Toward Learning Robust Detectors from Imbalanced Datasets Leveraging Weighted Adversarial Training

Kento Hasegawa[1(✉)], Seira Hidano[1], Shinsaku Kiyomoto[1], and Nozomu Togawa[2]

[1] KDDI Research, Inc, 2-1-15 Ohara, Fujimino, Saitama, Japan
{kt-hasegawa,se-hidano,kiyomoto}@kddi-research.jp
[2] Waseda University, 3-4-1 Okubo, Shinjuku, Tokyo, Japan
ntogawa@waseda.jp

Abstract. Machine learning is an attractive technique in the security field to automate anomaly detection and to detect unknown threats. Most of the real-world training samples to learn with neural networks are imbalanced from the viewpoint of their distribution and importance priority on each class. In particular, datasets for security problems are imbalanced in most cases. Learning from an imbalanced dataset may cause the degradation of a classifier's performance, especially in the minority but important classes. We thus propose a new robust learning method for imbalanced datasets using adversarial training. Our proposed method leverages adversarial training to expand classification areas of minority classes. Specifically, we design *weighted adversarial training*, where the perturbation size of adversarial examples is weighted according to the number of samples in each class. We conducted experiments with real-world datasets, and the results demonstrate that our proposed method increases classification performance in both binary and multi-class classifications. Namely, our proposed method makes classifiers more robust even if the dataset is imbalanced, which is useful for us to apply machine learning to security tasks.

Keywords: Neural networks · Adversarial training · Imbalanced datasets · Detection

1 Introduction

Machine learning is an attractive technique in the security field to automate anomaly detection and to detect unknown threats. They also provide us with a significant benefit in various applications, such as complex classification tasks, object recognition, and speech recognition. Toward the real-world applications leveraging machine learning techniques, how to collect high-quality training datasets is a serious concern. For example, the datasets of security-related tasks are imbalanced in most cases [13]. Carefully dealing with such an imbalanced dataset

M. Conti et al. (Eds.): CANS 2021, LNCS 13099, pp. 392–411, 2021.
https://doi.org/10.1007/978-3-030-92548-2_21

is difficult. Although evaluation of a deep neural network and its generalization are performed with balanced datasets, imbalanced datasets must be well considered for real-world applications.

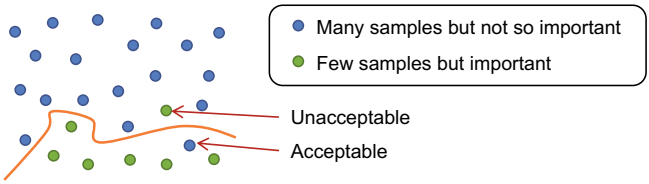

Fig. 1. An example of misclassification of an imbalanced dataset.

In the real-world application of machine learning, collecting a large number of samples to sufficiently train a classifier is essential in the practical use of machine learning. MNIST and CIFAR-10 datasets are well known to evaluate a classifier, in which the numbers of samples in each class are nearly even. However, collecting samples as the numbers of samples in each class become nearly even in the real world is quite difficult. As an example of an intrusion detection system (IDS), malicious traffic rarely appears in an ordinary situation. Moreover, malicious traffic is more remarkable than regular traffic because detecting all malicious traffic is the most important task for the IDS. In this case, we must avoid missing malicious traffics during classification. Figure 1 shows an example of misclassification. As shown in Fig. 1, we consider that misclassifying minority class samples is unacceptable. As exemplified above, considering the importance of each class is needed to classify imbalanced samples.

The existing study [12] investigates how to deal with imbalanced datasets and provides categories of the methods to learn imbalanced datasets. Sampling-based methods [4,11] and cost-sensitive learning methods [6,16] are the major approaches to tackle the imbalanced learning problem. Although several learning algorithms for imbalanced datasets have been developed, most of them focus on specific datasets and situations. Here, we study how to learn imbalanced datasets in order to apply machine learning techniques effectively to security tasks. Specifically, we aim to catch attacked samples as much as possible while keeping the total accuracy high enough.

