will be identified as the target in both *i*-vector and GMM-UBM, with a target utterance from the dataset [9]. Moreover, adversaries can also use voice conversion to transform the original utterance to the target utterance, and the transformed utterance will have similar features to the target utterance in speaker verification systems [10]. The deep-learning-based verification systems also can be spoofed (e.g., the spoofing attack in *d*-vector [11]). In our work, we aimed at the state-of-the-art TI-SV and proposed a novel attack method to spoof the state-of-the-art speaker verification system.

*2.2. Audio Adversarial Examples.* Adversarial examples have been utilized in different domains as spoofing attack methods and reported effective results in recent research [12–14], including the audio domain. Speech-to-text is an important problem in the audio domain; an effective solution has been universally utilized, which is called connectionist temporal classification [15] (CTC). With the development of technology, a spoofing attack that is based on adversarial examples has been proposed, one generator will produce a perturbation to make the new utterance sounds like the original one, and Carlini and Wagner [16]

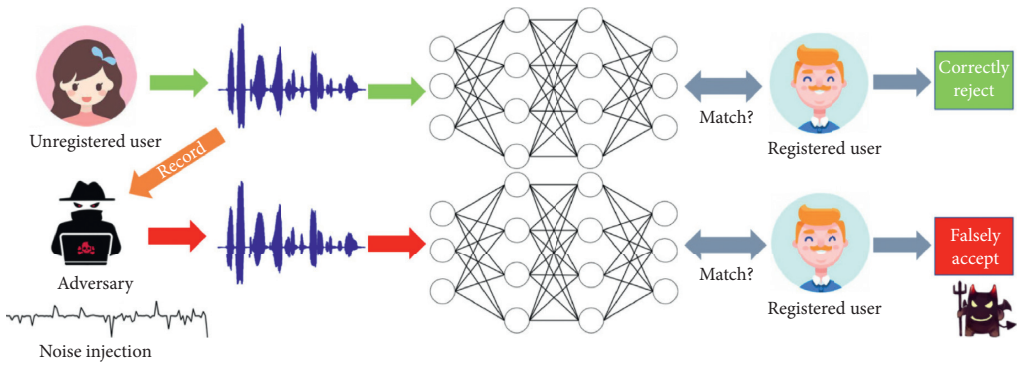



Figure 1: The adversary injects a special and slight perturbation into the original utterance, which does not change the waveform but influences the verification result.

proposed utilizing this method to spoof the speech-to-text system. The spoofing attack [17] to the speech-to-text system is reported as more imperceptible. Aiming at intelligent voice assistants, Qin et al. [18] proposed an attack to produce an incorrect command that cannot be detected by people. Our work is based on adversarial examples to attack the state-of-the-art speaker verification system. Compared with previous works, our target is the speaker verification system rather than the speech-to-text system and our attack is more imperceptible.

## 3. Background

In this section, we will elaborate on technologies and related concepts that were used in our work.

*3.1. Speaker Verification Systems.* First of all, we will briefly introduce two different forms of speaker verification systems, TD-SV and TI-SV. There are several works in TD-SV [19, 20], but TD-SV required users to say the fixed utterance in both enrollment and verification. The constraint makes TD-SV not able to be utilized in continued verification and high user experience requirement situation. Hence, TI-SV was proposed to mitigate this constraint. TI-SV focuses on finding the features from speakers independently; also several works [21, 22] were proposed by the benefit of these advantages. TI-SV is more practical and convenient than TD-SV. To this end, it is a meaningful work to explore the attack possibility on TI-SV.

Due to this point, our work focuses on exploring vulnerabilities in TI-SV, and we first reviewed the technologies of TI-SV. Through the review, we found that several works realized TI-SV based on *i*-vector [7] and GMM-UBM [8]. Further, we found that, with the rapid development of deep learning networks, it was also used in speaker verification to extract the voiceprint to representing the speaker's identity. The deep-learning-based TI-SV performs better than previous solutions based on *i*-vector and GMM-UBM. Because of the higher efficiency of the deep-learning-based TI-SV, our attack is targeting deep-learning-based TI-SV. We analyzed the main process of these solutions. For an utterance, the voiceprint extractor employs an embedding network to extract feature vectors which can represent the identity of the speaker from a processed utterance. After the process, the deep-learning-based TI-SV verifies feature vectors by different methods. We found that the loss function is a critical part that will influence the final performance directly during our experiment. Hence, we reviewed the deep-learning-based TI-SV, and we found the state-of-the-art loss function, tuple-based end-to-end [6] (TE2E) loss function, and generalized end-to-end [5] (GE2E) loss function, which have been widely used in real life and achieved great performance. Next, we will briefly illustrate GE2E and its prior work TE2E.

*3.1.1. Tuple-Based End-to-End.* TE2E outputs embedding vectors for one evaluation utterance and enrollment utterances by long short-term memory (LSTM); and the centroid of the enrollment embedding vectors is calculated. The centroid can be represented by $c_k$, where $k$ represents the $k^{th}$ speaker, and it is mean value vector of all utterances from the $k^{th}$ speaker. The evaluation embedding vector is represented by $e_j$, where $j$ represents the $j^{th}$ speaker. TE2E quantifies the distance between $c_k$ and $e_j$ by cosine similarity with the formula

$$s = w \times \cos\left(e_j, c_k\right) + b, \tag{1}$$

where $w$ and $b$ are the parameters that will be learned during training. Based on these definitions, TE2E loss is defined as follows:

$$L_T\left(e_j, c_k\right) = \delta\left(e_j, c_k\right)\sigma(s) + \left(1 - \delta\left(e_j, c_k\right)\right)(1 - \sigma(s)), \tag{2}$$

$$\sigma(x) = \frac{1}{\left(1 + e^{-x}\right)}, \tag{3}$$

where if $j$ equals $k$, then $\delta(e_j, c_k) = 1$; otherwise, $\delta(e_j, c_k) = 0$. Although TE2E can work well in both TI-SV and TD-SV, it has several disadvantages. Firstly, TE2E gets a scalar representing the similarity between embedding vector $e_j$ and single centroid $c_k$; this makes the network not able to capture features from other enrollment speakers. Therefore, they further proposed the GE2E model.

*3.1.2. Generalized End-to-End.* Compared with TE2E, GE2E builds the similarity matrix between $e_{(ji)}$ and all $c_k$ rather than a single $c_k$ with an $N \times M$ input, where $N$ represents $N$ speakers, $M$ represents $M$ utterances for each speaker, $j \in \{0, 1, \ldots, N-1\}$, $i \in \{0, 1, \ldots, M-1\}$, and $k \in \{0, 1, \ldots, N-1\}$. When calculating $c_k$, if $j == k$, then calculate $c_k$ with other $M-1$ utterances; otherwise, calculate $c_k$ with $M$ utterances. Then GE2E defined two different loss functions, the softmax loss function is defined by Equation (4), and the contrast loss function is defined by Equation (5).

$$L(e_{ji}) = -S_{ji,j} + \log \sum_{k=1}^{N} \exp(S_{ji,k}), \tag{4}$$

$$L(e_{ji}) = 1 - \sigma(S_{ji,j}) + \max_{\substack{1<k<N \\ k \neq j}} \sigma(S_{ji,k}), \tag{5}$$

where $\sigma = (1/(1+e^{-x}))$, $S$ is the similarity matrix, and $S_{ji,k}$ represents the similarity between the $i^{\text{th}}$ utterance of the $j^{\text{th}}$ speaker and the center of the $k^{\text{th}}$ speaker. $e_{ji}$ is the $i^{\text{th}}$ utterance of the $j^{\text{th}}$ speaker. The softmax loss function performs well on TI-SV and contrast loss function performs well on TD-SV. Because GE2E considers global features rather than local features, it can extract more unique features than TE2E.

Since the GE2E loss function has better performance than the TE2E loss function, our work utilized the GE2E loss function with softmax loss function to realize the TI-SV as our attack target.

*3.2. Adversarial Examples.* As a learning-based spoofing attack method, adversarial examples were proposed to inject tiny perturbation, which will lead the network to output incorrect results with high confidence. It was first proved effective in the image domain. The difference between the attacked image and the original image cannot be distinguished by the human eyes, but the attacked image will be classified into different results with the original image. In the audio domain, adversarial examples attack also exists. For instance, a generator is utilized to produce adversarial examples to spoof speech-to-text systems [16]. Here, we employed adversarial examples to generate the perturbation that cannot be perceived by ears to realize our attack.

## 4. Attack Model

In this section, we will illustrate the target and constraints of our spoofing attack.

As the final goal, the adversary wants to transform the illegal utterance to be identified into legal identity, which has been set as the target victim, hereafter A, by the adversary. To achieve this target, for any given utterance $x$ which is from anyone other than the victim, hereafter B, where $x$ belongs to B, the adversary tries to generate a slight perturbation $\delta$ and introduced $\delta$ into the audio waveform $x$ as a new audio waveform $x'$, where $x'$ equals $x + \delta$. The adversary wants $x'$ to be verified as A, which can spoof the speaker verification system in mobile phone, online systems, and so on.

We assume a black-box setting where the adversary only knows the output scores and the identified result of the speaker verification system, without the detailed structure of speaker verification system that is required in white-box setting. In addition, we assume that our adversarial examples $\delta$ can be directly introduced into the waveform without any noise (e.g., ambient noise when we play them over the air). These constraints are reasonable as they also appear in prior work [16]. We prefer to prove the possibility of this attack rather than the practical application. To make the work more confident, we also discussed several advanced attacks in Section 9 to overcome these constraints on this basis, which will improve the practical ability of our work. Note that our spoofing attack injects noise into the real utterance; thus, the antispoofing method by living detection [23, 24] is not effective aiming at this attack.

## 5. Attack System Design

In this section, we will start with the adversarial examples' requirements in our work. Next, we will design two different systems by unique loss functions and describe them.

*5.1. Adversarial Examples' Requirements.* As an effective attack, the adversarial examples in our experiments need to satisfy several requirements. In an adversarial example's generative process, we will inject perturbation into a given original utterance to generate a new utterance; we define the new utterance with the following equation:

$$x' = x + \delta, \tag{6}$$

where $\delta$ represents the perturbation that will be injected into the utterance and $x$ represents the original utterance. To make the attack effective, we need to restrict the process with the three following points: (1) $x'$ must be in an available range which let the waveform be able to be recovered as an utterance; (2) $\delta$ needs to be as slight as possible; (3) the speaker verification system will recognize $x'$ as the special target that the adversary sets before the attack. To better describe our requirements for adversarial examples, we formulate our requirements with the following formulations:

$$\begin{aligned} & x + \delta \in [-1, 1], \\ & \min \quad (\delta), \\ & s.t. \quad C(x+\delta) = t, \end{aligned} \tag{7}$$

where $C(\cdot)$ represents the classification and $t$ represents the target classification result. We make several designs in our adversarial example's generator to satisfy the above requirements. We will elaborate these designs in the remainder of the section.

*5.2. The Clip Function.* We first design the solution for requirement (1). It is difficult to limit the generator to generate $\delta$, which will make $x'$ in an available range. But we found that we can set any value, which is out of the available range,

as the minimum or maximum value for the available range. It hardly affects the auditory effect. Thus, we designed a special clip function and the generator needs each clip point in $\delta$ by the function to keep the new utterance available; the clip function is defined as follows:

$$\text{clip}(x+\delta) = \min(\max(x+\delta, -1), 1). \quad (8)$$

The above function satisfies requirement (1) well through our experiments. Next, we need to design special loss functions to satisfy requirements (2) and (3).

*5.3. Generalized Relevancy Based Attack.* Based on the clip function, we designed loss functions to satisfy requirements (2) and (3). As we introduced in Section 4, we need our attack to be able to work under the black-box setting; in other words, the adversary can only get the classification results and the confidence of each speaker from the speaker verification system. The current work needs to estimate a special $\theta$ for the loss function [25]; $\theta$ will deeply influence the attack performance. Our generator excludes $\theta$'s influence, which lets us need not choose special $\theta$ before the training. We mainly designed our generalized relevancy loss function based on GE2E loss function; the loss function was represented as follows:

$$\text{Loss}_{\text{initial}} = -\text{Score}_t + \omega \times \log \sum_{k=1}^{N} \exp(\text{Score}_k), \quad (9)$$

where Score represents the confidence score feedback from the speaker verification system between $x'$ and the speaker, $t$ represents the target speaker number, $k$ represents the $k^{\text{th}}$ speaker number, and $\omega$ represents the reciprocal of the speaker quantity. This loss function will expand the distance between $x'$ and the nontarget identities and shrink the distance between $x'$ and the target identity, while the training process will not rely on special $\theta$.

*5.4. Multifactor Based Attack.* In our experiment, the generalized relevancy based loss function, which is elaborated in the last section, can guide the generator to generate adversarial examples that can attack the speaker verification system successfully. However, the distortion of adversarial examples will be beyond our tolerance; people will hear obvious noise in adversarial samples and find the attack easily. They are also easily recognized by machines. To this end, we designed a stronger generator with another loss function designed by us, which can achieve the spoofing attack imperceptibly. Hence, we divide our loss function into two parts: one is to achieve the attack and the other is to limit the amplitude perturbation. In particular, we limit the distortion with a special design in the loss function. Note that because the first goal of our generator is to generate a new utterance that can spoof the speaker verification system, we only utilize the part which can limit the noise in the loss function, after the new utterance $x'$ can be verified as the target identity. We utilized the multifactor based loss function to realize spoofing attack. We found that the distortion of successful adversarial examples is much less than before, while the total success rate is close to the result reported by generalized relevancy loss function based attack. We designed this loss function as follows:

$$\text{Loss} = (1-L) \times \text{Loss}_{\text{initial}} + L \times \frac{A}{\log(x) - \log(\delta)}, \quad (10)$$

where $L$ equals 0 when $x'$ was rejected by the speaker verification system; otherwise, $L$ equals 0, and $A$ is a constant. We designed our generator with this loss function. When we initially train the generator, $L$ is set as 0; it will try its best to achieve attack first when $x'$ has been accepted by the speaker verification system; $L$ will be reset as 1; then the generator begins to reduce the distortion, which may cause $x'$ to be rejected by the speaker verification system; then $L$ will be reset as 0. Two different requirements will compete through the whole training process. Thus, the generator can find the slightest perturbation inject into the original utterance to realize the attack and we will get adversarial examples that cannot be recognized by humans or machines.

## 6. Experiment Setup

*6.1. Data Partition and Experimental Environment.* In this section, we will describe the design of our experiments. To prove the efficiency of our attacks, we examined our spoofing attack with an open dataset TIMIT which has been used in many other voice-related works [26, 27]. This dataset includes 630 speakers and 6300 sentences. Firstly, we utilized 462 speakers from the TIMIT training set to train the embedding network which is based on the GE2E loss function. Then we extract the embedding vector from the testing set by the embedding network for further speaker verification. We deploy our attacks locally and randomly select 4 illegal speakers and 5 legal speakers from the testing set. For each illegal speaker, we selected 5 sentences and acquired 20 ($4 \times 5$) sentences; then we trained our generator to produce perturbation for each sentence targeting 5 legal speakers, respectively. Through this process, we obtained 100 ($5 \times 20$) adversarial examples for spoofing attacks. Then we test these adversarial examples on the speaker verification system. Two different classifications were employed in our work to show the performance of spoofing attack on machine learning solutions and similarity threshold solutions; we also test two loss functions that were introduced in Section 5 to show the performance of different spoofing attack. We conducted the experiments on a server with Ubuntu 16.04 and Intel Xeon CPU E5-2678 v3 @ 2.50 GHz with 125 G RAM. We set the learning rate $\eta$ as $1e-2$, $A$ as 20, and the epoch as 500.

*6.2. Metrics.* We employed different metrics for evaluating the above results. For the verification part, we employed false acceptance rate (FAR), false rejection rate (FRR), and average classification error (ACE). They were defined with the following equations:

$$\mathrm{FAR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}},$$
$$\mathrm{FRR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}}, \tag{11}$$
$$\mathrm{ACE} = \frac{\mathrm{FAR} + \mathrm{FRR}}{2},$$

where TP represents the amount of correctly classified positive samples, TN represents the amount of correctly classified negative samples, FP represents the amount of incorrectly classified positive samples, and FN represents the amount of incorrectly classified negative samples. Except that, we also employ equal error rate (EER) as metrics which is the error rate when FAR equals FRR. Then, for attack part, we utilized success rate (SR) as the metric, and it is defined as follows:

$$\mathrm{SR} = \frac{\mathrm{success\_attack\_count}}{\mathrm{attack\_count}}. \tag{12}$$

These metrics are widely utilized in evaluating performance of verification.

It is easy to evaluate the performance of classification and verification, but it is difficult to evaluate distortion directly. We need a quantitative method to calculate the distortion. We evaluate an utterance in decibels (dB) and a universal description of relative volume. The following equations represent the MNR of the adversarial samples:

$$\mathrm{dB}(x) = \max_i 20 \cdot \log_{10}(x_i), \tag{13}$$

$$\mathrm{dB}_x(\delta) = \mathrm{dB}(\delta) - \mathrm{dB}(x), \tag{14}$$

where $\delta$ represents the perturbation we inject into the utterance. Equation (14) represents the MNR of the attack utterance [16]. We can also write the equation about MNR as follows:

$$\mathrm{MNR} = \mathrm{dB}_x(\delta) = \mathrm{dB}(\delta) - \mathrm{dB}(x). \tag{15}$$

Since the perturbation is smaller than the original utterance, the result will be a negative number. The result is smaller, and the distortion is tinier. Although this result can describe the maximum distortion well, it cannot tell the overall distortion. So, we also introduced another metric SNR which is used in previous work [25]; it can be defined by the following equations:

$$\mathrm{SNR} = 10 \log_{10}\left(\frac{P_x}{P_\delta}\right), \tag{16}$$

where $P_x$ is the signal power of the original utterance and $P_\delta$ is the signal power of the injected perturbation. When $SNR$ is large enough, the human nearly cannot hear the noise in the utterance.

Our work utilized the above two works as our evaluation metrics for evaluating distortion. They, respectively, represented the maximum distortion which shows waveform alert in detail and the total influence of the distortion which shows waveform alert in total.

## 7. Evaluation

In this section, we will evaluate our spoofing attack on the state-of-the-art TI-SV based on deep learning.

*7.1. Performance without Attack.* We need to study the performance of our speaker verification system on the TIMIT dataset which can prove that the spoofing result under attack is caused by our spoofing attack rather than the system's poor performance. Thus, we first run an evaluation to prove that the speaker verification system can verify the identities of users before we evaluate the performance of our spoofing attack. We train the embedding vector extractor by the training data from the TIMIT dataset. After training, we randomly select 100 people for evaluating the performance of the speaker verification system. When the LDA was used, FAR = 5%, FRR = 5%, and ACE = 5%. Figure 2 shows the ROC curve when it uses cosine similarity. The EER equals 5% and the area under curve (AUC) can reach 0.99. Meanwhile, we found that when we set the threshold value at 0.59 the TI-SV has the best performance through the experiments. These results show whichever classification is used by the TI-SV, and the identities of users can be correctly verified. Then we can examine our spoofing attack on the TI-SV based on these results.

*7.2. Performance with Spoofing.* In this section, we will first evaluate the distortion of our generator, which can generate adversarial examples with different loss functions. Firstly, we produce adversarial samples by cosine similarity score and randomly show three waveforms from the same utterance; the result is shown in Figure 3; the yellow waveform is generated by generalized relevancy based attack, the red waveform is generated by multifactor based attack, and the black waveform is the original waveform. We can observe that the waveform generated by our multifactor based attack is more similar to the original waveform than the waveform generated by our generalized relevancy based attack. These waveforms show, on the intuitive, that the multifactor based attack will have better performance than generalized relevancy based attack on imperceptibility. Beyond that, we need to describe the distortion on a quantitative level. Firstly, a distribution diagram for MNR in Figure 4 is shown, and the ordinate value represents the quantity in this range. The average MNR for our multifactor based attack is −33 dB, and it is −18 dB for generalized relevancy based attack. The best-reported MNR for our multifactor based attack and generalized relevancy based attack is −77 dB and −22 dB, respectively. Our multifactor based attack is more imperceptible than generalized relevancy based attack in this metric. Besides, our best performance is also better than −45 dB, which was reported by previous work [16]. The MNR describes the distortion under the maximum scene, and we still need to describe the distortion under the global scene. Thus, a comparison of SNR was made between two attacks.

Figure 5 shows the distribution of SNR, and the ordinate value represents the quantity in this range. The average SNR

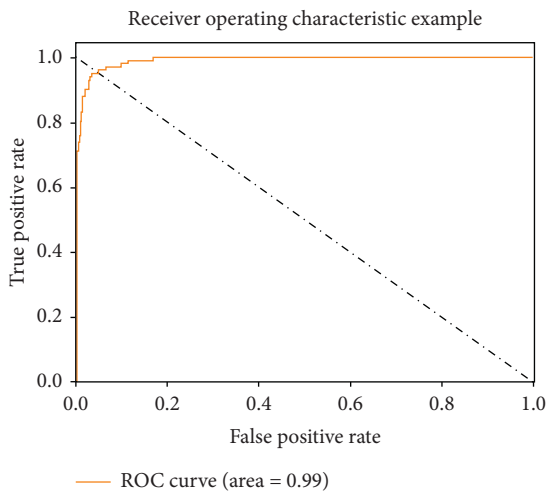



Figure 2: The ROC curve of the GE2E based TI-SV.

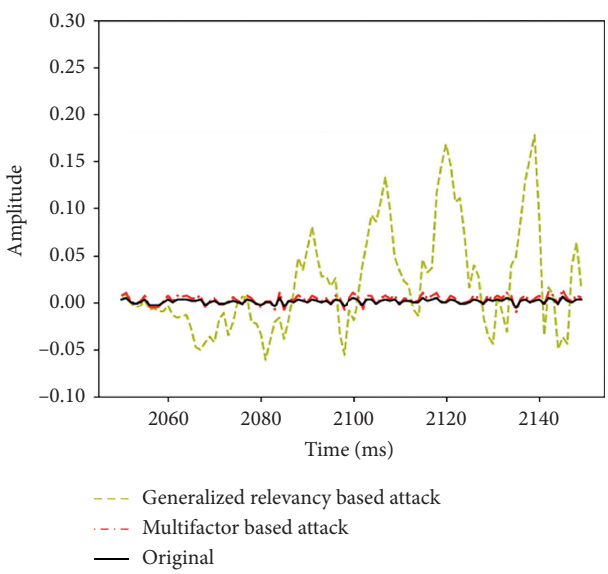



Figure 3: The different waveforms from the same utterance: (a) the yellow waveform is generated by generalized relevancy based attack, (b) the red waveform is generated by multifactor based attack, and (c) the black waveform is the original waveform.

for our multifactor based attack is 31 dB, and it is 17 dB for generalized relevancy based attack; meanwhile, the best result of SNR for multifactor based attack is 76 dB, which is larger than 26 dB that was reported by generalized relevancy based attack. The best performance is better than 31 dB in the previous work too. The above results can prove that our multifactor based spoofing attack will have better performance compared with previous work in terms of imperceptibility. In other words, our attack owns higher imperceptibility, which is a key feature for adversarial audio samples. After evaluating the imperceptibility of generalized relevancy based attack and multifactor based attack, we need to evaluate the performance of the SR in different classifications.

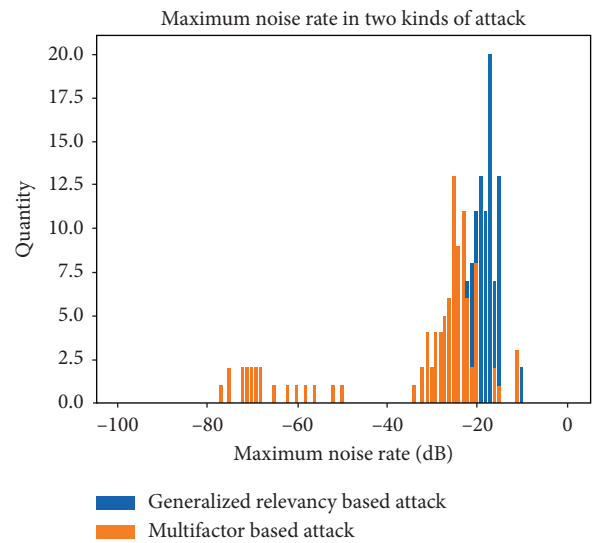



Figure 4: Maximum noise rate for generalized relevancy based attack and multifactor based attack. The ordinate value represents the quantity in this range.

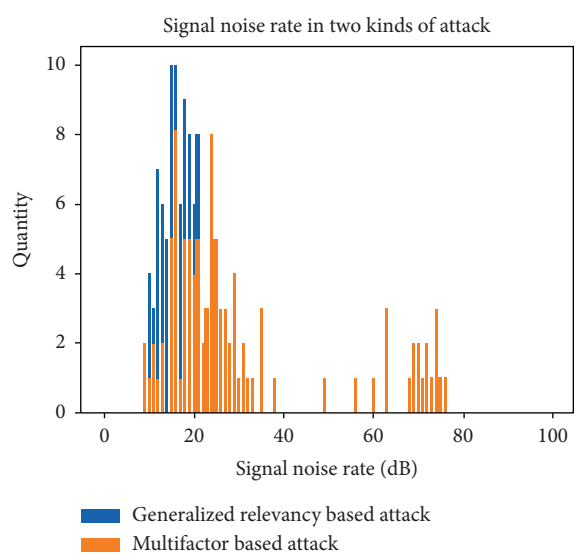



Figure 5: Signal noise rate for generalized relevancy based attack and multifactor based attack. The ordinate value represents the quantity in this range.

*7.2.1. Linear Discriminate Analysis.* We first tried to spoof the TI-SV, which uses the LDA to verify the identity of the user. We use the enrollments to train a two-class model used to distinguish the target speaker and nontarget speaker, which can realize a speaker verification task. The result is shown in SR1 of Table 1. Through the result, the SR of generalized relevancy based attack can achieve up to 83%. Meanwhile, the SR of multifactor based attack achieved up to 80%. This result proved that the two spoofing attacks can spoof the TI-SV with a similar SR when using LDA to achieve speaker verification.

*7.2.2. Cosine Similarity.* Since LDA and other machine learning classifications need training before using, which is

Table 1: The SR of spoofing attack. SR1 aims at LDA classification and SR2 aims at cosine similarity threshold classification. C1 represents the generalized relevancy based attack and C2 represents the multifactor based attack.

| | C1 | C2 |
|---|---|---|
| SR1 (%) | 83 | 80 |
| SR2 (%) | 86 | 82 |

inconvenient for enrolling a new user, as a more practical speaker verification system, the TI-SV can utilize cosine similarity and set a fixed threshold to verify the speaker's identity, which does not need training. We set the threshold value at 0.59 which has the best performance in Section 7.1. SR2 in Table 1 shows the results when we use generalized relevancy based attack and multifactor based attack. We can learn from the result that our multifactor based attack still has close performance to that of the generalized relevancy based attack. The SR for multifactor based attack is 82% and the SR for generalized relevancy based attack is 86%.

*7.2.3. Summary.* Through these results, we can learn that our multifactor based attack aimed at the state-of-the-art TI-SV has a high SR and it is more imperceptible, whatever the TI-SV model is based on machine learning classifications or threshold classifications. Our spoofing attack's SR is lower than that in the previous works; it is due to the fact that to our spoofing attack's target is the TI-SV, and it is more difficult to extract voiceprint than TD-SV that is targeted by previous works. Our multifactor based attack will have a slighter distortion with the original utterance, which is more imperceptible than the generalized relevancy based attack and previous works.

## 8. Attack Characteristics

The attack has two distinct characteristics: imperceptibility and target-independent ability. We will further analyze this part in the following paragraphs.

*8.1. Imperceptibility.* The imperceptibility will be changed continually during the procession of generation. It is important to analyze the procession of generation for showing the advantage of multifactor based attack on imperceptibility. We experiment for our attack to analyze the imperceptibility of the perturbation during the procession of generation. We randomly selected all sentences (20 sentences) in the illegal set and a random target in the legal set to observe the procession of generation. Figure 6 shows the average signal noise rate (dB) for each epoch. The result shows that the change of SNR is coincident between the multifactor based attack and the generalized relevancy based attack during the first phase of the procession. However, the multifactor based attack begins to increase the SNR, while the generalized relevancy based attack decreases the SNR persistently in the second phase. Note that the multifactor based attack not only stops the decreasing of the SNR but also explores the highest SNR for the successful attack. We can observe that the final SNR of multifactor based attack is larger than the end of the first phase and much larger than the final SNR of generalized relevancy based attack.

*8.2. Target-Independent Ability.* We consider that the voiceprint is only dependent on the biometric differences rather than the content. Thus, the factor of the physiological structure will determine the voiceprint. The previous work [28] has proven that the gender is an important factor for voice. Given that view, we analyze the target-independent ability by studying the influence of gender. We randomly selected ten sentences from different males and ten sentences from different females and partitioned them into two groups; each group includes five sentences from different males and five sentences from different females. We enrolled in the speaker verification system with users in one group. After that, we utilized sentences from females to attack the enrolled males; meanwhile, the same operation was done for sentences from males. Note that we use the cosine similarity for verification, since Section 7.1 has proven that it has the same efficacy as that of LDA. Table 2 shows the SR, average signal noise rate (ASNR), and average maximum noise rate (AMNR) after finishing the above experiment. The result shows that gender has little impact on the success rate and the perturbation of the attack. Our multifactor based attack has target-independent ability.

## 9. Discussion

In this section, we will discuss some advanced attack methods and defense methods for speaker verification systems.

*9.1. Universal Perturbation.* Current spoofing attacks need the generator to generate perturbations for each utterance, even targeting the same target. We hope our generator can produce a universal perturbation for a special target, which can use only one perturbation to realize the spoofing attack to a special victim. We can learn from some work that the voiceprint will exist in the tiny waveform [29], so it is possible to generate a stabilized perturbation that can include the whole voiceprint for one victim.

*9.2. Over-the-Air Injection.* Current spoofing attacks aimed at speaker verification systems can only work in the data layer or offer utterance for replaying attacks [25]. These restrictions limited the spoofing attack's range, which makes this attack unable to be utilized. An advanced attack could inject a slight perturbation when a legal speaker is verifying; the adversary can lead the speaker verification system into an incorrect permission space and all the legal user's operations in this space will be unsafe. Some attack in computer vision can spoof classifications by only changing one pixel's content [30]. If we can realize the spoofing attack by only changing one or several points in the utterance, we will have the ability to repeat playing the short perturbation over the air to attack the system when a legal speaker is verifying. Even more, we

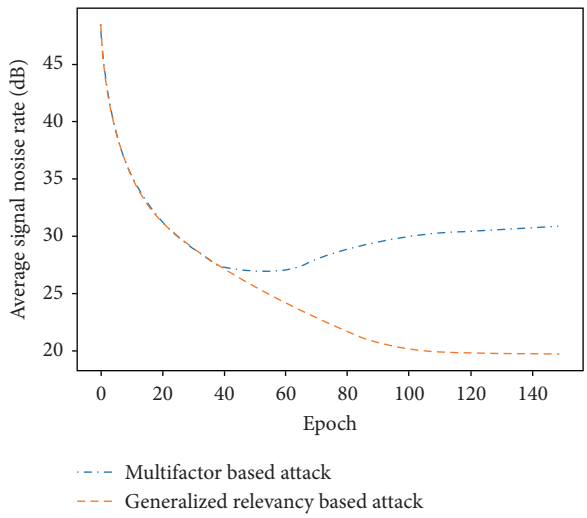



Figure 6: Signal noise rate for generalized relevancy based attack and multifactor based attack during the procession of generation.

Table 2: The successful rate, signal noise rate, and maximum noise of the analysis of the gender. "Male" and "female" represent the genders of the sentence's owner that is utilized to attack.

| | Male | Female |
|---|---|---|
| SR (%) | 88 | 84 |
| SNR (dB) | 38 | 30 |
| MNR (dB) | −35 | −31 |

can use ultrasound to complete injection instead of audible sound by the technology proposed in previous work [25]. It will further enhance the imperceptibility of the spoofing attack, which makes the attack more practical.

*9.3. Mitigation of Multifactor Based Attack.* Since our attack is effective, while it is hard to be detected by human or current speaker verification systems, we will discuss several possible defense methods as follows: A detection method is to train a detector using normal utterances and adversarial utterances. In the computer vision domain, they can employ a detector to detect the adversarial samples [31], although this work has a high false positive rate, and it is not robust when the adversary is aware of this defense. It also has the ability to distinguish between normal utterances and adversarial utterances.

Another method is adding transformation for the input (e.g., bit-depth reduction and JPEG compression [32] for images). We can mitigate the attack by applying input transformation such as a bit-depth reduction. Because this process will reduce the information in an utterance including injected perturbation, our attack will not succeed.

## 10. Conclusion

In this paper, we explore the vulnerability of the speaker verification system which will affect the security of users' economics, privacy, and even safety. We first conduct imperceptible audio adversarial examples to attack the state-of-the-art deep-learning-based TI-SV by our generalized relevancy based attack and multifactor based attack. We evaluate generalized relevancy based attack and multifactor based attack in two patterns to verify the speaker including both machine learning method and threshold method. The multifactor based attack can achieve 82% SR, when the distortion in the utterance only has MNR −77 dB and SNR 76 dB, which are much better than the values in the previous works. Our work also gives out several advanced attacks with a theoretical foundation, which will influence real life a lot. Due to the fact that our effective attack reveals the vulnerability of the speaker verification system, we also proposed several defense methods to mitigate the insecure problems of speaker verification systems.

## Data Availability

The voice data used to support the findings of this study have been deposited in the TIMIT repository (https://catalog.ldc.upenn.edu/LDC93S1).

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61972348, in part by the National Key Research and Development Program of China under Grant 2018YFB0803600, and in part by the Leading Innovative and Entrepreneur Team Introduction Program of Zhejiang under Grant 2018R01005.

## References

[1] Q. Zheng, A. Kumar, and G. Pan, "Contactless 3d fingerprint identification without 3d reconstruction," in *Proceedings of the 2018 International Workshop on Biometrics and Forensics, IWBF 2018*, pp. 1–6, IEEE, Sassari, Italy, June 2018.

[2] X. Wei, H. Wang, B. Scotney, and H. Wan, "Selective multi-descriptor fusion for face identification," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 12, pp. 3417–3429, 2019.

[3] Y. Chen, C. Wu, and Y. Wang, "T-center: a novel feature extraction approach towards large-scale iris recognition," *IEEE Access*, vol. 8, pp. 32365–32375, 2020.

[4] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*, pp. 5329–5333, IEEE, Calgary, Canada, April 2018.

[5] L. Wan, Q. Wang, A. Papir, and I. Lopez-Moreno, "Generalized end-to-end loss for speaker verification," in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*, pp. 4879–4883, IEEE, Calgary, Canada, April 2018.

[6] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016*, pp. 5115–5119, IEEE, Shanghai, China, March 2016.

[7] N. Dehak, L. J. Rodríguez-Fuentes, and E. Lleida, "I-vector representation based on GMM and DNN for audio classification," in *Proceedings of the Odyssey 2016: The Speaker and Language Recognition Workshop*, ISCA, Bilbao, Spain, June 2016, http://www.isca-speech.org/archive/Odyssey_2016/abstracts/Najim.html.

[8] D. A. Reynolds, "Universal background models," in *Encyclopedia of Biometrics*, S. Z. Li and A. K. Jain, Eds., Springer US, New York, NY, USA, pp. 1349–1352, 2009.

[9] Q. Li, H. Zhu, Z. Zhang, R. Lu, F. Wang, and H. Li, "Spoofing attacks on speaker verification systems based generated voice using genetic algorithm," in *Proceedings of the 2019 IEEE International Conference on Communications, ICC 2019*, pp. 1–6, IEEE, Shanghai, China, May 2019.

[10] Z. Wu and H. Li, "Voice conversion and spoofing Attack on speaker verification systems," in *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA 2013*, pp. 1–9, IEEE, Kaohsiung, Taiwan, November 2013.

[11] E. Variani, X. Lei, E. McDermott, I. Lopez-Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014*, pp. 4052–4056, IEEE, Florence, Italy, May 2014.

[12] M. Zha, G. Meng, C. Lin, Z. Zhou, and K. Chen, "Rolma: a practical adversarial attack against deep learning-based LPR systems," in *Proceedings of the Information Security and Cryptology-15th International Conference, Inscrypt 2019*, Nanjing, China, December 2019.

[13] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2006*, F. Lin, D. Lee, B. P. Lin, S. Shieh, and S. Jajodia, Eds., , March 2006.

[14] B. Biggio, I. Corona, D. Maiorca et al., "Evasion attacks against machine learning at test time," in *Proceedings of the Machine Learning and Knowledge Discovery in Databases-European Conference, ECML PKDD 2013*, H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezný, Eds., pp. 387–402, Springer, Prague, Czech Republic, September 2013.

[15] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, Cohen, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the Machine Learning, Twenty-Third International Conference (ICML 2006)*, A. W. Moore, Ed., pp. 369–376, ACM, Pittsburgh, Pennsylvania, USA, June 2006.

[16] N. Carlini and D. A. Wagner, "Audio adversarial examples: targeted attacks on speech-to-text," in *Proceedings of the 2018 IEEE Security and Privacy Workshops, SP Workshops 2018*, pp. 1–7, IEEE Computer Society, San Francisco, CA, USA, May 2018.

[17] X. Liu, K. Wan, Y. Ding, X. Zhang, and Q. Zhu, "Weighted-sampling audio adversarial example attack," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 4908–4915, 2020.

[18] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *Proceedings of the International Conference on Machine Learning. PMLR*, pp. 5231–5240, Long Beach, CA, USA, June 2019.

[19] H. Fujimura, N. Ding, D. Hayakawa, and T. Kagoshima, "Simultaneous flexible keyword detection and text-dependent speaker recognition for low-resource devices,," in *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2020*, M. D. Marsico, G. S. di Baja, and A. L. N. Fred, Eds., pp. 297–307, Scite Press, Valletta, Malta, February 2020.

[20] W. Wang, Y. Zhang, J. Xu, and Y. Yan, "Multiple temporal scales based speaker embeddings learning for text-dependent speaker recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019*, pp. 6311–6315, IEEE, Brighton, UK, May 2019.

[21] R. Jahangir, Y. W. Teh, N. A. Memon et al., "Text-independent speaker identification through feature fusion and deep neural network," *IEEE Access*, vol. 8, pp. 32187–32202, 2020.

[22] F. Zhao, H. Li, and X. Zhang, "A robust text-independent speaker verification method based on speech separation and deep speaker," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019*, pp. 6101–6105, IEEE, Brighton, UK, May 2019.

[23] Q. Wang, X. Lin, M. Zhou et al., "Voicepop: a pop noise based anti-spoofing system for voice authentication on smartphones," in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2062–2070, IEEE, Paris, France, May 2019.

[24] C. Yan, Y. Long, X. Ji, and W. Xu, "The catcher in the field: a fieldprint based spoofing detection for text-independent speaker verification," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '19*, pp. 1215–1229, Association for Computing Machinery, New York, NY, USA.

[25] G. Chen, S. Chen, L. Fan et al., "Who is real bob? adversarial attacks on speaker recognition systems," in *Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP)*, IEEE, San Francisco, CA, USA, May 2021.

[26] A. Bhowmick, A. Biswas, and M. Chandra, "Performance evaluation of psycho-acoustically motivated front-end compensator for TIMIT phone recognition," *Pattern Analysis and Applications*, vol. 23, no. 2, pp. 527–539, 2020.

[27] M. T. S. Al-Kaltakchi, W. L. Woo, S. S. Dlay, and J. A. Chambers, "Comparison of i-vector and GMM-UBM approaches to speaker identification with TIMIT and NIST 2008 databases in challenging environments," in *Proceedings of the 25th European Signal Processing Conference, EUSIPCO 2017*, pp. 533–537, IEEE, Kos, Greece, September 2017.