In this paper, we leverage adversarial training [8,22] that is a defense learning method against adversarial examples. This method makes a classifier robust by replacing a part of training samples with adversarial examples. Adversarial examples, which are crafted by adding perturbation to original samples, are used to alter decision boundaries to desirable shapes. Although this concept can be applied to overcome the issues on imbalanced datasets, there is no detailed discussion on combining adversarial training with imbalanced datasets in the security field. Therefore, we propose a new method leveraging adversarial training for imbalanced datasets and evaluate its effectiveness empirically. The contributions of this paper are summarized as follows:

1. We design *weighted adversarial training* to expand classification areas of minority classes in a given imbalanced dataset. It is based on adversarial training [17] and performed with the distribution of perturbation weights. Our proposed method optimizes the perturbation weights on the basis of the number of training samples of each minority class.
2. Our proposed method takes two approaches, called the *untargeted* and *targeted adversarial training*. We apply both approaches to binary and multiclass classifications and discuss the difference between them based on the experimental results.
3. We applied weighted adversarial training to an IDS using a traffic flow dataset, which is a security-related task. The experimental results demonstrate that our proposed method successfully works for both binary and multiclass classification tasks.
4. We further analyze the classification results by visualizing the classification area using the t-SNE algorithm.

This paper is organized as follows: Sect. 2 shows some of the related works with our study. Section 3 describes preliminary definitions and equations. Section 4 proposes a robust learning algorithm for imbalanced datasets leveraging weighted adversarial training. Section 5 shows the experiments and their results of our proposed method. Section 6 gives concluding remarks.

2 Related Works

Machine learning is now being leveraged in the security field for various purposes, such as malware detection and abnormal traffic detection. In order to effectively apply a machine learning technique to such an anomaly detection system, collecting high-quality and high-quantity training samples is important. In general, we rarely obtain abnormal samples in the real world, except for the large volume of an attack such as a distributed denial-of-service (DDoS) attack. When we try to collect attack samples from network traffic in an ordinal way, only a few attacked samples might exist. Therefore, the samples collected from the real world could be imbalanced. To effectively leverage machine learning techniques for security, we should carefully deal with such an imbalanced dataset.

How to deal with imbalanced datasets for machine learning has been discussed in recent studies. There are mainly two approaches in training imbalanced datasets: 1) a data-level approach and 2) an algorithm-level approach [12]. A data-level approach balances the number of samples in each class by over- or under-sampling the datasets. The Synthetic Minority Over-sampling TEchnique (SMOTE) method [4], ADAptive SYNthetic Sampling (ADASYN) method [11], and their improved variety of methods have often been adopted to imbalanced learning problems. In the over-sampling methods, new samples are generated next to the original samples that are likely to be classified mistakenly. However, effectively choosing the samples, which would be mistakenly classified, is difficult. Recently, generative adversarial networks (GANs) have been often used for sample generation [5,18,20]. An algorithm-level approach often

utilizes a loss function that considers the weights of each class in a dataset [6,16]. Some cost-sensitive methods are also leveraged to learn imbalanced datasets [14]. Margin-based methods have also been proposed in recent years [3,10].

Although the sampling-based approach is a simple way to learn imbalanced datasets, there are several drawbacks. In an over-sampled training approach, the generated samples might be redundant to improve classification performance. In an under-sampled training approach, the distribution of the dataset might be altered by sampling, which might lose potential representation. When we use a sampling-based approach, we should take care of the problems above.

Adversarial training is one of the approaches to make a classifier robust. This approach is essentially to enhance the robustness against adversarial examples [22]. Adversarial example images look natural for humans, but unsophisticated classifiers may misclassify them. The empirical description of adversarial examples is introduced by [8]. Recent attack methods such as Fast Gradient Sign Method (FSGM) [8] and Projected Gradient Descent (PGD) [17] have enabled to generate deceptive adversarial examples. Adversarial examples may become severe threats against physical world systems such as autonomous vehicles and robot vision [1]. In order to tackle the problem of adversarial example attacks, the first defense method has been proposed by [8]. PGD-based adversarial training [17] is one of the methods to defeat adversarial examples by learning them. The PGD-based method makes a classifier robust against adversarial examples by training adversarial examples generated during a training iteration. Other adversarial training methods, such as [24] and [25], have recently been proposed. Due to the detailed investigation of adversarial examples, the mechanism of deep neural networks has been well studied and becomes clarified. That will improve performance and enhance the robustness of deep neural networks [9].