[28] Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: recognizing speech from gyroscope signals," in *Proceedings of the 23rd USENIX Security Symposium*, K. Fu and J. Jung, Eds., pp. 1053–1067, USENIX Association, San Diego, CA, USA, August 2014, https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/michalevsky.

[29] Y. Wang, Y. Wang, and T. Tan, "Combining fingerprint and voiceprint biometrics for identity verification: an experimental comparison," in *Proceedings of the Biometric Authentication, First International Conference, ICBA 2004*, D. Zhang and A. K. Jain, Eds., pp. 663–670, Springer, Hong Kong, China, July 2004.

[30] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[31] Q. Guo, J. Ye, Y. Hu et al., "A multivariant partition-based method for audio adversarial examples detection," *IEEE Access*, vol. 8, pp. 63 368–463 380, 2020.

[32] S. Z. Li and A. K. Jain, Eds., *Encyclopedia of Biometrics*p. 875, Springer US, New York, NY, USA, 2009.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 8842539, 14 pages
https://doi.org/10.1155/2020/8842539

*Research Article*

# GroupTracer: Automatic Attacker TTP Profile Extraction and Group Cluster in Internet of Things

**Yixin Wu,[1] Cheng Huang,[1] Xing Zhang,[2] and Hongyi Zhou[2]**

[1]*College of Cybersecurity, Sichuan University, Chengdu 610065, China*
[2]*NSFOCUS, Beijing 100089, China*

Correspondence should be addressed to Cheng Huang; opcodesec@gmail.com

Received 3 September 2020; Revised 3 November 2020; Accepted 19 November 2020; Published 4 December 2020

Academic Editor: Ting Chen



As Advanced Persistent Threat (APT) becomes increasingly frequent around the world, security experts are starting to look at how to observe, predict, and mitigate the damage from APT attacks. In the meantime, the Internet of things devices are also risky and heavily exposed to the Internet, making them more easily used by hacker organizations to launch APT attacks. An excellent attacker can take down millions of Internet of things devices in a short time. Once the IoT botnet is built, attackers can use it to launch complex attacks which could damage Internet infrastructure and cause network disconnection. This paper proposes GroupTracer, a framework for observing and predicting the Internet of things attacks. GroupTracer is designed to automatically extract the TTP profiles (i.e., tactics, techniques, and procedures) that can describe the behavior of attackers through their tactics, techniques, and processes and dig out the potential attacker groups behind complex attacks. Firstly, it captures attacks by IoT honeypots and extracts relevant fields from logs. Then, attack behaviors are automatically mapped to the ATT&CK framework to achieve automatic TTP profiles extraction. After that, GroupTracer presents four feature groups, including TTP profiles, Time, IP, and URL features, a total of 18 features, mines potential attack groups through hierarchical clustering algorithm, and compares the clustering results with two baseline algorithms. As the ground truth labels are unknown, we apply three internal validation indexes to evaluate the cluster quantity. Experimental results showed that the proposed framework has achieved an excellent performance in exploiting potential groups as the Calinski–Harabasz index reaches 3416.93. Eventually, attack trees are generated for each cluster where nodes indicate attack commands and edges represent command sequences. These attack trees could help better understand each attack group's actions and techniques.

## 1. Introduction

The Global Research and Analysis Team (GReAT) at Kaspersky points out that the Advanced Persistent Threat (APT) activity has become increasingly complex and destructive [1] since these APT groups launch targeted attacks on critical infrastructure and attempt to compromise central networks. Meanwhile, the Internet of things has become the no. 1 security threat to personal privacy, corporate information security, and even critical infrastructure since IoT devices are inherently risky and easy to exploit while being heavily exposed to the Internet. What is worse, attackers can utilize open-source tools to quickly assemble malware that can scan, penetrate, and control IoT devices. Excellent hackers can take down millions of IoT devices in a short time. Once IoT botnets are formed, attackers can launch an APT attack to hazard the Internet infrastructure and cause network disconnections (e.g., Dyn cyberattack [2] and VPNFilter event [3]). The emerging challenge is how to observe and predict attacks on IoT devices by individuals or even attacker groups since the number of attacks on IoT devices, which are perfect tools for APT attacks, has risen dramatically.

*1.1. Describing Individual Behavior.* While behavior detection methods for attacks are mostly based on Indicators of Compromise (IOCs) extracted from rule-based methods or traditional blacklists, the information conveyed by such

IOCs is not enough to describe the abundant and varied network security environment due to the following reasons:

(i) IOC is unstable and is easily changed by attackers. For example, if adversaries are leveraging an anonymous proxy service like Tor, they may change IPs quite frequently with little effort and never be noticed.

(ii) IOC cannot express how the attacker interacts with the victim system, and the process of the attack cannot be represented.

(iii) Redundancy occurs when IOC is used to express an attack. In other words, more IOCs do not necessarily lead to a better description.

Bianco proposed the Pyramid of Pain [4], in which each level of the pyramid represents different types of attack indicators leveraged to detect the activities of the adversary, and the most valuable attack indicator is attacker TTPs. TTP profile [5] describes the flow that adversaries go through to accomplish their mission, from initial access to impact and at every step in between, which is abundant to support a comprehensive analysis of the aggressive behaviors of individuals or attack groups. Meanwhile, the defense is shifting from vulnerability-centric to threat-centric, and flexible and efficient security architecture can only be constructed with a sufficient understanding of the threat of the critical assets, which depends on an overall comprehension of the attack tactics, techniques, and behavior patterns (i.e., TTPs). However, at this stage, there is no mature method to normalize the description of attacks on IoT devices and map them to the analysis model. A method for automatic TTP profile extraction of IoT device attacks is expected.

*1.2. Clustering Attackers into Groups.* With the rapid growth of APT activities, the evolution of a threat landscape moves from a single hacker to well-organized attack actor groups (e.g., Darkhotel [6] and Turla [7]). How to find and depict the behavior of an attack actor group among an ocean of attacks becomes a challenge. Behavioral analysis in sandboxes [8, 9] and binary analysis [10, 11] seem like pleasant ways, which can match malicious samples used by attackers to known or novel malicious families and capture their behaviors to observe the similarities between these attackers. However, malicious family and attack group have a many-to-many relationship, and we cannot just rely on the analysis of malicious samples to find the group behind attacks. Considering the excellent performance of the data-driven approach in the field of network security [12–14], we try to tackle the challenges from a data-driven perspective.

Given the challenges presented above, this paper aims to develop mapping knowledge bases from attacker payloads to the ATT&CK framework to extract the TTP profile and generate behavior fingerprint for attackers to discover groups behind active campaigns. The ultimate purpose is to observe the behavior of attack actor groups and predict attacks in the Internet of things.

*1.3. Contributions.* Three critical contributions of the paper are as follows:

(i) *Comprehensive Description of Attacker Behavior.* GroupTracer leverages four feature groups (TTP profile, Time, IP, and URL) that are derived from log data to characterize different actions of attackers, which addresses the emerging challenge of the observation and prediction of attacks on IoT devices by individuals. The TTP profile depicts the technique, tactic, and procedure of the attacker. The Time feature group provides statistical characteristics based on attack duration, number of attacks, and time zone of the attacker. The IP and URL feature groups both involve the type of IP/URL and the malicious index, while the latter also analyzes the download file.

(ii) *Automatic TTP Profile Extraction.* Considering that the data source is honeypot log data, which collects payloads utilized by attackers, we construct the 1st and 2nd knowledge bases, which store the mappings between commands and TTPs. By using these knowledge bases, GroupTracer maps commands derived from payloads to the ATT&CK framework to extract the TTP profile, which bridges the gap between cyber threat intelligence (CTI) and the attacker.

(iii) *Group Cluster and Attack Tree Generation.* GroupTracer proposes four feature groups and hierarchical clustering algorithm to build attack group cluster model which aims at finding out the potential groups behind complex attacks. In order to better understand each attack group's behaviors, GroupTracer also introduces attack tree construction method where nodes describe attack commands and edges represent command sequences. The evaluation result shows that GroupTracer can achieve excellent performance as the Calinski–Harabasz index reaches 3416.93.

The remainder of this paper is organized as follows: Section 2 explains the related work about the fundamental techniques used in our framework. Section 3 presents the data collection, flow of feature processing, application of clustering algorithm, and attack tree creation in GroupTracer. The entire experiment and evaluation process is elaborated in Section 4. Finally, the conclusion and future work are discussed in Section 5.

## 2. Related Work

*2.1. Application of IoT Honeypot.* To specialize in cyberattacks and defend against them, tools for proactive defense are presented. For instance, honeypot that can capture attacks, document intrusion information about instruments and behaviors of hackers, and prevent attacks outbounding the compromised system [15] has been widely leveraged in cybersecurity. Due to the vulnerable and destructive nature of IoT devices [16–18], the number of

IoT honeypots based on different protocols is rapidly increasing. Currently, some IoT honeypots have already existed [19]. The work in [20] utilizes IoTPOT, a novel honeypot that stimulates the Telnet-enabled IoT devices, which handles commands sent by attack actors, analyzes malicious families on different CPU architectures, and provides an in-depth analysis of ongoing attack behavior. However, IoTPOT focuses on observing the characteristics of malicious families (e.g., spread tendency and ultimate goal) and relationships between these families. It does not employ existing data to analyze the behavior of the aggressors behind attacks and associations between them in detail, which is the center of cyber threat intelligence. A honeypot that emulates the ZigBee gateway and aims at assessing ZigBee attack intelligence and IoT cyberattack behavior is proposed in the text [21]. Although this paper analyzes the commands in the honeypot data at great length and classifies them into six categories of attacks, it does not mine the TTP of these attacks, which helps analysts in threat modeling. Heo and Shin [22] analyze the connection-level log data to study Telnet service scanning to provide solid evidence for the existence of IoT botnet, whereas the dataset contains only connection metadata, so there is no way to analyze the payload in packages, which is a critical evidence for attacking, thus not entirely convincing.

In conclusion, most published methodologies have focused on a single service such as Telnet and ZigBee and analyzed features of malicious families. GroupTracer is more widely used for protocols where command execution vulnerabilities occur and depicts the characteristics of attack behaviors. Besides, most of the previous studies have not analyzed payloads at the TTP level, and some even have not analyzed payloads at all. GroupTracer extracts attack techniques, tactics, and procedures (TTPs) from payloads and utilizes payloads to build attack trees for potential attack groups to more specifically demonstrate their attack behaviors.

*2.2. Cyber Threat Intelligence and TTP Extraction.* Gartner defines cyber threat intelligence (CTI) as evidence-based knowledge, which can be utilized to inform decisions concerning the subject's response to menace or compromise [23]. With the rapid evolution of the cyber threat landscape, the demand for high quality and fast speed of CTI exchange that allows organizations to respond to emerging threats at the tactical level is becoming increasingly urgent. TTP describes the techniques, tactics, and attack patterns used by the adversary and can be presented in structured text formats that meet the high demand. Husari et al. [24] develop TTPDrill that can achieve the automatic and context-aware analysis of CTI to generate TTPs precisely. Their work bridges the gap between unstructured cyber threat intelligence and structured techniques, tactics, and procedures. Nevertheless, their data source is the cyber threat intelligence, which means that only after CTI is produced, can TTPDrill construct a complete attack pattern. Our work aims at decreasing the time-to-defend even more.

*2.3. Group Cluster.* Clustering and correlating have been studied extensively and are employed in a multitude of data-driven domains, including security and privacy areas [25]. In a similar direction to this paper, the work in [26] applies an unsupervised method to characterize and classify security-related anomalies and attacks that exist in honeypots without learning phase, labeled traffic, or attack signature database. Cho et al. [27] compare the similarity of the distributed domain to predict the same group, which provides the possibility of response to future attacks. Azevedo et al. correlate IOCs from different OSINT feeds and cluster them to obtain enriched IOCs. This work allows the identification of attacks that was impossible by analyzing IOCs individually. One work that inspires us comes from Ghiëtte et al. [28]. They dissect the SSH protocol to fingerprint tools based on cipher suites and SSH version strings, employing key exchange algorithms and SSH banners to cluster similar tool usage into collaborating individuals and even campaigns. However, as [4] said, adversaries can employ or create another tool that has the same capability to evade detection.

By comparison, GroupTracer employs honeypot log data, in which timestamps, IP addresses, and payloads sent by attackers are usually recorded, and considers four different perspectives (e.g., TTPs and Time) to generate feature groups. For example, it generates TTP profiles by mapping payloads to ATT&CK framework based on command characteristics. Then, it clusters similar adversaries' behaviors into groups based on these features. Our work draws on the strengths of the mentioned studies and improves their weaknesses.

## 3. Framework

The ultimate goal of this paper is to automatically extract TTP profiles and cluster attack groups in the Internet of things. Figure 1 overviews the flow of GroupTracer. Firstly, it captures attacks, generates raw data, and extracts features from specific fields (e.g., timestamp, payload, and timezone). We deploy numerous honeypots on the Internet to capture attacks. Secondly, it enriches these features. As for generating the TTP profile feature group, it cuts payload into commands, maps these commands to the ATT&CK framework, and then generates Abstract Syntax Tree of the commands for a second mapping to techniques and tactics. After generating all feature groups, encoding and TF-IDF algorithm are utilized to vectorize these string-type features. Thirdly, it combines all feature vectors and leverages the hierarchical clustering algorithm to cluster these attackers into groups. Finally, attack trees for each group, where nodes are commands and edges are command sequences, are created from their payloads to characterize attack profiles.

*3.1. Raw Data Collection.* The log format of open-source honeypot contains general fields and particular fields. There are 12 general fields standard in all honeypots. src_ip is the source IP, and src_port is the source port number. sensor_ip and dst_port represent the IP and destination port number

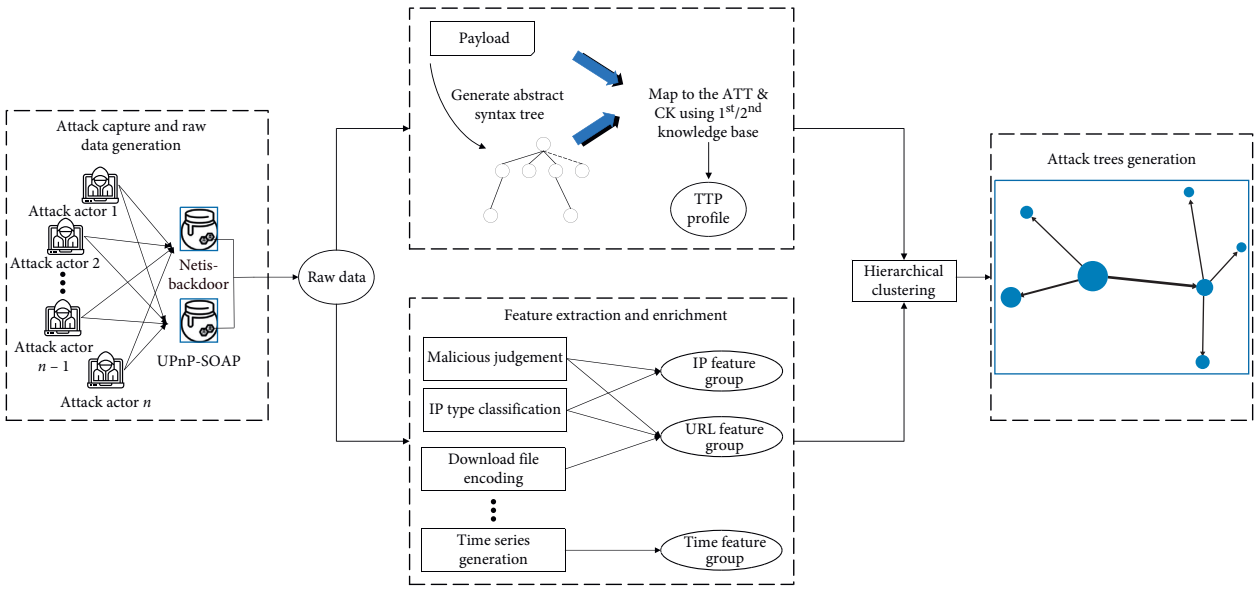



Figure 1: The architecture of GroupTracer.

of the honeypot, respectively. timestamp represents the time when the event occurred. protocol represents the underlying protocol used by the honeypot. geoip describes the current geographic location of the IP and related information like time zone, which is usually obtained by the external API GeoLite [29]. pot_version is added by the honeypot developer to indicate the current version of the honeypot. fingerprint is mostly a hash value generated from a string of src_ip, timestamp, and a specific honeypot-specific field. log _type depicts the type of event recorded in the log, and honeypot_type describes the type of honeypot. container_id represents the ID of the docker container. The particular field is determined according to the specific protocol used by the IoT honeypot. GroupTracer mainly leverages three general fields, namely, src_ip field, *timestamp* field, and geoip field in honeypot log data, as these three fields can provide information to generate the Time and IP feature groups. Given that the payload may appear in different fields, GroupTracer will accurately locate the corresponding field through string matching. To ensure the universality of the framework and the integrity of the payload, the contents of all fields that have payloads are spliced.

*3.2. Feature Extraction.* The following subsections detail how GroupTracer converts these fields into four feature groups, namely, the TTP profiles, Time, IP, and URL. src_ip field is considered to be the primary key in all fields because the probability of an IP being used by multiple groups is minimal, even if the individual IP is assigned dynamically.

*3.2.1. TTP Profile Feature Group Generation.* The TTP profile consists of tactics and techniques used by attack actors. Tactic depicts the common strategy of a threat action (e.g., execution and defense evasion). ATT&CK framework provides 12 categories for corresponding techniques. GroupTracer extracted these names as tactics in TTP profiles. Technique describes attack techniques implemented by attack actors under a specific tactic. For example, defense evasion is a tactic that can be performed by a technique named clear command history.

GroupTracer produces the TTP profile primarily from a command execution perspective. There is a crucial issue to be compromised. On the one hand, the classification of commands should be as accurate as possible. On the other hand, command parameter values sometimes affect the classification needlessly. The following two commands illustrate both cases. Both of these commands can be classified as technique file deletion. (i) can be further subdivided into technique clear command history, whereas (ii) can only be classified as technique *file deletion*. For (i), the whole statement is a valid classification basis, while for (ii), only *rm* can serve as a valid classification basis, and all other parts are redundant.

(i) *rm -rf* ~/.bash_history

(ii) *rm -rf xb.sh xb.sh xb2.sh xb1.sh*

Aiming at solving the above dilemma, the quadratic mapping method is proposed. Figure 2 illustrates the process of quadratic mapping. There are two knowledge bases in GroupTracer. One saves the mapping of the entire command statement to tactics and techniques for the first mapping, and the other stores the mapping of the abstract command structure to tactics and techniques for the second mapping. In the first mapping, GroupTracer splits the original payload into several commands and maps these commands to the 1st knowledge base to attain some of the TTP profiles. As shown in Figure 3, GroupTracer generates the Abstract Syntax Tree [30] for each payload to obtain command nodes and extract abstract command structures. Then, these abstract structures are put as input into the 2nd knowledge base to acquire the

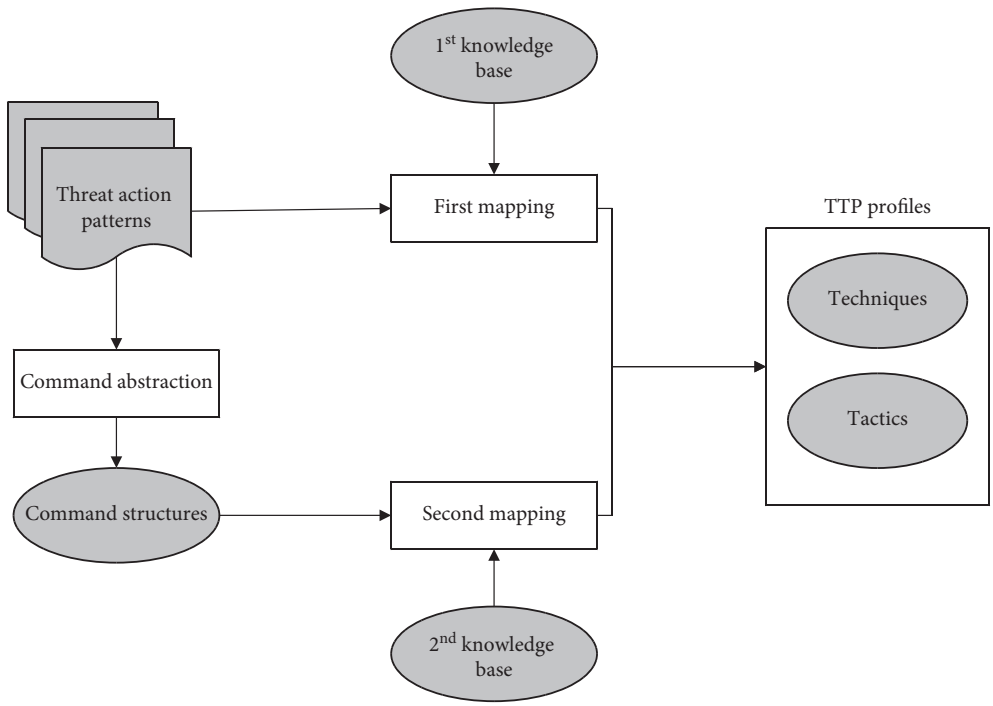



Figure 2: The process of generating the TTP profile. GroupTracer cuts payload into commands, maps these commands to the ATT&CK framework, and then abstracts the structure of the commands for a second mapping to techniques and tactics. The product of the first mapping and the second mapping constitutes the TTP profiles.