As mentioned above, adversarial training helps us to improve the robustness of a classifier. As a result, that should enhance the classification performance as well as defend the classifier against adversarial attacks. From the perspective of improving classification performance, adversarial training is one of the promising approaches. Recently, as a new approach, adversarial training has been used for an imbalanced classification task [23]. The authors have proposed a new algorithm, the Wasserstein PGD (WPGD) model, which deals with the imbalanced dataset and manages the trade-off between the accuracy and robustness of the classifier. The WPGD model utilizes a Wasserstein distance to evaluate the difference between the genuine and predicted class. Based on the idea, the WPGD model introduces the Wasserstein loss function when generating perturbation. However, the WPGD model could not consider the borderline between the neighbor classes, particularly in multiclass classification. Another study [15] addresses the imbalanced classification task by translating some majority class samples to the target minority class. The translation of majority class samples is performed by adding a small noise to the majority class samples toward the target minority class, and the translated samples are re-labeled as the target minority class. From the recent studies, adversarial training is one of the promising approaches to cope with imbalanced classification.

In this paper, we proposed a new method addressing imbalanced classification with a PGD-based approach.

3 Preliminaries

This section introduces the backgrounds and notations necessary for our proposed method.

3.1 Machine Learning

Let $\mathbf{x} \in \mathbb{R}^d$ be a d-dimensional feature vector, and let $\mathbf{y} = [y_1, \ldots, y_s] \in \{0, 1\}^s$ be a one-hot vector that indicates the class of the feature vector $\mathbf{x}$. If $\mathbf{x}$ belongs to a class $i \in [s]$, $y_i = 1$, otherwise $y_i = 0$. We denote by $\mathcal{D}$ a training dataset consisting of N pairs of a feature vector $\mathbf{x}$ and the class label $\mathbf{y}$. In supervised learning, an s-class classifier $F : \mathbb{R}^d \to \mathbb{R}^s$ is generated from a training dataset $\mathcal{D}$. The classifier F is parameterized by θ, and θ is chosen to minimize an expected loss function $\mathcal{L}_\theta(\mathcal{D})$ of the training dataset $\mathcal{D}$. Given a loss function $l_\theta(\mathbf{x}, \mathbf{y})$ of a training sample $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$, $\mathcal{L}_\theta(\mathcal{D})$ can be written as $\mathcal{L}_\theta(\mathcal{D}) = \frac{1}{N} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} l_\theta(\mathbf{x}, \mathbf{y})$. In this paper, we especially focus on a deep neural network whose final layer has a softmax function. Let $o(\cdot)$ denote the output of the last layer before the softmax layer. The model's output is expressed as $F(\mathbf{x}) = \mathrm{softmax}(o(\mathbf{x}))$. When the number of samples in a training dataset $\mathcal{D}$ is greatly different for each class, we call $\mathcal{D}$ an *imbalanced dataset*. We also call learning a robust classifier F from an imbalanced dataset *imbalanced learning*.

3.2 Adversarial Examples

Adversarial examples are used to deceive a classifier and induce misclassification. Let $\mathbf{r} \in \mathbb{R}^d$ be a small perturbation. Given a sample $\mathbf{x}$, an adversarial example $\mathbf{x}'$ is generated by $\mathbf{x}' = \mathbf{x} + \mathbf{r}$. However, optimizing the perturbation $\mathbf{r}$ is a difficult problem in terms of computational complexity. Thus the focus of interest of early studies was to generate adversarial examples effectively.

The *PGD* [17] method is a well-known solution for generating adversarial examples. This method is optimized for the L_∞ norm of the perturbation. The perturbation is iteratively updated K times. Let α be the step size of the perturbation at each iteration, and let ϵ be the maximum size of the perturbation. Given a sample $(\mathbf{x}, \mathbf{y})$, the PGD method generates the adversarial example with the following update function:

$$\begin{cases} \mathbf{x}'^{(0)} & = \mathbf{x} \\ \mathbf{x}'^{(t+1)} & = \mathrm{clip}_{\mathbf{x},\epsilon}\left[\mathbf{x}'^{(t)} + \alpha\, \mathrm{sign}(\nabla_{\mathbf{x}'^{(t)}} l_\theta(\mathbf{x}'^{(t)}, \mathbf{y}))\right], \end{cases} \tag{1}$$

where $\mathrm{clip}_{\mathbf{x},\epsilon}[\mathbf{a}]$ projects each element $a_i \in \mathbf{a}$ onto the range $[x_i - \epsilon, x_i + \epsilon]$.

If it is required to misclassify a sample with a label $i \in [s]$ to a specific class $j \in [s] \setminus i$, the update function at t-th iteration can be written as follows:

$$\mathbf{x}'^{(t+1)} = \underset{\mathbf{x},\epsilon}{\mathrm{clip}}\left[\mathbf{x}'^{(t)} - \alpha\, \mathrm{sign}(\nabla_{\mathbf{x}'^{(t)}} l_\theta(\mathbf{x}'^{(t)}, \mathbf{y}_j))\right], \tag{2}$$

where $\mathbf{y}_j$ is a vector that indicates a class j. We call the method with the Eq. (1) (resp. the Eq. (2)) the *untargeted* (resp. *targeted*) PGD method.

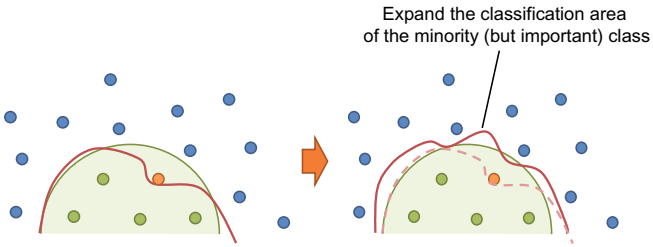

Fig. 2. The concept of the proposed method. (Color figure online)

3.3 Adversarial Training

The basic idea of *adversarial training* is to inject adversarial examples themselves into the training dataset to make the model robust to adversarial examples. For instance, the expected loss function $\mathcal{L}_\theta(\mathcal{D})$ for adversarial training using the untargeted PGD method can be written as follows:

$$\mathcal{L}_\theta(\mathcal{D}) = \frac{1}{N} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \left[l_\theta(\mathbf{x},\mathbf{y}) + \max_{\mathbf{r}\in\mathcal{S}(\mathbf{x})} l_\theta(\mathbf{x}+\mathbf{r},\mathbf{y}) \right], \tag{3}$$

where $\mathcal{S}(\mathbf{x})$ is the perturbation constraint for a given sample $\mathbf{x}$. In contrast, different loss functions can be defined for adversarial training with the targeted PGD method based on the purpose. We thus design a special loss function to imbalanced learning in Sect. 4.2. Hereinafter, we call adversarial training with the untargeted (resp. targeted) PGD method the *untargeted* (resp. *targeted*) *adversarial training*.

4 Method

In this section, we propose a robust learning method for imbalanced datasets leveraging adversarial training.

4.1 Overview

Our proposed method aims to expand the classification area of the minority but important classes as much as possible. Figure 2 illustrates how to determine a decision boundary between the majority and the minority classes in binary classification. In Fig. 2, the blue samples belong to the majority class, while the green samples belong to the minority class. The red line is the decision boundary formed by a classifier. The green shaded area shows the ideal distribution of the minority class samples. In the left figure, while the orange sample originally belongs to the minority class, it is outside the red line; therefore, the classifier misclassifies it as the majority class. We thus consider expanding the decision boundary towards the majority class in order to avoid such misclassification.

The right figure in Fig. 2 shows an example of the expansion. Our proposed method actualizes this kind of manipulation by introducing *weighted adversarial training* to imbalanced learning.

4.2 Problem Settings

We formally define the problem for our proposed method and provide the loss function for our imbalanced learning. Here let us consider an ideal training dataset $\mathcal{D}^*$. Let $\mathcal{D}_i^*$ be the set of training samples $(\mathbf{x}^*, \mathbf{y}) \in \mathcal{D}^*$ such that $\mathbf{y}$ indicates a class $i \in [s]$. Given an imbalanced training dataset $\mathcal{D}$, we assume that $\mathcal{D}_i^* \approx \{(\mathbf{x}^*, \mathbf{y}) \mid \|\mathbf{x}^* - \mathbf{x}\|_\infty \leq \epsilon_i', (\mathbf{x}, \mathbf{y}) \in \mathcal{D}_i\}$, where $\mathcal{D}_i$ is the set of training samples $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ such that $\mathbf{y}$ indicates the class i, and ϵ_i' is the perturbation size for the class i. $\|\cdot\|_\infty$ is the L_∞ norm. In other words, we assume that any ideal training sample $(\mathbf{x}^*, \mathbf{y}) \in \mathcal{D}^*$ can be represented by adding an appropriate d-dimensional perturbation $\mathbf{r}$ satisfying $\|\mathbf{r}\|_\infty \leq \epsilon_i'$ to a training sample $\mathbf{x}$ in an imbalanced dataset.