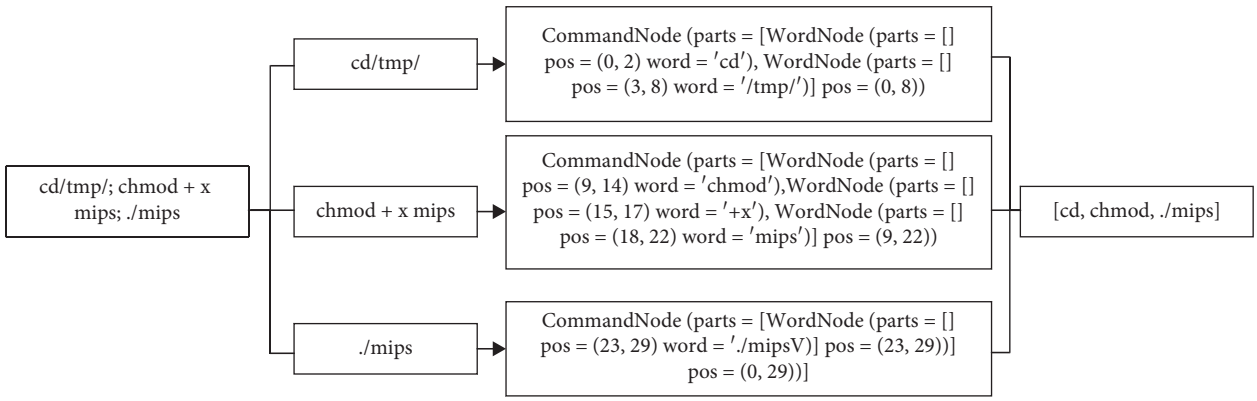



Figure 3: GroupTracer generates the Abstract Syntax Tree for each payload to obtain command nodes and then extracts the abstract structure from command nodes.

remaining TTP profiles. Table 1 only shows some mapping samples due to space limitation. After obtaining the TTP profiles, the GroupTracer encodes the corresponding string into a numeric feature vector. The final data structure can be described as follows:

$$\text{TTP} = [\text{techniques}, \text{tactics}], \tag{1}$$

where there are variable-length techniques and tactics.

*3.2.2. IP and URL Feature Groups Generation.* The features related to IP and URL feature groups are shown in Table 2, the first three of which are common to both feature groups, and the last one is unique to the URL feature group. The country can be obtained through Ipdb [31]. VirusTotal [32] provides multiple antivirus scanning engines (e.g., Kaspersky URL advisor, Malware Domain Blocklist [33], and Dr.Web Link Scanner [34]) used for URL scanning. GroupTracer first utilizes VirusTotal to scan for unknown IPs/URLs and then regards the number of antivirus engines that return malicious results as the malicious index for those IPs/URLs. According to the purpose, there are seven types of IP addresses shown in Table 3. This framework uses the RTBAsia API [35] to get the classification of IP types. Download is an optional option for the URL feature group, depending on the type of vulnerability in the honeypot, because in some cases the download is to install a backdoor for subsequent manipulation, while in others it is to provide

Table 1: Some examples of mapping from commands to TTP profiles.

| Command | Technique | Tactic |
|---|---|---|
| *1st knowledge base* | | |
| show running-config | Credential dumping | Credential access |
| show startup-config | Credential dumping | Credential access |
| *2nd knowledge base* | | |
| ftpget | Remote file copy | Lateral movement |
| wget | Remote file copy | Lateral movement |
| curl | Remote file copy | Lateral movement |
| rcp | Remote file copy | Lateral movement |
| copy | Remote file copy | Lateral movement |
| show archive config | Credentials in files | Credential access |
| show history | Input capture | Collection |
| show logging | Input capture | Collection |
| tar | Data compressed | Exfiltration |
| zip | Data compressed | Exfiltration |
| rar | Data compressed | Exfiltration |
| shutdown | System shutdown/reboot | Impact |
| reboot | System shutdown/reboot | Impact |
| del | File deletion | Defense evasion |
| rm | File deletion | Defense evasion |
| adduser | Create account | Persistence |
| usermod | Account manipulation | Persistence |
| groupadd | Account manipulation | Persistence |
| dir | File and directory discovery | Discovery |
| ls | File and directory discovery | Discovery |
| cd | File and directory discovery | Discovery |
| echo | Data from local system | Collection |
| cat | Data from local system | Collection |
| more | Data from local system | Collection |
| pwd | Data from local system | Collection |
| whoami | Data from local system | Collection |

some tools for privilege escalation under certain circumstances. When attacking different honeypots, it is likely that hackers in a group download file for different purposes, which will greatly affect our performance in clustering only by downloading file names. After attaining these feature groups, the GroupTracer encodes the corresponding string into a numeric feature vector.

*3.2.3. Time Feature Group Generation.* As the attacker groups tend to use the tool framework to attack the specified target, there is a corresponding regularity in the IP time zone, attack duration, number of attacks, etc. Algorithm 1 describes the generation of the basic time-series features for a given $T_{ip}$, which represents the set of timestamp for a specific IP. $\widehat{T}$ is a collection of $T$ used as input to generate the start time, the attack duration, and the number of attacks in each duration. In addition, GroupTracer reuses this code to select the final threshold. When selecting threshold, we nest a for loop in the outermost layer, and the threshold value increases by 1. Meanwhile, we count the number of time_interval in each $T_{ip}$ and assign their sum to the variable num_interval. If num_interval has not changed compared to the previous loop, we jump out of the cycle and output the final threshold.

The threshold, which indicates how small the time interval between two attacks is before they are considered to belong to the same attack period, is utilized to divide the attack duration. Partitioning each attack period and characterizing the behavior (e.g., duration and number of attacks) of each attack steadily can be the key to the reliability of time analysis, since the members of a group may be similar in these respects. To select a number as the initial threshold, GroupTracer maps all the time interval values into the following 5 time buckets: <1 s, [1 s, 1 min], (1 min, 1 h], (1 h, 1 day], >1 day. The results show that 99.91% of time gaps fall into the first four buckets, so the initial threshold is set to 1 h. Then, GroupTracer employs Algorithm 1 to accomplish the threshold selection procedure. It first counts the total number of attack durations for each IP and then adjusts the threshold slowly until the number of attack periods for most IPs is almost unchanged. If multiple thresholds have the same result, choose the smaller one first, as the threshold always tends to choose the smaller one.

After we get the final threshold and basic time-series features, we extract the statistical features from the basic ones, namely, the start time, the attack duration, and the number of attacks in each attack duration. GroupTracer draws a new time series for the relevant features of each IP, taking the start time as the independent variable and the attack duration and number of access as the dependent variables. Several features that proved to be significant time-series characteristics in an early stage shown in Table 4 are applied. After generating the time series, GroupTracer will automatically calculate these selected features to produce the final feature vectors.

By encoding the timezone field appearing in the data, GroupTracer finally turns the string-type features into integer vectors. Eventually, statistical features from timestamp and encoded time zone constitute the Time feature group.

*3.3. Group Clustering Algorithm.* As the hacker groups tend to utilize customized frameworks, features like time gap and TTP profiles have specific patterns. Therefore, attacker behavior naturally forms clusters. This paper is aimed at identifying such natural clusters to dig out potential groups behind active campaigns.

We employ the well-known hierarchical clustering algorithm [36] as it captures the hierarchical structure between clusters, which helps security experts to observe the relationship between clusters and subclusters. Moreover, hierarchical clustering is suitable for arbitrary shape clustering and is insensitive to the input order of samples.

Table 2: Features related to IP & URL feature groups.

| # | Feature name | Description |
|---|---|---|
| 1 | Country | Describes the country to which the IP/URL belongs. |
| 2 | Malicious index | Leverages the VirusTotal API to determine the maliciousness of the IP/URL. |
| 3 | IP address type | Utilizes the RTBAsia API to classify IP/URL type. |
| 4 | Download (optional) | The file that the attack actor downloaded by executing the command. |

Table 3: Seven types of IP addresses.

| # | Types |
|---|---|
| 1 | Internet data center |
| 2 | Fixed IP Internet access line |
| 3 | Ordinary broadband |
| 4 | Mobile broadband |
| 5 | Backbone node |
| 6 | Known crawler API |
| 7 | Small operator |

Figure 4 depicts a bottom-up approach to perform hierarchical clustering. Each sample represents a unique cluster in the beginning, and then we choose Euclidean distance to calculate the similarity between each cluster and merge these clusters successively. The threshold applies when forming the final flat clusters. The Euclidean distance is computed as follows [37]:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \quad (2)$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ are two points in Euclidean n-space.

*3.4. Attack Tree Creation Method.* After digging out potential groups, GroupTracer gathers all payloads and generates attack trees for each cluster to embody and better understand group behaviors. Algorithm 2 is to process all payloads from a cluster. $P_T$ represents all payloads collected from a given cluster $T$.

In the directed graph, nodes are command names and edges represent the command sequence between two commands. The out-degree of each command determines the size of each node. If a command exists only at the end of the payload, its size can also depend on the in-degree. The width of each edge is decided by the weight, which describes the frequency of occurrence of the command sequence. For instance, we have some payloads from a cluster, whose command sequences shown in Table 5 are obtained after command segmentation and abstraction. GroupTracer runs Algorithm 2 to generate the attack tree, as illustrated in Figure 5. Line 3–line 13 generate a list that stores two-step command sequences. For example, it turns cd chmod ./$t$ into [cd chmod, chmod ./$t$]. Line 14–line 16 counts the number of occurrences of all two-step command sequences.

## 4. Experiment and Metrics

*4.1. Dataset.* In this section, we first describe the datasets obtained from two kinds of IoT honeypots: UPnP-SOAP multiport honeypot and the Netis router backdoor honeypot. The time of data collection, the number of IPs, and the number of log entries are shown in Table 6.

*4.1.1. UPnP-SOAP Multiport Honeypot.* In the UPnP service, SOAP protocol assists in defining device types and other related information [38]. Therefore, there are a huge number of IoT devices that provide SOAP services, some of which do not require authentication. Honeypot simulates the behavior of the 11 ports most frequently scanned (e.g., 52881, 5500, and 2048) and the relevant SOAP service path and returns the corresponding information after being scanned. Six nodes are deployed on the Internet, averaging about 700 log entries per day. The dataset contains 153,413 log entries from 2,652 IPs over 196 days in 2019 (Table 6). Each log entry is identified by src_ip, timestamp, timezone, body, and query_string. The src_ip in our datasets is globally unique.

*4.1.2. Netis Router Backdoor Honeypot.* The Netis router listens on port 53413 (UDP) by default. After sending a specific string to it, the attacker can gain root login and then execute the corresponding command to perform a series of malicious behaviors. The honeypot has seven nodes deployed on the Internet, averaging about 300 log entries per day. Our Netis dataset contains 241,593 log entries from only 373 IPs over 279 days in 2019 (Table 6). Unlike UPnP-SOAP honeypot, the log entry in Netis is characterized by src_ip, timestamp, timezone, and data.

Given that both types of honeypots are simulated command execution vulnerabilities, we can leverage commands executed by attackers to discover their technology and tactics. Therefore, GroupTracer can be utilized to analyze the data of these two honeypots at the same time. Our datasets contain ten types of techniques that can be grouped into six tactics. These tactics are as follows [39]:

(i) Defense evasion: avoiding being detected while adversaries are intruding on the victim system.

(ii) Discovery: figuring out the environment of the victim system.

(iii) Lateral movement: moving through the victim environment. Adversaries might download their tools from remote servers to achieve lateral movement.

```
Require: T̂
Ensure: Basic Time-series features for all IPs
threshold = t_0
for all T_ip ∈ T̂ do
    sortByTime(T_ip)
    start = end = T_ip[0]
    for i ∈ [0, len(T_ip) − 1] do
        delta = T_ip[i + 1] − T_ip[i]
        if delta ≤ threshold then
            end = T_ip[i + 1]
            cnt = cnt + 1
        else
            unit = {start′: start, ′duration′: start − end, ′cnt′: cnt}
            time_interval.append(unit)
            start = T_ip[i + 1]
            end = start
            cnt = 1
        end if
    end for
    if start ≠ end then
        unit = {start′: start, ′duration′: start − end, ′cnt′: cnt}
        time_interval.append(unit)
    end if
    time_intervals.append(time_interval)
end for
```

Algorithm 1: Basic time-series feature generation for a given $T_{ip}$.

Table 4: Several significant time-series characteristics.

| # | Feature name | Description |
|---|---|---|
| 1 | maximum | The largest value of the time series |
| 2 | minimum | The smallest value of the time series |
| 3 | length | Number of attack periods per IP |
| 4 | mean | A measure of the central tendency |
| 5 | median | The "middle" value |
| 6 | standard_deviation | The square root of its variance |
| 7 | variance | The expectation of the squared deviation of a random variable from its mean |
| 8 | sum_value | Calculates the sum over the time-series values |

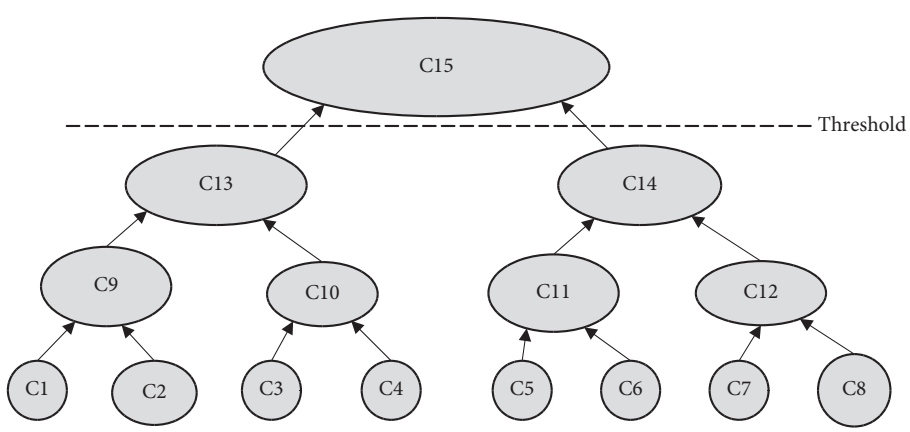



Figure 4: The hierarchy of the attacker behavioral clusters.

```
Require: Payloads P_T
Ensure: Attack tree for T
edges = []
weighted_edges = {}
for all payload ∈ P_T do
    payload.ast()
    tmp = payload.split()
    if len(tmp) > 2 then
        for i ∈ [0: len(tmp) − 1] do
            edges.append(tmp[i: i + 2])
        end for
    else
        edges.append(tmp)
    end if
end for
for all edge ∈ edges do
    weighted_edges[edge] = weighted_edges.get(edge, 0) + 1
end for
G = Digraph()
G.add_edges(weighted_edges)
DrawAttackTree(G, width = weighted_edge, node_size = G.degree())
```

Algorithm 2: Attack tree generation for a given cluster $T$.

Table 5: The occurrence frequency statistics of command sequence from a certain cluster.

| # | Command sequence | Frequency |
|---|---|---|
| 1 | cd wget | 5 |
| 2 | cd rm | 2 |
| 3 | cd chmod./t | 2 |
| 4 | cd chmod./s | 1 |
| 5 | cd chmod./nig | 1 |

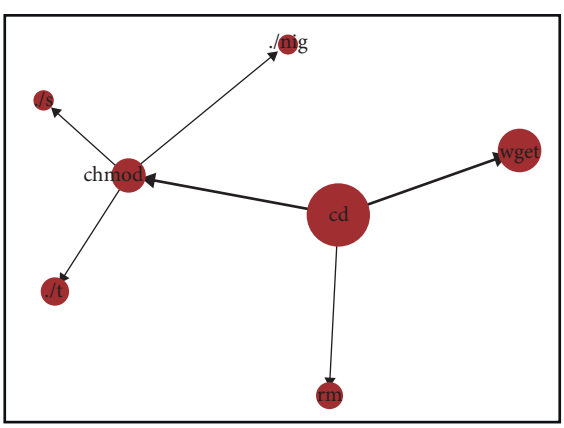



Figure 5: The attack tree for a given cluster.

(iv) Execution: running malicious code. The purpose of adversary-controlled code might be communicating with C&C servers or stealing data.

(v) Impact: expanding the impact of the intrusion on the victim system.

Table 6: Datasets from UPnP-SOAP and Netis backdoor honeypots.

| Dataset | Time | # of unique sources | # of entries |
|---|---|---|---|
| UPnP-SOAP | Apr.21–Nov.03 2019 | 2,652 | 153,413 |
| Netis backdoor | Mar.21–Dec.25 2019 | 373 | 241,593 |
| Total | Mar.21–Dec.24 2019 | 3,025 | 395,006 |

(vi) Collection: gathering information related to the adversary's objectives.

Table 7 shows techniques corresponding to each tactic and commands mapped to these techniques in our datasets.

*4.2. Clustering Performance Evaluation.* After clustering, we need measurement indicators to evaluate the effect. In general, the measurement for the quality of a clustering algorithm can be categorized into two kinds of criteria [40]: internal validation and external validation. External criteria are based on the previous knowledge about the data and require that ground truth labels are known. However, the labels of samples in this paper are not available. Thus, the internal validation is more suitable for our evaluation. More specifically, three internal indexes are utilized in this evaluation. These indexes measure if clusters are well compact and separated.