However, in order to generate the ideal training dataset $\mathcal{D}^*$ from a given training dataset $\mathcal{D}$, the maximum perturbation size ϵ_i' should be optimized for each class i. The values of ϵ_i' for minor classes will be larger than those for majority classes. To simplify the problem, we thus assume that ϵ_i' can be represented as a function g of the number of training samples for each class i, $n_i = |\mathcal{D}_i|$, i.e., $\epsilon_i' = \xi \cdot g(n_i)$, where ξ is a positive parameter. Examples of the function g are given in Sect. 5.1. In addition, it is difficult to involve all samples in $\mathcal{D}^*$ in the training of a classifier F, as the number of feature vectors $\mathbf{x}^*$ satisfying $\|\mathbf{x}^* - \mathbf{x}\|_\infty$ for a given sample $\mathbf{x}$ is exponentially large. We thus suggest adding only samples that change the shape of the decision boundary. The classification area expanded with such samples will cover many other samples that are not included in a training dataset $\mathcal{D}$ and are inside the added samples. We finally define an expected loss function for a given imbalanced dataset $\mathcal{D}$ as follows:

$$\mathcal{L}_\theta(\mathcal{D}) = \frac{1}{N} \sum_{i \in [s]} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_i} \{l_\theta(\mathbf{x}, \mathbf{y}) + \max_{\mathbf{r}} l_\theta(\mathbf{x} + \mathbf{r}, \mathbf{y})\}, \tag{4}$$

$$\text{s.t. } \|\mathbf{r}\|_\infty \leq \xi \cdot g(n_i). \tag{5}$$

The Eq. (5) is formulated as untargeted adversarial training. It should be noted that untargeted adversarial training might cause *label leaking* effect [17]. Because of this effect, we expect that untargeted adversarial training will expand more classification area than targeted adversarial training. Therefore, untargeted adversarial training gains better classification performance in terms of recall.

Our proposed learning method generates a classifier F that minimizes the loss function $\mathcal{L}_\theta$ while optimizing the parameter ξ. This optimization problem is similar to the problem for adversarial training shown by the Eq. (3). The key differences are that (1) the maximum size of perturbation, ϵ_i' $(= \xi \cdot g(n_i))$, is different between classes, and that (2) it is required to find out the optimal values of ϵ_i' that improve the accuracy of minority classes while keeping that

of majority classes. Since the perturbation for adversarial examples should be sufficiently small for avoiding detection, a fixed small value of ϵ is used for adversarial training. However, in imbalanced learning, it is preferable to enlarge the classification area as much as possible. Therefore, our proposed method optimizes the values of ϵ'_i by exploring an optimal solution of ξ.

Application to Multiclass Classification. In imbalanced learning for multiclass classification, the classification area of each minority class should be expanded in the directions of multiple neighbor classes. However, there is a possibility that training only a single adversarial example for a training sample cannot realize this purpose due to the inappropriate structure of classification areas. We thus suggest generating multiple adversarial examples for a training sample. Each adversarial example is used for expanding the classification area towards the corresponding neighbor class. In Sect. 5.3, we show through experiments that imbalanced learning with multiple adversarial examples has higher performance than that with a single adversarial example. The loss function for multiclass classification can be written as follows:

$$\mathcal{L}_\theta(\mathcal{D}) = \frac{1}{N} \sum_{i \in [s]} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}_i} \{ l_\theta(\mathbf{x}, \mathbf{y}) + \sum_{j \in [s] \setminus i} \max_{\mathbf{r}_j} -l_\theta(\mathbf{x} + \mathbf{r}_j, \mathbf{y}_j) \},$$
$$\text{s.t. } \|\mathbf{r}_j\|_\infty \leq \xi \cdot g(n_i), \tag{6}$$

where $\mathbf{y}_j$ is a vector that indicates a class j, and $\mathbf{r}_j$ is a perturbation for the class j.

Unlike the Eq. (5), the Eq. (6) is formulated as targeted adversarial training. This is because untargeted adversarial training cannot generate multiple different adversarial examples for a training sample. Our proposed learning method for multiclass classification seeks the optimal multiple perturbations $\mathbf{r}_j$ that minimize the Eq. (6).