(i) Calinski–Harabasz (CH) [41]:

The index CH is defined as follows:

Table 7: Techniques corresponding to each tactic and commands mapped to these techniques in our datasets.

| Tactic | Technique | Command |
|---|---|---|
| Defense evasion | Disabling security tools | service iptables stop |
| | Clearing command history | history -c |
| | File deletion | rm |
| | File and directory permissions modification | chmod |
| Discovery | Process discovery | ps |
| | File and directory discovery | cd; ls; dir |
| Lateral movement | Remote File copy | tftp; wget; curl; ftpget |
| Execution | Exploitation for client Execution | sh;./mips;./zuki;./nig |
| Impact | Network denial of service | ultimate |
| Collection | Data from local system | echo; more; cat |

Table 8: The evaluation of GroupTracer and two comparison baselines using the Calinski–Harabasz, the Silhouette Coefficient, and the Davies–Bouldin index.

| Algorithm | Calinski–Harabasz | Silhouette Coefficient | Davies–Bouldin | # of clusters | Value |
|---|---|---|---|---|---|
| GroupTracer | 3416.9311 | 0.5389 | 0.7367 | 4 | 10 (threshold) |
| K-means model | 1212.7809 | 0.3246 | 1.1807 | 3 | 3 ($K$) |
| Meanshift model | 2199.9206 | 0.5004 | 0.6769 | 2 | 50% (quantile) |

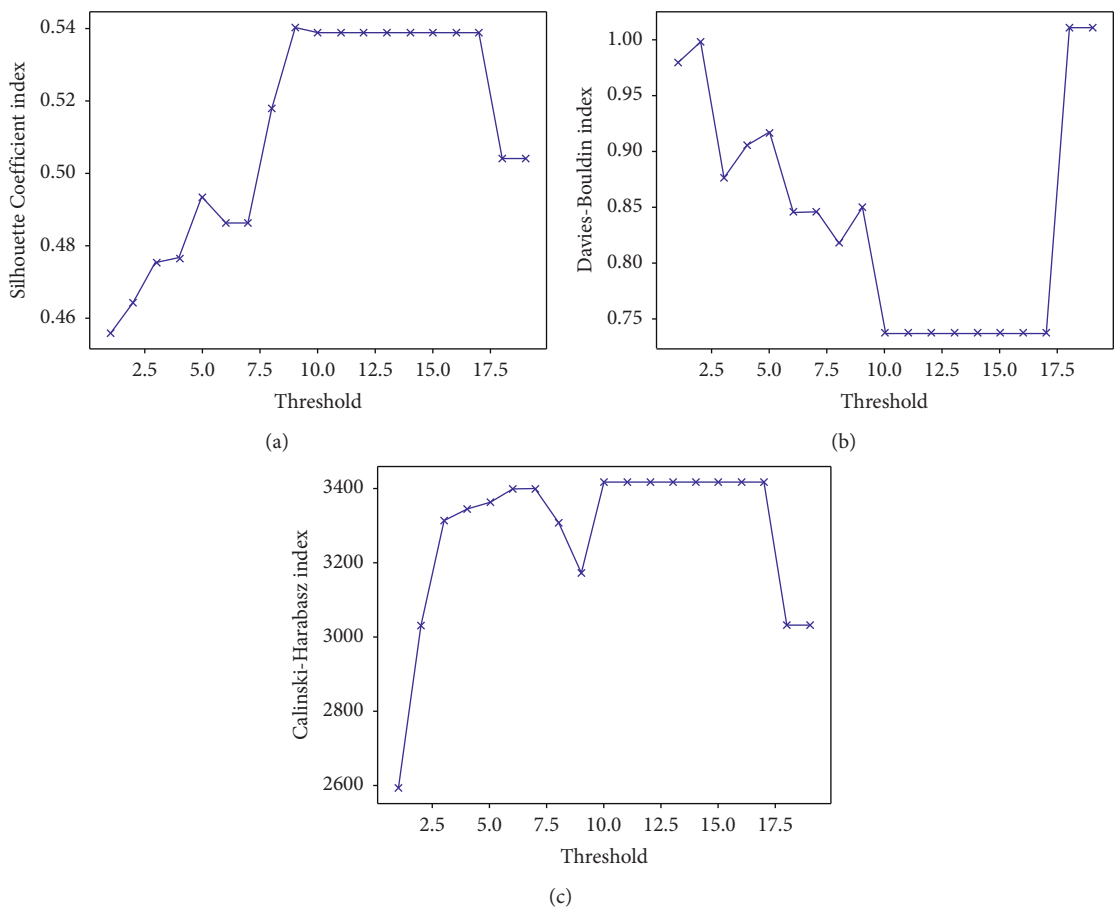



Figure 6: The evaluation of GroupTracer using the three indicators, where the value of threshold ranges from 1 to 20: (a) Silhouette Coefficient; (b) Davies–Bouldin; (c) Calinski–Harabasz.

$$\mathrm{CH} = \left(\frac{\mathrm{trace}(S_B)}{\mathrm{trace}(S_W)}\right) \cdot \left(\frac{n_p - 1}{n_p - k}\right), \quad (3)$$

where $S_B$ denotes the between-cluster scatter matrix and $S_W$ is the within-cluster scatter matrix. $n_p$ is the number of samples, and $k$ represents the number of

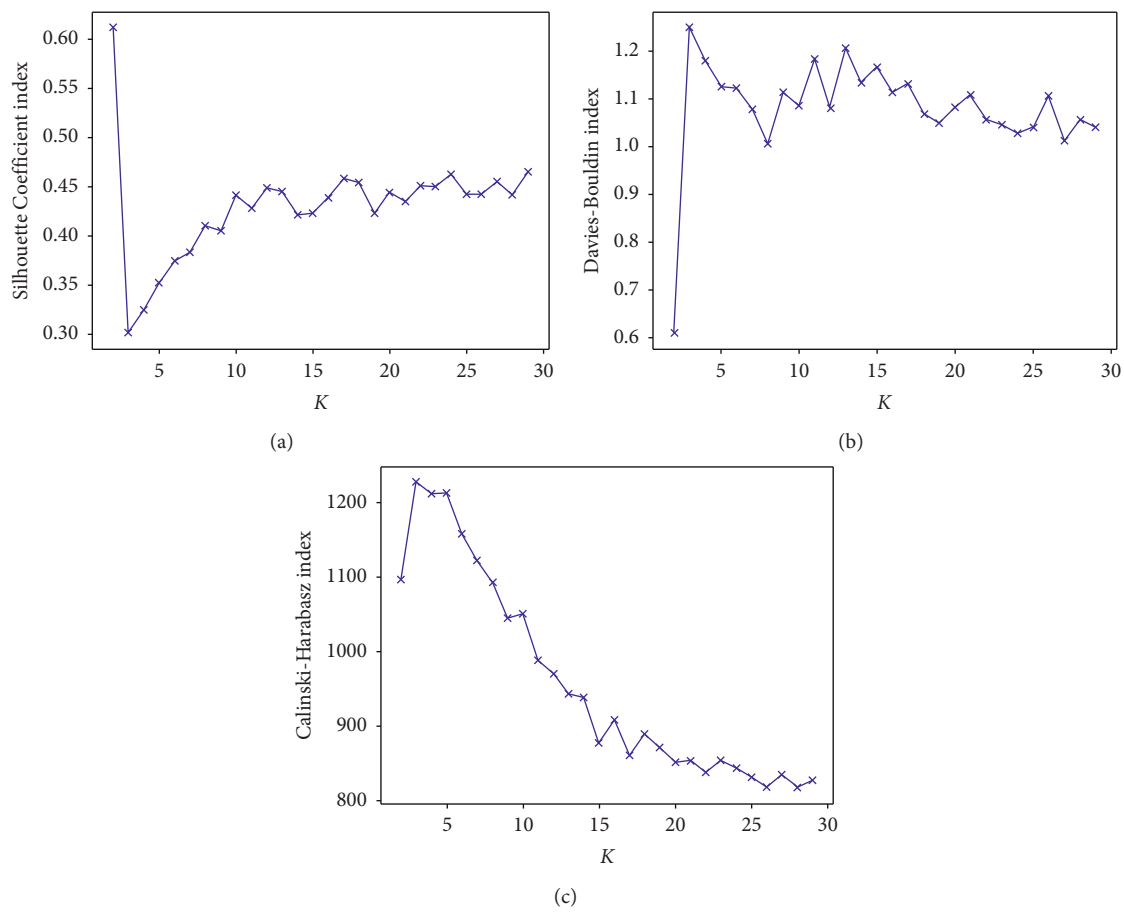



Figure 7: The evaluation of $K$-means model using the three metrics: (a) Silhouette Coefficient; (b) Davies–Bouldin; (c) Calinski–Harabasz.

classes. The larger value of CH indicates a better clustering solution.

(ii) Silhouette Coefficient [42]:

The Silhouette Coefficient $s$ is composed of two scores: $a$ means distance between a sample and the rest in the same cluster. $b$ is the distance between a point and all other samples included in the next nearest class. $s$ for all samples is given as the mean of the Silhouette Coefficient for each point. As for single sample $i$, $s$ can be computed by

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}. \quad (4)$$

The Silhouette Coefficient index ranges from −1 to 1; −1 represents a weak clustering effect, and 1 means a good classification effect. 0 indicates the overlap of clusters. A higher $s$ indicates a better clustering quality.

(iii) Davies–Bouldin (DB) [43]:

DB can be measured as follows:

$$\mathrm{DB} = \left(\frac{1}{k}\right) \sum_{i=1}^{k} \max_{i \neq j} \left\{ \frac{d(X_i) + d(X_j)}{d(k_i, k_j)} \right\}, \quad (5)$$

$k$ denotes the number of clusters. $i$, $j$ represent different cluster labels ($j \neq i$). $d(X_i)$ and $d(X_j)$ are the distance from all samples in cluster $i$ and $j$ to their respective cluster centroids. $d(k_i, k_j)$ is the distance between these centroids. A smaller DB value means a better clustering result.

*4.3. Experiment Design.* In the following, we evaluate GroupTracer by examining the cluster quality, i.e., how well clusters capture similar attack actors. The primary evaluation is for the group clustering of GroupTracer. We compare the evaluation results of GroupTracer based on the three indicators we mentioned above with the performance of group clustering based on the other two baseline algorithms to conclude that GroupTracer has excellent performance.

*4.3.1. Comparison Baselines.* Meanshift [44] and $K$-means [45] clustering algorithms are chosen to be the baselines. Meanshift is a centroid-based algorithm that requires an iterative step. It continuously calculates the expected moving distance of the center point and moves until the final condition is reached. For a given sample set, $K$-means divides it into $K$ clusters, minimizing a criterion (e.g., within-cluster sum-of-squares). $K$ is a positive integer number and must be predefined.

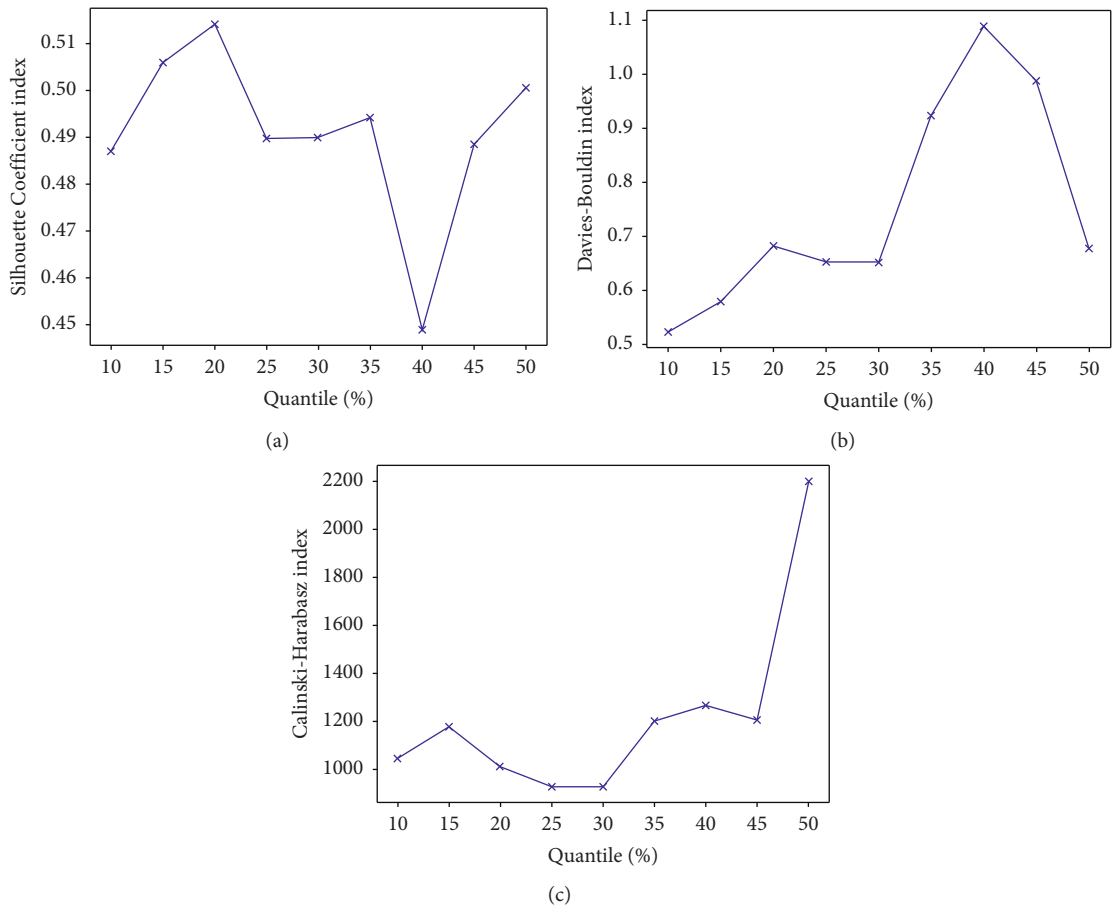



Figure 8: The evaluation of Meanshift model using the three metrics: (a) Silhouette Coefficient; (b) Davies–Bouldin; (c) Calinski–Harabasz.

We first extract the required features according to Section 3.2 and convert them into usable feature matrices. Before performing the final group clustering, we normalize the feature matrices. The reason is that clustering algorithms all use a distance measure to determine if object $i$ is more likely to belong to the same cluster as object $j$ than the same cluster as object $k$. These distance measures are affected by the scale of the variables. By putting all variables into the same range, we can weigh all variables equally, especially when the feature vectors are generated differently.

The main idea of normalization is to calculate the $p-$norm of each sample and then divide each element in the sample by the norm. The result of this process is that the $p-$norm of each processed sample is equal to 1. In the experiment, $p$ is set to 2 and $L2-$norm of vector $x = (x_1, \ldots, x_n)$ can be defined as [37]

$$\|x\|_2 = \left(|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2\right)^{1/2} = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}. \quad (6)$$

For the purpose of eliminating the influence of irrelevant variables as much as possible, we leverage all data in the dataset for experiments of each algorithm. For GroupTracer, we iterate the threshold value to pick the threshold with the highest clustering quality. The number of iterations is 20. Then we generate multiple versions of $K$-means clustering to attain the best clustering solution (Calinski–Harabasz index). Meantime, we run the Meanshift algorithm with the window size changing to get the best effect.

*4.4. Experiment Result and Discussion.* The evaluations of GroupTracer and two comparison baselines using the metrics we mentioned above are shown in Table 8. When running each algorithm, the value of different independent variables (e.g., threshold and quantile) brings the most significant change to the Calinski–Harabasz index, so this index is considered as the primary reference index when evaluating each algorithm. As a result, the Calinski–Harabasz index of GroupTracer reaches 3416.93 when the threshold is set to 10, which is about three times the K-means model and 1.5 times the Meanshift model. Taking the Silhouette Coefficient index as the definitive reference, the performance of GroupTracer is still the best. Although our algorithm performance is not the best after taking the Davies–Bouldin index into account, the gap is also within the acceptable range. GroupTracer generates 4 clusters, while the $K$-means model generates three classes. In the same way, the Meanshift model only generates 2 clusters for all data.

The evaluation of GroupTracer using the three metrics is illustrated in Figure 6, where the value of threshold ranges from 1 to 20. The Silhouette Coefficient index is on the rise

until the threshold reaches 9. When the threshold value ranges from 10 to 17, the index stabilizes at a relatively high level in Figure 6(a). The Davies–Bouldin index shows a downward trend as a whole until the threshold reaches ten and starts to stabilize at the lowest point (0.7367). After that, the index starts to rise rapidly in Figure 6(b). When the threshold value is 10–17, the Calinski–Harabasz index is maintained at the highest level (3416.93) in Figure 6(c), and the number of clusters is also kept at 4. In summary, all three indicators show that when the number of clusters is 4, the cluster quality becomes the highest and reaches a stable state.

When the CH index reaches the highest point, the Silhouette Coefficient and the DB index both have the worst effect in Figure 7. Similarly, when the CH index reaches the highest level, the DB index is in a lower position in Figure 8. It can be seen in these two figures that the changing trends of these indicators are not matched, which means that the two baselines are not well applied to our datasets.

## 5. Limitation and Future Work

Our research proposes a framework that can dig out potential groups behind active campaigns. This new technique can make full use of information from attack campaigns. However, our current design is preliminary. We only focus on the IPs used by the potential groups and do not go any further to track which specific groups were involved, that is, to try to correspond to the real groups [46]. Moreover, GroupTracer can only deal with honeypot logs that contain attack payloads.

In future work, we expect to combine NLP techniques with cyber threat intelligence to precisely match these potential groups to the real-world APT groups, in which the attack tree may be helpful to extract a group profile. Moreover, an experiment to prove the effectiveness of the attack tree should also be carried out. Further, we expect to apply more data sources such as system log and network traffic to expand our knowledge base and perform more comprehensive analysis.

## 6. Conclusion

In this work, we propose GroupTracer, a framework for attack actors clustering from IoT honeypot logs. GroupTracer is aimed at extracting the TTP profile automatically and digging out potential groups behind active campaigns. By mapping payloads to the ATT&CK framework, GroupTracer can effectively extract structured TTP profiles using two knowledge bases. Besides, this framework leverages four feature groups (namely, Time, TTPs, IP, and URL), a total of 18 characteristics, derived from log entries to capture the natural hierarchical structures for attacker groups. Finally, GroupTracer constructs attack trees for each cluster to embody the group actions. In the experiment, we compare our algorithm with two baseline algorithms. The evaluation of 395,006 log entries from 3,025 IPs reveals the high performance of GroupTracer, in which the Calinski–Harabasz index reaches 3416.93. Moreover, our proposed framework is generalizable as it is from a log accounting perspective, so its application is not limited to the IoT honeypot.

## Data Availability

The research 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 conflicts of interest.

## Acknowledgments

This work was supported in part by National Natural Science Foundation of China (61902265) and National Key Research and Development Program (2016YFE0206700 and 2018YFB0804503).

## References

[1] GReAT, Apt Trends Report, 2019, https://securelist.com/apt-trends-report-q2-2019/91897/.
[2] H. Scott, "Dyn," 2019, https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/.
[3] Talos, Vpnfilter, 2019, https://blog.talosintelligence.com/2018/05/VPNFilter.html.
[4] D. J. Bianco, "The pyramid of pain," 2019, https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html.
[5] J. Friedman and M. Bouchard, *Definitive Guide to Cyber Threat Intelligence: Using Knowledge about Adversaries to Win the War against Targeted Attacks*, Isightpartners, Amsterdam, Netherlands, 2015.
[6] MITRE, Darkhotel, 2019, https://attack.mitre.org/groups/G0012/.
[7] MITRE, 2019, Turla, https://attack.mitre.org/groups/G0010/.
[8] M. Alazab, "Profiling and classifying the behavior of malicious codes," *Journal of Systems and Software*, vol. 100, pp. 91–102, 2015.
[9] S. S. Hansen, T. M. T. Larsen, M. Stevanovic, and J. M. Pedersen, "An approach for detection and family classification of malware based on behavioral analysis," in *Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–5, IEEE, Kauai, HI, USA, February 2016.
[10] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pp. 183–194, ACM, New Orleans, LA, USA, March 2016.
[11] A. Makandar and A. Patrot, "Malware analysis and classification using artificial neural network," in *Proceedings of the 2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15)*, pp. 1–6, IEEE, Bangalore, India, December 2015.
[12] K. Borgolte, C. Kruegel, and G. Vigna, "Meerkat: detecting website defacements through image-based object recognition," in *Proceedings of the 24th {USENIX}Security Symposium ({USENIX}Security 15)*, pp. 595–610, Washington, DC, USA, August 2015.
[13] T. Taylor, X. Hu, T. Wang et al., "Detecting malicious exploit kits using tree-based similarity searches," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pp. 255–266, ACM, New Orleans, LA, USA, March 2016.

[14] Z. Tu, R. Li, Y. Li et al., "Your apps give you away: distinguishing mobile users by their app usage fingerprints," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–23, 2018.

[15] F. Zhang, S. Zhou, Z. Qin, and J. Liu, "Honeypot: a supplemented active defense system for network security," in *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 231–235, IEEE, Chengdu, China, August 2003.

[16] J. A. Jerkins, "Motivating a market or regulatory solution to iot insecurity with the mirai botnet code," in *Proceedings of the 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1–5, IEEE, Las Vegas, NV, USA, January 2017.

[17] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. R. Sadeghi, and S. Tarkoma, "Iot sentinel: automated device-type identification for security enforcement in iot," in *Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2177–2184, IEEE, Atlanta, GA, USA, June 2017.

[18] A. K. Simpson, F. Roesner, and T. Kohno, "Securing vulnerable home iot devices with an in-hub security manager," in *Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 551–556, IEEE, Seattle, WA, USA, March 2017.

[19] M. Wang, J. Santillan, and F. Kuipers, "Thingpot: an interactive internet-of-things honeypot," 2018, https://arxiv.org/abs/1807.04114.

[20] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: analysing the rise of iot compromises," in *Proceedings of the 9th {USENIX} Workshop on Offensive Technologies {WOOT}*, Washington, DC, USA, August 2015.

[21] S. Dowling, M. Schukat, and H. Melvin, "A zigbee honeypot to assess iot cyberattack behaviour," in *Proceedings of the 2017 28th Irish Signals and Systems Conference (ISSC)*, pp. 1–6, IEEE, Co Kerry, Ireland, June 2017.

[22] H. Heo and S. Shin, "Who is knocking on the telnet port: a large-scale empirical study of network scanning," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 625–636, ACM, Incheon, South Korea, June 2018.

[23] Gartner, The Definition of Cyber Threat Intelligence, 2019, https://www.gartner.com/en/documents/2487216.

[24] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: automatic and accurate extraction of threat actions from unstructured text of cti sources," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 103–115, ACM, Orlando, FL, USA, December 2017.

[25] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the International Conference on Machine Learning*, pp. 478–487, New York City, NY, USA, June 2016.

[26] P. Owezarski, "Unsupervised classification and characterization of honeypot attacks," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM) and Workshop*, pp. 10–18, IEEE, Rio de Janeiro, Brazil, November 2014.

[27] H. Cho, S. Lee, B. Kim, Y. Shin, and T. Lee, "The study of prediction of same attack group by comparing similarity of domain," in *Proceedings of the 2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1220–1222, IEEE, Jeju Island, South Korea, October 2015.

[28] V. Ghiëtte, H. Griffioen, and C. Doerr, "Fingerprinting tooling used for {SSH} compromisation attempts," in *Proceedings of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses {RAID} 2019*, pp. 61–71, Beijing, China, September 2019.

[29] MaxMind, Geolite, 2020, https://dev.maxmind.com/geoip/geoip2/geolite2/.

[30] I. Neamtiu, J. S. Foster, and M. Hicks, "Understanding source code evolution using abstract syntax tree matching," in *Proceedings of the 2005 International Workshop on Mining Software Repositories*, pp. 1–5, Saint Louis, MO, USA, May 2005.

[31] Ipipdotnet, Ipdb-Python, 2019, https://github.com/ipipdotnet/ipdb-python.

[32] Chronicle Security, Virustotal, 2019, https://www.virustotal.com/gui/home/url.

[33] RiskAnalytics, Malware Domain Blocklist, 2019, http://www.malwaredomains.com/.

[34] Dr.Web, Web Link Scanner, 2019, https://free.drweb.cn/.

[35] RTBAsia, Rtbasia Api, 2019, https://www.rtbasia.com/ip-lab.

[36] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[37] N. Dunford and J. T. Schwartz, *Linear Operators Part I: General Theory*, Vol. 243, Interscience Publishers, New York, NY, USA, 1958.

[38] K. S. Kim, C. Park, and J. Lee, "Internet home network electrical appliance control on the internet with the upnp expansion," in *Proceedings of the 2006 International Conference on Hybrid Information Technology*, pp. 629–634, IEEE, Jeju Island, South Korea, November 2006.

[39] ATT&CK Matrix, Tactic, 2020, https://attack.mitre.org/.

[40] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz, "Internal versus external cluster validation indexes," *International Journal of Computers and Communications*, vol. 5, pp. 27–34, 2011.

[41] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics—Theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[42] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[43] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[44] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 790–799, 1995.

[45] J. A. Hartigan and M. A. Wong, "Algorithm as 136: a $k$-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[46] MITRE, Apt Groups, 2020, https://attack.mitre.org/groups/.

Hindawi
Security and Communication Networks
Volume 2020, Article ID 6641844, 13 pages
https://doi.org/10.1155/2020/6641844

*Review Article*

# Research and Analysis of Electromagnetic Trojan Detection Based on Deep Learning

**Jiazhong Lu,[1] Xiaolei Liu,[2] Shibin Zhang,[1] and Yan Chang[1]**

[1]*School of Cybersecurity, Chengdu University of Information Technology, Chengdu 610225, Sichuan, China*
[2]*Institute of Computer Application, China Academy of Engineering Physics, Mianyang, Sichuan 621900, China*

Correspondence should be addressed to Xiaolei Liu; liuxiaolei@caep.cn

Received 22 October 2020; Revised 28 October 2020; Accepted 9 November 2020; Published 25 November 2020

Academic Editor: Ting Chen



The electromagnetic Trojan attack can break through the physical isolation to attack, and the leaked channel does not use the system network resources, which makes the traditional firewall and other intrusion detection devices unable to effectively prevent. Based on the existing research results, this paper proposes an electromagnetic Trojan detection method based on deep learning, which makes the work of electromagnetic Trojan analysis more intelligent. First, the electromagnetic wave signal is captured using software-defined radio technology, and then the signal is initially filtered in combination with a white list, a demodulated signal, and a rate of change in intensity. Secondly, the signal in the frequency domain is divided into blocks in a time-window mode, and the electromagnetic signals are represented by features such as time, information amount, and energy. Finally, the serialized signal feature vector is further extracted using the LSTM algorithm to identify the electromagnetic Trojan. This experiment uses the electromagnetic Trojan data published by Gurion University to test. And it can effectively defend electromagnetic Trojans, improve the participation of computers in electromagnetic Trojan detection, and reduce the cost of manual testing.

## 1. Introduction

With the development of information technology, electronic devices of various functions are continuously designed, such as computers and printers, which generate electromagnetic radiation during use. Electromagnetic radiation can also leak data in the device, threatening information security. Electromagnetic leakage is divided into two categories. The first type is passive leakage, which is an inevitable leakage caused by electronic equipment in normal work. Electromagnetic waves leaking from the display were captured by Hidema Tanaka [1]. The second category is active leakage, such as the experiment by Kuhn and Anderson [2], which conducts electromagnetic leakage in the form of actively transmitting specified information. This type of electronic leakage is malicious hardware or software in an electronic device that regularly leaks a specified signal to the outside world by controlling the electronic device. This type of electromagnetic leakage is called an electromagnetic Trojan [3]. Different from other common computer Trojan viruses, this type of electromagnetic Trojan does not use the system equipment such as the network to exchange information with the outside world, which causes the electromagnetic Trojan to break through physical isolation and be more difficult to detect. Such electromagnetic Trojans not only threaten the information security of ordinary users but may even threaten the physically isolated internal network.

At present, the main defense methods for electromagnetic Trojan attacks focus on passive defense, such as electromagnetic shielding and signal interference. The active detection methods mainly focus on monitoring the electromagnetic signals in the range, establishing a white list of normal signals and capturing and storing electromagnetic signals. The experienced security personnel observed the signal spectrum to find the electromagnetic Trojan attack, which is extremely inefficient [4]. Moreover, due to a large number of electromagnetic waves in the space, the amount of data captured per second can reach 1G to 2G [5]. Therefore, the classification efficiency by manual means is low. At the same time, the black/white list-based

electromagnetic wave data processing scheme cannot effectively detect the new electromagnetic Trojan attack.

Although the study of electromagnetic leakage has not stopped so far, there is no effective and active defense method. Our summary study found that the defense methods of electromagnetic Trojans can be divided into the following categories: (1) cutting off the transmission channel; (2) applying protection on the device; (3) electromagnetic wave record analysis. In summary, most of the defense methods for electromagnetic Trojans focus on the protection of electromagnetic wave channels, or the electromagnetic Trojans are defended by software-defined radio-related APIs.

We propose to monitor the electromagnetic wave signals in the monitoring area and characterize the signals. The flow calculation of big data is used to process electromagnetic wave data with huge data volume online. At the same time, we combine the fast classification function of deep learning to detect abnormal signals. Therefore, the protection of information security can greatly improve the detection efficiency of the electromagnetic Trojan signal, improve the security of the physical isolation network, and promote the development and progress of the electromagnetic Trojan detection.

## 2. Related Work

Xu et al. [6] used electromagnetic wave leakage to design a hardware Trojan for information acquisition of specific equipment. Their experiments show that the electromagnetic Trojan can silently obtain the 128 bit AES encryption key stored in the crypto chip.

Xu et al. [7] conducted a theoretical analysis of the electromagnetic leakage of the power line and established a radiation leakage model based on the power line. The electromagnetic leakage of the power line was verified by the measured data.

At the Black Hat Conference in August 2018, the Eurecom research team presented at the conference and demonstrated the results of using the electromagnetic leakage principle to attack mixed-signal wireless chips. By capturing the electromagnetic leakage of the chip, the key information of the encryption chip is obtained, so that the highly secure cryptographic algorithm is no longer safe.

In our previous work [8], considering the features of different power sources in different locations, combined with spark streaming technology and machine learning classification technology, a joint platform for electromagnetic signal anomaly detection based on big data analysis is proposed. The electromagnetic signal is abnormally detected by feature comparison and small-signal analysis, and the position and number between the signal sources are determined by three-point positioning and signal attenuation. The experimental results show that the method can detect abnormal electromagnetic signals and classify abnormal electromagnetic signals well, and the accuracy rate can reach 95%, and the positioning accuracy can reach 89%. However, our previous method can only detect abnormal electromagnetic signals, but it cannot detect electromagnetic Trojans.

Aiming at the above problems, this paper proposes an electromagnetic environment anomaly detection method based on deep learning. The electromagnetic Trojan signal is analyzed and modeled from the perspectives of time, frequency domain, and time domain. A large number of features are proposed to distinguish the normal signal from the electromagnetic Trojan signal, and the electromagnetic signal analysis system in the big data environment is designed. The system supports electromagnetic signal capture, filtering, characterization, and online/offline analysis in a big data environment. The system has a strong ability to expand through a hierarchical division of labor. Finally, the electromagnetic signal detection data set is constructed by using the electromagnetic Trojan disclosed by Ben Gurion University. The algorithm and design prototype system proposed in this paper are tested and evaluated.

### 2.1. Our Contribution

(1) This paper proposes the features of the new electromagnetic Trojan, which can analyze the signal in the frequency domain, time domain, and signal working mode and distinguish between abnormal signals and noise signals.

(2) This paper presents a new signal research perspective. In the traditional electromagnetic wave analysis technology, most of the research angles are the time domain and frequency domain information of the signal. In this paper, the working mode of the electromagnetic Trojan is compared and analyzed, and the sampled signal is compared to the flow data packet. A complete signal flow is analogous to the one-session mode. This method can improve the detection accuracy.

(3) We designed an electromagnetic signal analysis system that satisfies the big data environment. Through strict modular segmentation, the multidevice collector is supported in parallel to collect signals, filter data, and summarize and analyze the signals, so that the system has high ductility. At the same time, the monitoring bandwidth of the signal is increased through multidevice time slice rotation, various signal filtering schemes are designed, and the pressure of system data processing is reduced by signal filtering.

## 3. Electromagnetic Trojan Signal Detection Method

This paper proposes a joint platform for electromagnetic signal detection based on big data analysis. The electromagnetic Trojan in hardware is detected by the features of electromagnetic signals in time, information quantity, and energy.

*3.1. System Design.* This paper proposes a scheme for capturing electromagnetic waves using software-defined radio technology and proposes several new features for characterizing the electromagnetic wave signals. The method proposed

in this paper overcomes the determination of the existing scheme and combines the advantages of the existing scheme to perform electromagnetic Trojan detection. With the help of software-defined radio technology, the scope of electromagnetic wave monitoring is increased, and the problem of a large amount of data caused by increasing the signal acquisition range is overcome by proposing a large number of new features into the machine learning model for classification.

In the detection method of this chapter, the electromagnetic signal is first collected, the main acquisition range is 10 Hz–3000 MHz signal data, and the abnormality and background noise are distinguished; secondly, the electromagnetic signal is analyzed and filtered; the purpose is to screen out the meaningful signal, reduce storage and detection expenses, characterize the filtered signal from the perspective of energy, information volume, time, etc. in the time domain and the frequency domain, convert it into a time-dependent feature sequence depicting the state of the signal corresponding to the time state, and finally use LSTM. The method performs classification detection, selects a suspicious abnormal signal, and locates and alarms the signal according to the intensity information of the signal.

Our approach procedures are shown in Figure 1, which contains the key stages and necessary operations of the Trojan signal detection method.

The first stage is the electromagnetic signal capture phase, and the signal capturer and signal analysis plug-in are connected to the laboratory's server. In this way, the server can directly control the arrester for real-time signal capture and analyze the electromagnetic signal to find out the signal data being transmitted within the monitored frequency range. This automated work is a huge improvement in detection efficiency.

The second stage is the electromagnetic signal filtering stage. For large data electromagnetic signals, the signal must be filtered to save detection cost and shorten the detection time. Therefore, most of the normal, white, background, and noise signals need to be excluded. Three filtering methods are used in this method to effectively do this.

The third stage is the feature extraction stage. The filtered signal is divided into blocks in the form of the time window, and the filtered electromagnetic wave signals are extracted from the angles of energy, time, and information entropy, and a serialized electromagnetic wave feature matrix is generated.

The fourth stage is anomaly detection, and there is no effective characterization of the working mode and periodic features of the signal in the serialized signal feature matrix. Therefore, the LSTM algorithm is used to further mine the time regularity of the signal and feature extraction. Then, softmax is used to classify the features of LSTM mining and detect abnormal signals.

*3.2. Electromagnetic Signal Processing.* Signal processing includes signal acquisition and signal filtering.

*3.2.1. Signal Acquisition.* This paper uses the software-defined radio (SDR) technology to capture radio signals in space. This technology can easily capture, analyze, and

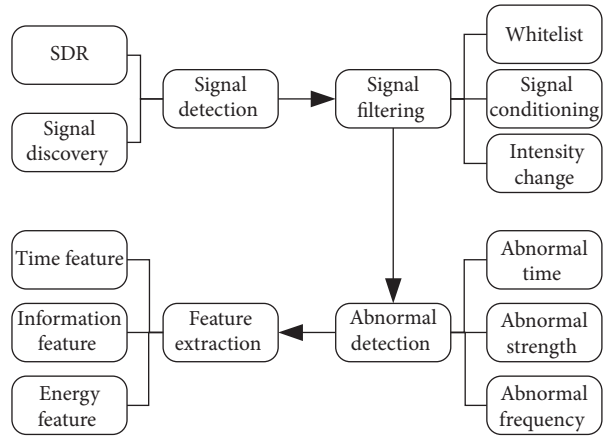



Figure 1: Approach procedures.

process signals in each frequency band in the current location. Software-defined ratios can acquire electromagnetic wave signals within a specified range by dynamically configuring the operating frequency band and sampling range. Such a collection method does not need to modify the hardware device and can conveniently collect the signals in the monitoring range by using the time slice rotation mode.

Since the modulation modes of the signals in the respective frequency bands are different, the specific bearer content of the signal cannot be obtained in the data acquisition phase. Therefore, the signals we acquire at this stage are complex signal data in the frequency domain, and then these signals are further processed according to the frequency band and modulation protocol.

Unlike other signal receiving devices, the normal signal acceptance process requires the sender and the receiver to agree on the transmission time, the transmission band, and the coding mode. In our signal acquisition process, we need to monitor the electromagnetic signals in the range, which makes it impossible to specify the working frequency range of the signal. Therefore, we need one more signal discovery process than other signal receiving devices in the working process. This paper proposes a signal discovery method based on the signal processing scheme in random signal analysis. The source of the background noise signal is mainly the low-energy signal after the band-pass filtering of the signal outside the sampling range. Therefore, although the background signal is indeterminate in intensity, the intensity value follows a uniform distribution in the distribution, the intensity mean difference is small, and the intensity variance is also smaller; the signaled channel and the unsignaled edge zone will suddenly increase the intensity of the sampling point, so the intensity mean and intensity variance will increase compared to the background signal; the sampling point is the signaled channel. When the intensity of the sampling point reaches the maximum value, the intensity variance will be larger. In this paper, by sampling the number of sampling points, the sampling points with signals are searched in a rolling manner, and the corresponding information is sent to the signal filtering module for filtering, which reduces the load caused by the useless data on the

system. So, in addition to capturing the signal, the module needs to pick the true signal from the background noise. The collection procedures are as shown in Figure 2. Set the center frequency $f$ and the sampling rate $s$ for each device and then control the device for sampling. The variance and the mean are calculated every $k$ samples. If the mean is large, the sample point is retained; otherwise, the sample point variance is calculated. If the variance is greater than the defined variance threshold, discard these points; otherwise, it is considered that there is a signal at these sampling points to save the data.

Figure 3(a) shows the signal power distribution of the corresponding frequency range. The ordinate unit is dBm. Figure 3(b) is the variance distribution of the intensity of the corresponding sampling point. In Figure 3, we sample 4096 units and perform variance calculation. We can see that there are two obvious signals in Figure 3(a). In the corresponding position in Figure 3(b), you can see two peaks with large variances (the first half of Figure 3(b)). Some data have no obvious variance fluctuations, and the reason why the data in the first half of Figure 3(a) are significantly different is that the power and intensity are logarithmically processed during the conversion, so the small difference between the intensities is logarithmic. The process is enlarged, so the variance changes during the calculation process are not obvious. The signal can be effectively found by the variance of the power of the sampling points in the corresponding frequency range [9].

(1) Whitelist Signal: the whitelist mechanism is added, the signal frequency is excluded in the whitelist, and the whitelist library supports user customization, such as broadcast signals and special frequency signals. The whitelist is shown in Table 1.

(2) It can mediate the signal: this type of signal can be filtered by signal protocol unpacking. For example, the FM broadcast signal is unpacked by the FM protocol to filter out the FM signal, as shown in Table 2.

(3) Intensity change signal: the filtering scheme adopts multiple collection points to collect at the same time, then calculates the intensity of the collected signal and filters according to the intensity of the collected signal, and filters out the signal transmission source are not within the system scope.

*3.2.2. Signal Filtering.* Before data analysis, the data need to be preprocessed; the purpose is to delete duplicate information and redundant information and reduce the data processing load of the system under the premise of preserving abnormal data. In the electromagnetic Trojan signal analysis platform of this paper, three filtering modes are adopted:

In filtering electromagnetic signals, some fixed frequencies may be used inside the system as a frequency band for secure communication or a frequency band not suitable for monitoring. Therefore, for these frequency bands, we have established a whitelist list as shown in Table 1. First, we

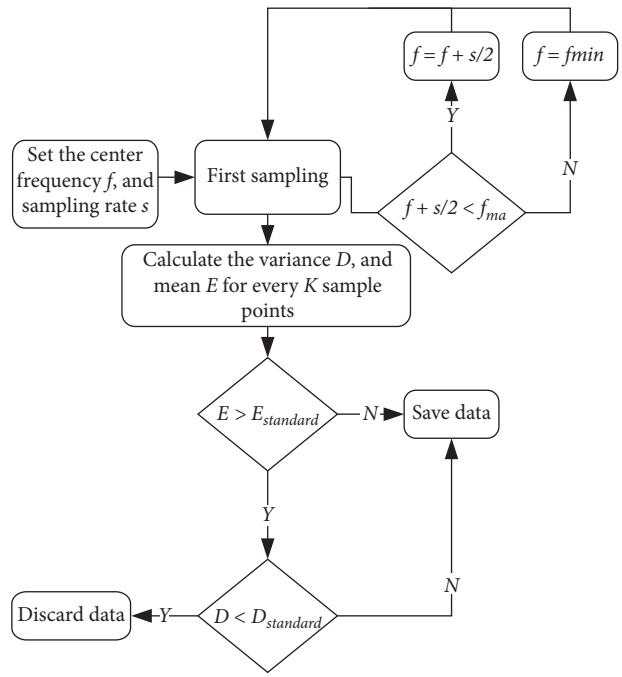



Figure 2: Collection procedures.

use whitelist signal filtering to compare the input signal frequency with the frequency in the whitelist and directly filter it into a normal signal. If it is not in the turn, it can be used to mediate signal filtering.