4.3 Algorithm

We then consider an algorithm to solve θ minimizing the Eq. (5) with fixed values of ϵ'_i. Our proposed algorithm is based on the PGD method shown in Sect. 3. While the PGD method uses the non-weighted parameters α and ϵ for all classes, we introduce weight vectors $\boldsymbol{\alpha}' \in \mathbb{R}^s$ and $\boldsymbol{\epsilon}' \in \mathbb{R}^s$. In the PGD method, α is used to control the perturbation size at each iteration, and ϵ indicates the maximum perturbation size. $\boldsymbol{\alpha}'$ and $\boldsymbol{\epsilon}'$ have nearly the same meanings as α and ϵ, yet both of the parameters consider the weights of the corresponding classes. $\boldsymbol{\alpha}'^{\mathsf{T}}\mathbf{y}$ obtains the perturbation size at an iteration corresponding to a class $i \in [s]$. Similarly, $\boldsymbol{\epsilon}'^{\mathsf{T}}\mathbf{y}$ obtains the maximum perturbation size corresponding to the class i. Therefore, given a training sample $(\mathbf{x}, \mathbf{y})$, the update function of the weighted adversarial training is expressed as follows:

$$\begin{cases} \mathbf{x}'^{(0)} = \mathbf{x} \\ \mathbf{x}'^{(t+1)} = \\ \quad \text{clip}_{\mathbf{x}, \boldsymbol{\epsilon}'^{\mathsf{T}}\mathbf{y}} \left\{ \mathbf{x}'^{(t)} + \boldsymbol{\alpha}'^{\mathsf{T}}\mathbf{y} \, \text{sign}(\nabla_{\mathbf{x}'^{(t)}} \mathcal{L}_\theta(\mathbf{x}'^{(t)}, \mathbf{y})) \right\}. \end{cases} \tag{7}$$

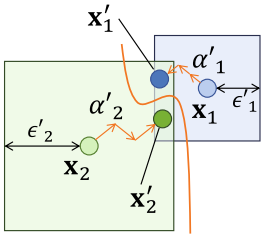

Fig. 3. An example of the weighted adversarial training. (Color figure online)

Algorithm 1. Imbalanced leaning by weighted adversarial training

Inputs: Classifier F, training dataset $\mathcal{D}$, minibatch size m, number of iterations of generating adversarial examples K, and ratio of trained adversarial examples p.

1: Initialize the classifier F.
2: **repeat**
3: Obtain m samples from a training dataset $\mathcal{D}$ so that the number of samples of each class is even, and store them as a minibatch B.
4: Iterate adversarial training steps according to the Eq. (7) for K times, and generate adversarial examples.
5: Replace p% of the samples in B with the generated adversarial examples and store them as a minibatch B'.
6: Perform one training step of the classifier F using the minibatch B'.
7: **until** training converged.

Figure 3 depicts the proposed method. The blue sample $\mathbf{x}_1$ belongs to the majority class, and the green sample $\mathbf{x}_2$ belongs to the minority class. Here, we define weight vectors $\boldsymbol{\alpha}'$ and $\boldsymbol{\epsilon}'$ as $\boldsymbol{\epsilon}' = (\epsilon'_1, \epsilon'_2) \in \mathbb{R}^2$, and $\boldsymbol{\alpha}' = (\alpha'_1, \alpha'_2) \in \mathbb{R}^2$, where the first class is the majority class and the second class is the minority class. Since we regard that the minority class is more important than the majority class, we set variables as $\epsilon'_1 < \epsilon'_2$ and $\alpha'_1 < \alpha'_2$. Then, the distribution of the perturbation is illustrated as the blue and green shaded areas shown in Fig. 3. Based on the PGD-based adversarial example generation and training, the adversarial examples are generated as $\mathbf{x}'_1$ and $\mathbf{x}'_2$. As a result, the decision boundary can be described as the orange curve in Fig. 3. Since the classification area for the minority class is expanded, the classification performance of the minority class is expected to be improved by the proposed method. Note that, in case of a multiclass classification problem, we define $\boldsymbol{\epsilon}' = (\epsilon'_1, \cdots, \epsilon'_s) \in \mathbb{R}^s$ and $\boldsymbol{\alpha}' = (\alpha'_1, \cdots, \alpha'_s) \in \mathbb{R}^s$.

The entire learning process is described in Algorithm 1. To balance the learning dataset during the training epoch, we draw the samples from the training dataset so that the number of samples in each class is even in the minibatch. In order to further enhance the classification performance for imbalanced datasets, the perturbation of the generated adversarial examples in the proposed method is weighted based on $\boldsymbol{\alpha}'$ and $\boldsymbol{\epsilon}'$ as shown in the Eq. (7).

5 Experiments

In this section, we evaluate the effectiveness of our proposed method through experiments with real-world datasets related to security tasks. We compare with well-known sampling methods for imbalanced learning and show that our proposed method improves the classification performance of minority classes.