Among the received electromagnetic signals, there are a large number of electromagnetic signals that are normal electromagnetic signals conforming to the specifications, such as broadcast signals and mobile communication signals. Therefore, a demodulate signal filtering scheme is added after the whitelist signal. In the mediation signal filtering, the signal demodulation plug-in is used to demodulate and filter the electromagnetic signal and combined with the distribution of the signal frequency band in the analysis region, the signal that can be normally demodulated and conforms to the specification is used as a normal signal, and the protocol cannot be parsed or not. The signal of the local signal list is output to the intensity change signal for filtering. The demodulate electromagnetic signal protocol in the partially monitored frequency band is shown in Table 2.

In the intensity change signal, the greater the intensity change value, the closer the signal source is to the collection point, otherwise, the farther the signal source is from the collection point. The following relationship exists between the wireless signal transmission power $PR$ and the received power PT:

$$\mathrm{PR} = \frac{\mathrm{PT}}{(r * n)}, \tag{1}$$

where $r$ is the distance between the transmitting point and the receiving point and $n$ is the environmental constant. From this, we can deduct the relationship between the fixed point of the launching point and the fixed distance of the two receiving points and the distance between the transmitting point and the intensity:

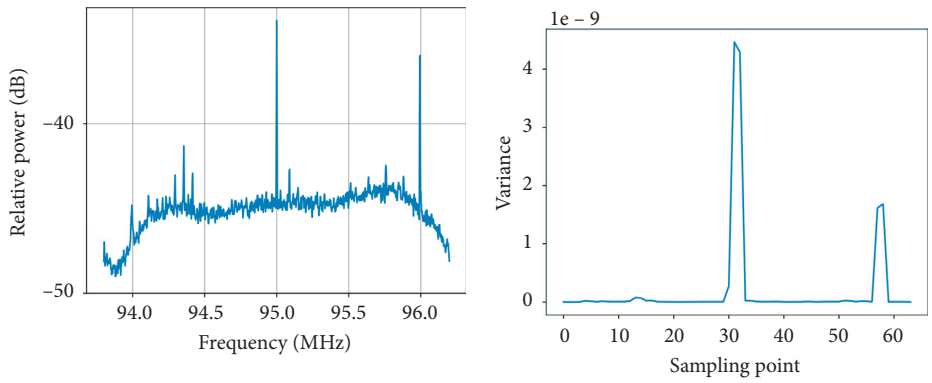



Figure 3: Performance of the frequency domain signal on the variance. (a) Signal frequency domain diagram from 94 to 96 MHz; (b) Sample point variance distribution from 94 to 96 MHz.

Table 1: Partial whitelist signals.

| Frequency band (MHz) | |
|---|---|
| 1.7 | 50 |
| 1.9 | 144.100 |
| 3.5/3.8 | 144.200 |
| 10.7 | 1000 |
| 2.4 | 2412 |

Table 2: 10 Hz – 3000 MHz partial signal protocol.

| Frequency band (MHz) | Protocol |
|---|---|
| 1710–1725<br>885–909 | Mobile 2G signal protocol |
| 2010–2025 | Mobile 3G signal protocol |
| 1880–1890<br>2320–2370<br>2575–2635 | Mobile 4G signal protocol |
| 909–915<br>1745–1755 | Unicom 2G signal protocol |
| 1940–1955 | Unicom 3G signal protocol |
| 2300–2320<br>2555–2575<br>1755–1765 | Unicom 4G signal protocol |
| 825–840 | Telecom 2G signal protocol |
| 1920–1935 | Telecom 3G signal protocol |

$$\frac{1}{r} = \frac{(\mathrm{PT}_1/\mathrm{PT}_2 - 1)}{x_{12}}, \tag{2}$$

where $r$ is the distance from the signal transmission point to the receiving point 2, $\mathrm{PT}_1$ is the power received at the receiving point 1, $\mathrm{PT}_2$ is the power received at the receiving point 2, and $x_{12}$ is the distance between the two receiving points. The ratio between the distance and the signal strength can be obtained from the distance between the two points, as shown in Figure 4. The signal strength of the sample point is inversely proportional to the distance. The greater the signal strength, the closer the distance.

*3.3. Electromagnetic Trojan Signal Feature Extraction.* Based on the working mode of electromagnetic Trojan, combined with the experience of security schemes in APT traffic detection [10], this paper quantifies the electromagnetic waves in terms of time angle and information transmission. During the characterization of electromagnetic waves, our classification is based on the center frequency of the channel occupied by the signal, and then the time-window division method is used to perform dicing extraction on the signal. This feature extraction method can greatly compress our data volume while keeping the behavior of the signal as much as possible. Because of the limitation of electromagnetic leakage in one-directional propagation, electromagnetic Trojans usually use channels that are not commonly used or used at a lower frequency in the communication process. Therefore, it is a good choice to distinguish the center frequency of the signal.

*3.3.1. Time Feature.* Electromagnetic Trojans work by using electromagnetic radiation for information leakage, but software-defined radio technology does not allow the use of electromagnetic waves from the circuit for the reception. Therefore, the working mode of the electromagnetic Trojan can only be a one-way transmission and a connectionless working mode. Through the study of the electromagnetic Trojan, we found that, to work properly in this mode, the electromagnetic Trojan will take a certain heartbeat mechanism to determine whether the Trojan host is active. And before sending important information, it will send a signal to synchronize. Therefore, the electromagnetic Trojan signal has a strong feature in time. So, in time, we extracted the features as shown in Table 3.

Interval features include minimum interval, maximum interval, average interval, and interval variance. The electromagnetic signal sent by the electromagnetic Trojan attacker is automatically sent with the programming. The transmission of most electromagnetic Trojan signals has a

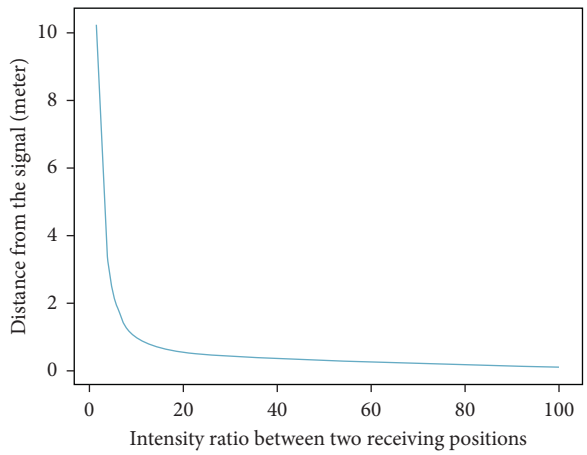



Figure 4: Relationship between two receiving position intensity ratios and signal source distance.

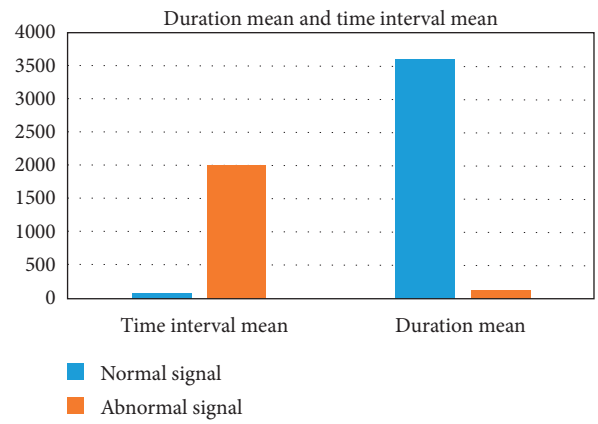



Figure 5: The difference between the normal signal and the abnormal signal in time.

time interval. The average interval is mostly within a few seconds. We analyze nearly 2000 signals of 6 kinds of electromagnetic Trojans. It was found that less than 10% of the signals had an average transmission time of more than 10 seconds. In the normal signal, this average interval is shorter than that of the Trojan signal, and the transmission time interval of most normal signals is relatively small. For example, the WIFI signal appears as a normal distribution in the time interval, and the minimum interval and the maximum interval are not large, so the signal variance values are small; the Mobile 2G broadcast signal has a heartbeat feature in time, showing a certain regularity, so the signal interval variance used by the channel is small; while the Trojan signal has obvious periodicity and lacks synchronization mechanism, it shows strong regularity in time interval, but the use rate of the electromagnetic Trojan is lower. Therefore, the variance value is generally larger. Therefore, the features of the time interval can be used to distinguish the Trojan signal from the normal signal, as shown in Figure 5.

*3.3.2. Information Volume Features.* We extracted the information volume features of the signal as shown in Table 4. Compared with the features of the normal signal on the channel, the Trojan signal also has obvious differences. Since the channel for normal signal transmission is well-divided, there are multiple devices sharing the channel, so the volume of information on the channel is generally stable and the duty ratio of the channel is high.

We have found that the electromagnetic Trojan cannot accurately control the power during the process of transmitting signals. Compared with the normal signal transmitting equipment, there will be large fluctuations in signal power, occupied channel bandwidth and strength. We block the time window and channel of the signal and treat each block as a packet in the network transmission. All the packets in the window are studied as one session. Each block extracts power data separately and finds the maximum power block, the minimum power block, and the variance between all power blocks in one session. At the same time,

Table 3: Time-based electromagnetic wave features.

| Features | Description |
|---|---|
| Minimum interval | Minimum time for signal interval |
| Maximum interval | Maximum time of signal interval |
| Average interval | Average time of signal interval |
| Interval variance | Variance of signal interval |
| Maximum duration | Minimum time for uninterrupted signal |
| Maximum duration | Maximum time for uninterrupted signal |
| Duration variance | Variance of the uninterrupted signal |
| Average duration | Average time of the uninterrupted signal |

the background power in the current environment and the parameter conditions is extracted.

In the working process of the electromagnetic Trojan, to be able to transmit the secret data, the electromagnetic Trojan generally selects the channel with more idle time; otherwise, the normal device communication will be affected during the transmission process or the interference may cause the stealing signal to be unsuccessful. We have to send it out. When a normal device communicates, as long as it meets the communication specifications of the current frequency, it will have its blocking error correction system, so we propose the channel duty cycle (the ratio of the total duration of the signal to the total time of the block). There is a clear difference between the electromagnetic Trojan signal and the normal signal at the channel duty cycle.

The bandwidth refers to the frequency range occupied by the signal. In this paper, the bandwidth decision scheme is the first zero-point method [9]. When the signal strength drops to 0 for the first time, the frequency range is the bandwidth. According to Shannon's theorem, it can be known that, in the noise-and signal-independent Gaussian white noise channels, assuming the signal power S and the noise power N are constant, the relationship between the channel capacity $C$ (b/s) and the channel bandwidth W (Hz) [11] are as shown in equation (3). It can be seen that the channel capacity is proportional to the bandwidth. Therefore, by taking the value of the bandwidth used in the actual transmission of the signal, it can represent the amount of information transmitted by the current signal:

Table 4: Information volume feature.

| Features | Description |
|---|---|
| The number of blocks | The number of blocks that the signal is divided into at a time |
| Average total power per block | Average power per block |
| Single block maximum power | Maximum power per signal |
| Single block minimum power | Minimum power per signal |
| Block power variance | The amount of noise interference in the environment where the signal is located |
| Minimum duty cycle | The minimum amount of data for a single transmission of a signal |
| Average bandwidth | The bandwidth of each frame of the current signal divided by the number of frames |
| Bandwidth variance | Bandwidth change rate, measuring the stability of data transmission |
| Maximum bandwidth | The maximum bandwidth of the signal to be sent, used to calculate the channel width |
| Minimum bandwidth | The minimum amount of data sent by a single signal |

$$C = W \log_2\left(1 + \frac{S}{N}\right). \tag{3}$$

*3.3.3. Energy Features.* The collected signals are stored in the form of complex signals, so we define the energy according to the definition of the complex signal. The energy is defined as shown in the following equation:

$$E = \sqrt{I^2 + Q^2}. \tag{4}$$

There is a close relationship between the magnitude of the energy $E$ and the strength of the transmitted signal and the amount of information transmitted.

As shown in Figure 6, the collected complex signals are plotted in a complex coordinate system. Figure 6(a) shows the step-by-step situation of the sampling point when the channel is idle. Figure 6(b) shows the distribution of sampling points when the distance is constant and there is a small amount of data for transmission. Figure 6(c) shows the same distance. The distribution of sample points sends large amounts of data.

According to the definition of energy in equation (4), the visual display of energy in the graph is the distance from the sampling point to the off-center. As can be seen from Figure 6, as the amount of data transmitted by the channel increases, the energy distribution on the channel also appears. The obvious difference is that energy can be used to distinguish the amount of data transmitted on the current channel.

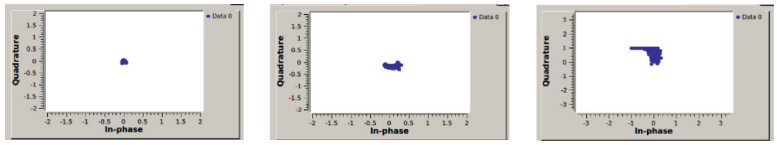
So, we can find the relationship between energy and the amount of transmission at a certain distance. Since the normal signal and the electromagnetic Trojan signal have significant differences in the usage rule and the amount of transmitted information, the stability of signal transmission is poor. Therefore, from the perspective of energy, we propose the features of maximum energy, minimum energy, energy mean, maximum energy change rate, and minimum energy change rate.

*3.3.4. Harmonic Features.* Harmonic signal features are the most prominent signal features of electromagnetic Trojans. Harmonic is a signal component that appears to be an integer multiple of the fundamental frequency when the signal is converted from the time domain to the frequency domain by the Fourier transform. The frequency is several times that of the fundamental wave. The generation of harmonics affects the normal transmission of the signal, which is a noise during data transmission and affects the normal transmission of signals on the surrounding channels. Therefore, the suppression of harmonics is an extremely important part in the signal transmission structure. Since the spectrum shifts before the signal is transmitted, the signal is converted from the low frequency to the high frequency, so the multiple of the frequency may change, but the frequency interval between the harmonics cannot be changed.

The electromagnetic Trojan's transmitting equipment is generally used for displays and wires. Compared with the dedicated signal transmitting equipment, in addition to the relevant features discussed above, there is also a problem with the depth of the harmonic signal. As shown in Figure 7(a), the abscissa is the frequency of the signal, the vertical axis is time, and the depth of the color represents the signal strength here (where the deeper the color, the stronger the intensity, and the stronger the signal is). Figure 7(b) shows the normal signal. It can be seen that the signaled part is a black line and no harmonics are found. To prevent harmonics from causing noise effects, harmonic suppression has been used before the signal is transmitted to minimize the influence of harmonics, so this is a big difference between the electromagnetic Trojan and the normal signal.

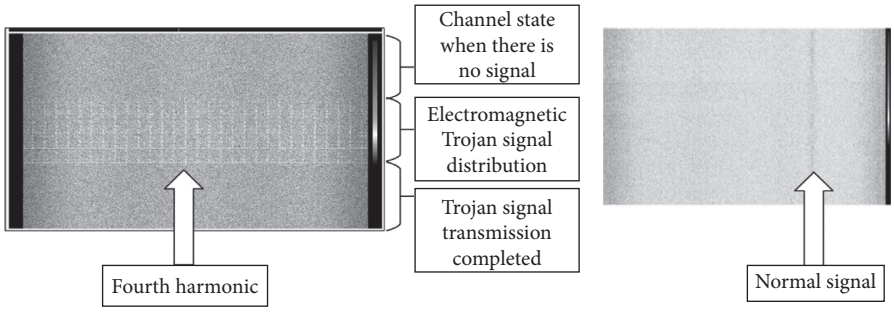
## 4. Electromagnetic Trojan Detection and Signal Classification

*4.1. Experimental Environment.* The acquisition equipment of this experiment is TV stick RTL2832Ux3, HACKrfx1. The acquisition range is TV bar RTL2832U (10 Hz to 1800 MHz, maximum sampling rate 3.2 MHz) and HACKrf (10 Hz to 6 GHz, maximum sampling rate 128 MHz). Rolling monitoring of electromagnetic signals from 10 Hz to 6 GHz [11] is possible by HACKrf one + sdr (software-defined radio). The data compression sensing algorithm and serialized memory are used to compress the data, and the compression effect is greater than 0.5. The hardware environment configuration of the experiment is shown in Table 5.




(a) (b) (c)

Figure 6: Relationship between energy size and data transfer volume. (a) Energy map when the channel is idle; energy maps for (b) a small amount of data transmission and (c) a large amount of data transmission.




(a) (b)

Figure 7: Signal waterfall diagram. (a) Electromagnetic Trojan harmonic signal waterfall map; (b) normal signal waterfall map.

*4.2. Experimental Data Set.* The GSMem system, which was released in 2015 by Mordechai Guri [12] of Ben Gurion University, was used in this experiment. The system can signal the electromagnetic leakage of the computer CPU. In the transmission process, it only needs 4 KB of CPU cache space, no root or administrator privileges, no need to call the system API, and intel or AMD CPU signal transmission can be made on Windows or Linux system, and for the signal acceptance, only a mobile phone supporting the GSM communication protocol is required. Figure 8 shows a screenshot of a CPU Trojan provided by Mordechai Guri. By running the CPU Trojan on the target computer, you can receive the information to be compromised on the phone next to it.

The system-bus-radio system provided by William Entriken was also used in this experiment. The system is written in C language, and an electromagnetic Trojan is installed on the display to transmit the data to be transmitted in the form of electromagnetic waves.

*4.3. Data Collection.* The experimental data are divided into two parts. One part is the normal electromagnetic signal data, the signal comes from the fixed collection point in the laboratory, the frequency is from 10 MHz to 3000 MHz, and the acquisition time is one week, marked as normal signals. Another part of the Trojan data is generated using the public system. The sampling method, acquisition time, and acquisition point are consistent with the normal data and marked as abnormal signals. In the data acquisition process, we use signal filtering and signal extraction to characterize the data, which greatly reduces the amount of data processing. The data set is shown in Table 6.

Table 5: Hardware environment configuration.

| Project | Configuration |
|---|---|
| Server | Dell PowerEdge R730XD |
| Processor | 2 Intel® Xeon® E5 – 2630 v3, 8 cores 16 threads |
| RAM | 8 * 16 GB RDIMMs |
| SDR | HACKrf one, RTL2832U *R*820 T |
| Hard disk | 2 TB * 10 |
| Collector | HACKrf one * 1, RTL2832U * 3 |

*4.4. Experimental Indicators.* In this experiment, the following indicators were selected for evaluation: FN, FP, TN, TP, FPR, FNR, TPR, TNR, precision, recall, and F1-score. False negative (FN) refers to the number of samples that are determined to be electromagnetic Trojan signals but are normal signals. False positive (FP) is judged to be a normal signal but is the number of samples of the electromagnetic Trojan signal. True negative (TN) is that the electromagnetic Trojan signal is correctly determined as the number of samples of the electromagnetic Trojan signal. True positive (TP) is that the normal signal is correctly determined as the number of samples of the normal signal.

FP rate (FPR) is the ratio of the number of electromagnetic Trojan signal samples predicted to be normal signals to the actual number of negative samples, the ratio of FP to FP + TN. FN rate (FNR), which is predicted as the ratio of the number of normal signals of the electromagnetic Trojan signal to the actual number of normal signals, that is, the ratio of FN to FN + TP. TP rate (TPR) is the ratio of the number of normal signals correctly predicted to the actual number of normal samples, the ratio of TP to FN + TP. TN rate (TNR) is correctly predicted as the ratio of the number of electromagnetic Trojan signals to the actual number of

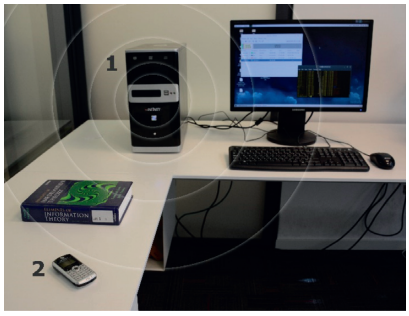



Figure 8: Display electromagnetic trojan.