5.1 Setup

Dataset. We use multiple datasets included in CICIDS2017 [21], which is designed to evaluate network-based IDSs. CICIDS2017 has benign and attacked network traffic samples. Each record consists of the statistics of network traffics. In the experiments, we perform both binary and multiclass classification with the datasets.

Specifically, CICIDS2017 contains seven datasets for a machine-learning purpose. Each dataset includes statistical samples of network packets with different times and different attacks. Some datasets have binary-class labels that show benign or attacked, and others have multiclass labels that show benign or one of the attack types. In order to deal with the imbalanced classification, we use the five datasets: **Bot**, **Infiltration**, **DoS**, **Pataoor**, and **WebAttacks**, out of the seven ones in CICIDS2017. In the selected five datasets, the 'Benign' class is the largest (i.e., it is the majority class), and the other attacked classes are smaller than the 'Benign' class (i.e., they are the minority classes). The two datasets **Bot** and **Infiltration** contain binary-class labels, and the other three datasets **DoS**, **Pataoor**, and **WebAttacks** contain multiclass labels. Table 1 summarizes the contents of the datasets we use in this paper. Each sample consists of 78 feature values and a label. The feature values represent the statistical characteristics of the packet flow, such as a destination port, the total length of the packets, the flow packets per second. The label shows the attack types, including benign.

In the experiments, we standardize the dataset so that the mean of each column is 0 and the variance of that is 1. Note that we replace 'NaN' and infinity values with 0 in the pre-processing phase because they appear in the 'Flows per second' and 'Packets per second' columns at '0' duration time.

After the standardization, we use 80% of the dataset as training samples and the rest as test samples.

Models. To evaluate the classification performance, we perform experiments with several models. The model used in this paper is summarized as follows:

1. **Normal:** A normal classifier with a cross-entropy without any balancing methods is used.
2. **B.B.** (Balanced mini-Batch): The 'Normal' model is used, but the trainer draws the training samples so that the number of samples in each class is equal in a minibatch (only the third step in Algorithm 1).
3. **SMOTE:** The training samples are over-sampled by the SMOTE [4] method beforehand.

Table 1. Contents of CICIDS 2017 Dataset.

File names	Classes	Samples
Patator: Tuesday-WorkingHours.pcap_ISCX.csv		
	Benign	432,074
	FTP-Patator	7,938
	SSH-Patator	5,897
DoS: Wednesday-workingHours.pcap_ISCX.csv		
	Benign	440,031
	DoS Hulk	231,073
	DoS GoldenEye	10,293
	DoS slowloris	5,796
	DoS Slowhttptest	5,499
	Heartbleed	11
WebAttacks: Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv		
	Benign	168,186
	Web Attack Brute Force	1,507
	Web Attack XSS	652
	Web Attack Sql Injection	21
Infiltration: Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv		
	Benign	288,566
	Infiltration	36
Bot: Friday-WorkingHours-Morning.pcap_ISCX.csv		
	Benign	189,067
	Bot	1,966

4. **ADASYN:** The training samples are over-sampled by the ADASYN [11] method beforehand.
5. **WPGD:** A robust classification method employing a Wasserstein loss [23].
6. **UT-wPGD:** Our proposed method is applied to the classifier with UnTargeted Weighted PGD-based perturbation. The perturbation is generated with the untargeted adversarial training. Its amount is weighted based on the cardinality of the class where the sample belongs.
7. **T-wPGD:** Our proposed method is applied to the classifier with Targeted Weighted PGD-based perturbation. The perturbation is generated with the targeted adversarial training toward another class. Its amount is weighted based on the cardinality of the class where the sample belongs.

We use a multi-layer perceptron model to train and classify the dataset. The structure and hyper-parameters of the multi-layer perceptron are the same in all the models above. The model has three middle layers with $[64, 32, 32]$ units. We use sigmoid activation functions in the middle layers and a soft-max function in the output layer. The optimization method is Adam. We train a model for 50 epochs with a minibatch size of 256.

We tune the hyper-parameters of the PGD models by changing them. The weighting functions determine the weighting values with the cardinality of each class as an argument. Here, we set the maximum size of the perturbation vector

Table 2. Weighting function. $n_{\mathsf{max}} = \max_i n_i$ in the dataset $\mathcal{D}$.