Table 6: Experimental data set.

| Data set | Feature size (GB) |
|---|---|
| Normal signal | 531 |
| CPU signal | 52 |
| Display signal | 34 |

electromagnetic Trojans, the ratio of TN to FP + TN. The precision accuracy value is the ratio of the number of correctly classified samples to the total number of samples. The recall is the ratio of the number of samples correctly classified by the classifier to the actual number of positive samples. F1-scoreis the weighted harmonic average of accuracy and recall that is used to measure the effectiveness of the test method. The specific formula is as follows:

$$\begin{aligned} A\,(\text{accuracy}) &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\ P\,(\text{precision}) &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ R\,(\text{recall}) &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ F1\ \text{score} &= \frac{2 * P * R}{P + R}. \end{aligned} \tag{5}$$

## 5. Anomaly Detection and Comparative Experiment

We used two evaluation methods to evaluate the three deep learning algorithms of LSTM, RNN, and CNN, which are the cross method and percentage segmentation method. LSTM is the classification algorithm used in this article, and RNN is the method used in the latest hardware Trojan detection patent in 2018. The electromagnetic waves are characterized according to the features extracted in the patent, and then the electromagnetic signals are classified using the RNN algorithm used in the patent.

In the experiment, the collected and filtered signals contain a large number of noise signals and normal signals in addition to the Trojan signal. Therefore, we divide the classification model into three parts: the input layer, the LSTM layer, and the classifier. The input layer is responsible for converting the characterized vector into a feature sequence for input to the LSTM layer for analysis. The LSTM layer receives the serialized signal data input from the input layer, selectively memorizes and learns the current input data, and then retains the important information and generates a 128-dimensional feature vector for characterizing the current model state. The classifier layer accepts the 128-dimensional vector of the LSTM layer for electromagnetic Trojan signal recognition.

The cross method uses randomization of data into 10 equal parts, 9 of which are used as training sets, and the remaining one is used as a test set for accuracy and accuracy. The detection process needs to be repeated 10 times. The advantage of this is that the data are completely randomized and scrambled, resulting in higher accuracy and lower false positives, eliminating the problem of false negatives caused by duplicate data.

In this experiment, the three deep learning algorithms of LSTM, CNN, and RNN are verified by the cross method. The experimental results are shown in Figure 9. It can be seen from the figure that the accuracy rate of LSTM is 96.41%, the accuracy rate is 98.65%, the performance of RNN is relatively poor, the accuracy rate is 93.51%, and the accuracy rate is 95.56%; the performance of CNN algorithm is even worse, the accuracy rate only 85.62%, and the accuracy rate is only 91.75%. The accuracy in this experiment is higher than that of other algorithms. The features extracted by this experiment are better for the recognition of electromagnetic Trojan signals. The false-negative rate of the system is lower, but the false alarm rate is slightly higher. The reason may be that the features extracted in this paper are not characterized by time series, and the regularity of time series is the main difference between electromagnetic Trojan signal and normal signal, so the performance of circulating neural network is better. For feature classification with a long duration, LSTM performs much better than RNN, so LSTM has higher accuracy for electromagnetic Trojan recognition.

The percentage division method divides the data into two parts, one for 75% and the other for 25%, with 75% of the data for training and the rest for testing, as shown in Figure 10. The advantage is that the algorithm detection operation is not complicated, saving time and high efficiency. However, its accuracy and false-negative rate are slightly inferior to those of the cross, which is suitable for classification with a small amount of data and obvious features. In this experiment, because the electromagnetic signal data is very large and has many features, the overall test results are not satisfactory, but it can be seen that LSTM has higher accuracy than CNN and RNN and CNN has the worst performance; the reason may be, in the detection process, the detection system mainly focuses on the working mode of the electromagnetic Trojan signal. The discrimination between the electromagnetic Trojan signal and the normal signal is mainly in the period, so the deep neural network has a better performance. However, the accuracy and accuracy are consistent with those in the cross-certification, indicating that the system's false-negative rate is still low and the false positive rate needs to be improved.

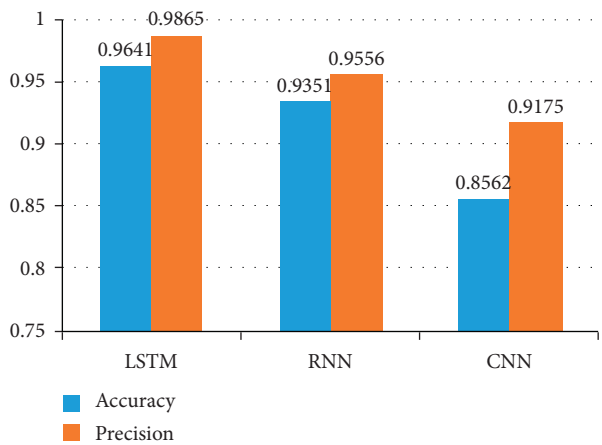



Figure 9: 10-fold cross-validation results.

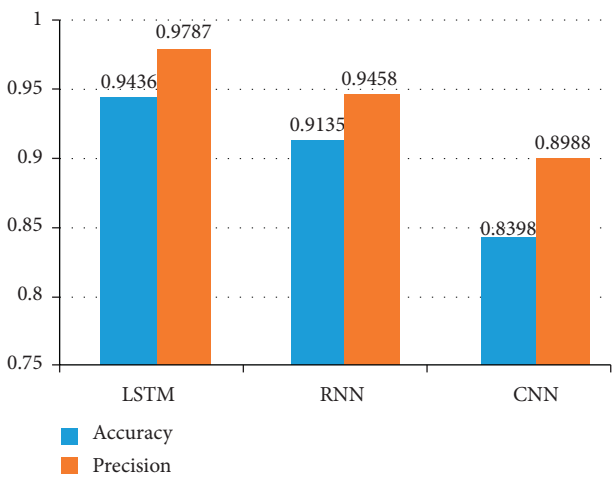



Figure 10: Percentage division method results.

Of course, evaluation of the merits and demerits of an algorithm is highly accurate. Therefore, we also need to use other indicators to judge the performance of the algorithm. This paper uses the recall rate and F1-measure to evaluate the pros and cons of the detection model. The recall rate and F1-Measure of the three algorithms are shown in Figure 11.

It can be seen from the figure that the detection result of the LSTM algorithm is due to the RNN algorithm. The reason may be that the forgetting gate is added to the LSTM algorithm, which can selectively memorize and forget the data, and the data with a longer period will have better performance, so it has a good classification effect on electromagnetic Trojan signals with long duration.

## 6. Detection Accuracy Factor

Through many experiments on a large amount of data, we found that the setting of the gain parameters and the sampling point distance has a great influence on the accuracy of the electromagnetic Trojan detection system.

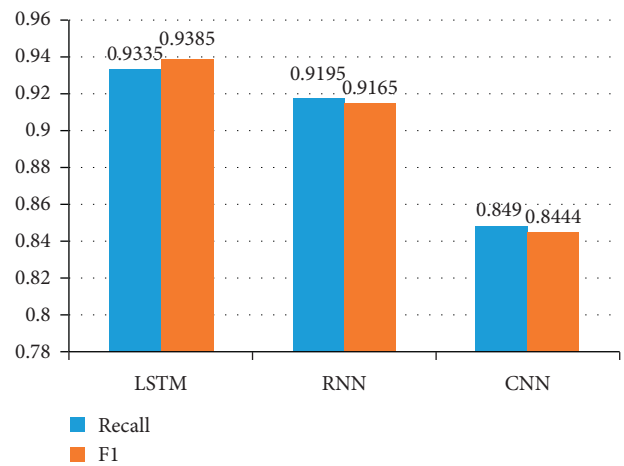



Figure 11: Recall rate and F1 for different algorithms.

*6.1. Effect of Gain.* The gain is the amplification ratio of the acquired signal in dB. The gain calculation formula is shown in the following equation:

$$Z = 10\lg\left(\frac{p_1}{p_2}\right). \tag{6}$$

where $P_1$ is the amplified power, $P_2$ is the power before amplification, nd then $P_1$ is the power after $P_2$ is amplified by $Z$ dB. Usually 10 dB represents 10 times magnification, 100 dB represents 20 times magnification, and −10 dB represents 10 times attenuation. Since the gain is multiplied for all signals (whether it is a noise signal or a signal in normal use), all input samples are amplified by a specified multiple, so the noise is amplified and the gain is amplified. The setting of the value will have an impact on the accuracy of the system.

As shown in Figure 12, the gain is set to 50 dB, and the sampled signal is a waterfall graph when the sampling point is 1 m away from the electromagnetic Trojan signal source. The data in the figure are divided into three parts. According to the gradient color in the legend, it can be seen that the deeper the color, the stronger the signal strength, and the lighter the color, the weaker the signal strength. The left part of the figure consists of several random points, which are noise signals (background signals). The cause is that the signal sampling uses band-pass sampling. A band-pass filter is used to filter the signal outside the sampling bandwidth. The signal strength outside the sampling bandwidth after filtering is greatly reduced, resulting in low-power noise. Two obvious signals can be seen in the figure. For the convenience of description, the line on the left side is set to signal A and the line on the right side is set to signal B.

In Figure 12, signal A is the heartbeat broadcast part of the electromagnetic Trojan signal and signal B is an FM broadcast signal at the experimental location. The broadcast signal is selected to compare the intensity change of the Trojan signal and the background noise intensity variation in the control variable. Figure 13 shows the waterfall graph with the sampling gain set to 10 DB and the same set of signals sampled when the sampling point is 1 m away from the

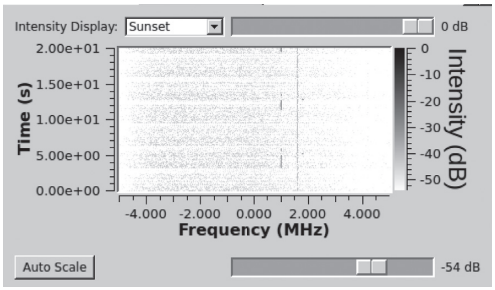



Figure 12: Waterfall diagram with the gain set to 50 and distance of 1 meter.

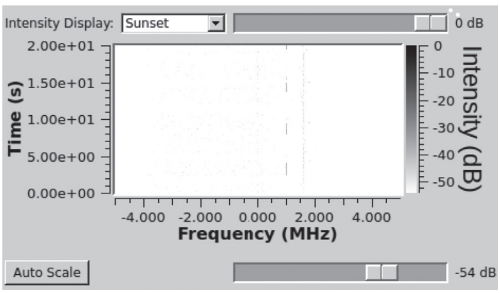



Figure 13: Waterfall diagram with the gain set to 10 and distance of 1 meter.

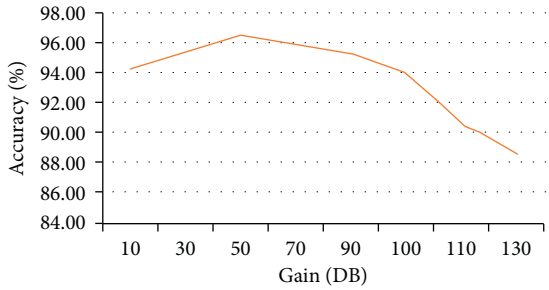



Figure 14: Detection accuracy and gain relationship.

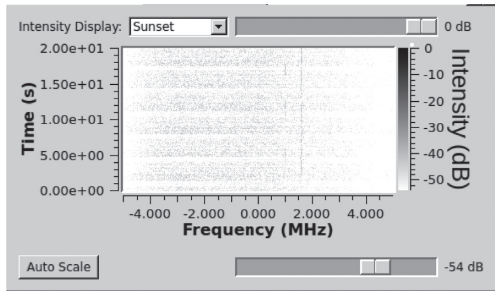



Figure 15: Waterfall diagram with the gain set to 10 and distance of 1 meter.

electromagnetic Trojan source. Comparing Figures 12 and 13, it can be seen that, when the gain is appropriately reduced, the background signal has basically disappeared in the figure, and only the signals A and B are left in the figure. Meanwhile, as compared with Figure 12 with a larger gain, it can be seen that both the signal A and the signal B are greatly weakened. This article sets up a series of experiments for different gains to study the relationship between gain setting and detection accuracy. The test results after the experiment are shown in Figure 14.

It can be seen from Figure 14 that, under the condition of fixed distance, the system detection accuracy increases first and then decreases with the increase of gain, and selecting the appropriate gain value can improve the accuracy of system detection. The reason for this is that the electromagnetic Trojan signal is not transmitted by a professional signal transmitting device. A professional signal transmitting device uses an amplifier to amplify the signal when transmitting the signal, and the electromagnetic Trojan does not have an amplifier when transmitting the signal. Existence: the strength of the signal is slightly higher than the noise signal strength. When the antenna gain is too large, the signal filtering module will filter a part of the electromagnetic Trojan signal as a noise signal, thereby reducing the detection accuracy of the system; meanwhile, when the gain is too small. Electromagnetic Trojan signals and noise signals cannot be distinguished, and data loss can also occur. The difference in gain affects the extraction of the overall features of the signal, but it affects the accuracy of deep neural network modeling and differentiation.

*6.2. Influence of Sampling Distance.* The detection accuracy of the anomaly detection system is also different for the data sampled by different sampling points. Figure 15 shows the signal waterfall diagram when the gain is fixed at 50 dB and the sampling point is 3 meters away from the electromagnetic Trojan signal source. In Figure 12, with the same gain, the sampling point is 1 meter from the Trojan signal source. The intensity of the noise signal and signal B does not change significantly with distance, but the intensity of the electromagnetic Trojan signal is significantly weakened. When the distance increases, the electromagnetic Trojan signal will not be displayed in the waterfall chart. In Figure 4, the electromagnetic wave intensity varies with the propagation distance curve. The electromagnetic wave intensity is inversely proportional to the propagation distance, and the intensity loss is the fastest near the electromagnetic signal emission source. Our experiments used HACKrf one to test the distance and detection accuracy. Set the acceptance gain to 50 dB and test the distance between the acquisition point and the signal source to get the curve, as shown in Figure 15. When the distance is close, the accuracy of electromagnetic Trojan detection is higher. As the distance increases, the accuracy of electromagnetic Trojan detection is lower and lower. When the distance of the electromagnetic Trojan from the sampling point exceeds 4 m, the detection accuracy is greatly reduced, as shown in Figure 16. A study of the propagation distance of CPU leaking Trojans in a paper by Mordechai Guri et al. is as shown in Figure 17. When the electromagnetic Trojan spreads over a certain distance, the attenuation of the Trojan horse will be confused with the signals of other devices around it, failing to properly capture the electromagnetic Trojan signal.

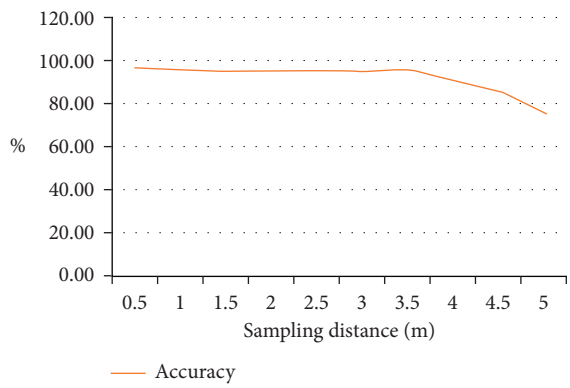



Figure 16: Detection accuracy and distance relationship.

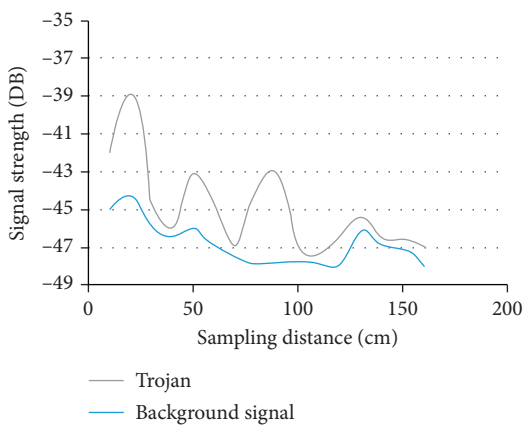



Figure 17: Signal strength and sampling distance relationship.

Comprehensive analysis: the gain and sampling point distance have an impact on the detection accuracy of the system. In the process of using HACKrf one, the gain setting is about 50 dB; the closer the electromagnetic Trojan signal source is to the sampling point, the better it is. The sampling density of the sampling points should be improved to improve the detection accuracy of the system.

## 7. Conclusion

In this paper, we characterize electromagnetic signals from the perspective of time, energy, and information and use deep learning to detect and analyze electromagnetic Trojans. Then, the factors affecting the accuracy of the system detection during the Trojan work are quantitatively analyzed, and the gain setting suitable for using HACKrf one and the optimal acquisition radius of the Trojan signal is found. At the same time, this paper also tests the electromagnetic Trojan in the big data environment and compresses the three methods of deep learning. It can be seen that the LSTM algorithm has advantages in detecting electromagnetic Trojan signals.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 62076042 and 61572086), the Key Research and Development Project of Sichuan Province (Nos. 2020YFG0307 and 2018TJPT0012), the Key Research and Development Project of Chengdu (No. 2019-YF05-02028-GX), the Innovation Team of Quantum Security Communication of Sichuan Province (No. 17TD0009), and the Academic and Technical Leaders Training Funding Support Projects of Sichuan Province (No. 2016120080102643).

## References

[1] H. Tanaka, "Information leakage via electromagnetic emanation and effectiveness of averaging technique," in *Proceedings of the2008 International Conference on Information Security and Assurance (isa 2008)*, pp. 98–101, IEEE, Busan, Korea, April 2008.

[2] M. G. Kuhn and R. J. Anderson, "Soft tempest: hidden data transmission using electromagnetic emanations," *Information Hiding*, Springer, in *Proceedings of the International Workshop on Information Hiding*, pp. 124–142, April 1998.

[3] J. Balasch, B. Gierlichs, and I. Verbauwhede, "Electromagnetic circuit fingerprints for hardware trojan detection[C]." in *Proceedings of the 2015 IEEE International Symposium on Electromagnetic Compatibility (EMC)*, pp. 246–251, IEEE, Dresden, Germany, August 2015.

[4] M. Cozzi, J.-M. Galliere, and P. Maurine, "Thermal scans for detecting hardware Trojans," *Constructive Side-Channel Analysis and Secure Design*, Springer, in *Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 117–132, April 2018.

[5] W. C. Lu, X. J. Liu, and G. Y. Fang, "Ground Penetrating Radar Data Compression Acquisition Based on Perceptual Compression," *Journal of Forestry Research*, vol. 3, pp. 1433–1452, 2011.

[6] Q. Xu, X. H Jiang, L. H. Yao et al., "Summary of hardware trojan detection and prevention research," *Journal of Network and Information Security*, vol. 3, no. 4, pp. 1–13, 2017.

[7] Y. Y. Xu, Q. W. Huang, W. Fan et al., "Modeling and experimental analysis of electromagnetic information leakage based on power line," *Science China Physics, Mechanics & Astronomy*, vol. 57, no. 57, pp. 22–66, 2014.

[8] J. Z. Lu, W. N. Niu, X. L. Liu et al., "A locable abnormal electromagnetic signal joint detection algorithm," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, no. 13, Article ID 1958009, 2019.

[9] Xu Kejun, *Signal Analysis and Processing [M]*, Tsinghua University Press Ltd., Beijing, China, 2006.

[10] J. Z. Lu, K. Chen, Z. L. Zhuo et al., "A temporal correlation and traffic analysis approach for APT attacks detection," *Cluster Computing*, vol. 201712 pages, 2017.

[11] A. Beitler, A. Caracas, T. Eirich et al., p. 219, 2018 Synchronization in software-defined radio systems: U.S. Patent Application 10/067.

[12] M. Guri, A. Kachlon, O. Hasson et al., "GSMem: data exfiltration from air-gapped computers over {GSM} frequencies," in *Proceedings of the 24th {USENIX} Security Symposium ({USENIX} Security 15)*, pp. 849–864, Washington, D.C., USA, August 2015.