Name	Function
ln	$g_{\mathsf{ln}}(n_i) = \ln(n_{\mathsf{max}})/ln(n_i)$
lnm	$g_{\mathsf{lnm}}(n_i) = \ln(n_{\mathsf{max}}/n_i)$
sqrt	$g_{\mathsf{sqrt}}(n_i) = \sqrt{n_{\mathsf{max}}/n_i}$
4thr	$g_{\mathsf{4thr}}(n_i) = \sqrt[4]{n_{\mathsf{max}}/n_i}$

$\boldsymbol{\epsilon}'$ as $k \cdot \boldsymbol{\alpha}'$ and the number of iteration K as $3/2 \cdot k$ so that $\boldsymbol{\epsilon}' = k \cdot \boldsymbol{\alpha}'$ will clip too large perturbation generated by $K \cdot \boldsymbol{\alpha}'$. In the experiments, we set $k = 10$. As shown in Sect. 4.2, the weight of the i-th class ϵ'_i can be represented as a function g with the cardinality of the class $n_i = |\mathcal{D}_i|$, i.e., $\epsilon'_i = \xi \cdot g(n_i)$. In the experiments, we set ξ to 0.001, 0.005, 0.01, and 0.05. Also, for comparison, we set the weights of the Wasserstein matrix according to $g(n_i)$. Table 2 describes the weighting functions used in the experiments.

5.2 Evaluation Metrics

In the experiments, we evaluate the experimental results with several metrics.

In the evaluation, we focus on not missing any attacked samples. In other words, our top priority is to catch attacked samples as much as possible while keeping the total accuracy high enough. We introduce the evaluation metrics from the viewpoint of this priority.

A classifier classifies the samples in a training dataset as either a majority class or a minority class in binary classification. Let $\mathcal{D}_-$ be a set of majority class samples, and $\mathcal{D}_+$ be a set of minority class samples in a dataset. Then, the classifier classifies the two-class dataset $\mathcal{D} = \mathcal{D}_- \cup \mathcal{D}_+$ as either the majority or minority class. We denote by $\mathcal{E}_-$ a dataset classified as the majority class and denote by $\mathcal{E}_+$ a dataset classified as the minority class by the classifier. To see the overall classification performance, we can use the accuracy. However, the accuracy puts weight to the majority class. Therefore, we also refer to the recall to evaluate whether we have detected a minority class of attacked samples. Then, the accuracy and recall values are expressed as follows:

$$\text{Accuracy} = \frac{|\mathcal{D}_- \cap \mathcal{E}_-| + |\mathcal{D}_+ \cap \mathcal{E}_+|}{|\mathcal{D}|} \tag{8}$$

$$\text{Recall} = \frac{|\mathcal{D}_+ \cap \mathcal{E}_+|}{|\mathcal{D}_+|} \tag{9}$$

where $|\cdot|$ is a cardinality of a class.

In multiclass classification, we obtain the average score for each class. Here, let $\mathcal{E}_i$ be the dataset labeled as the i-th class by the classifier. Then, we define the overall accuracy, Accuracy_M as follows:

$$\text{Accuracy}_M = \frac{\sum_{i \in [s]} |\mathcal{D}_i \cap \mathcal{E}_i|}{|\mathcal{D}|} \tag{10}$$

Table 3. CICIDS2017 binary classification results.

Dataset	Model	Parameters	Accuracy	**Recall**
Bot				
	Normal		0.996	0.648
	B.B.		0.984	0.988
	SMOTE		0.978	0.990
	ADASYN		0.984	0.990
	WPGD	$g_{\mathsf{lg}}, \xi = 0.01$	0.981	0.990
	UT-wPGD	$g_{\mathsf{lgm}}, \xi = 0.01$	0.974	**0.995**
	T-wPGD	$g_{\mathsf{lg}}, \xi = 0.05$	0.981	0.993
Infiltration				
	Normal		1.000	0.778
	B.B.		0.996	0.778
	SMOTE		1.000	0.889
	ADASYN		1.000	0.889
	WPGD	$g_{\mathsf{lg}}, \xi = 0.01$	0.996	0.778
	UT-wPGD	$g_{\mathsf{lgm}}, \xi = 0.01$	0.996	**1.000**
	T-wPGD	$g_{\mathsf{4thr}}, \xi = 0.005$	0.998	0.889

There are two averaging methods in terms of the recall score: micro-averaging and macro-averaging [7]. The micro-averaging is an average weighted by the class distribution, and it is equivalent to the overall accuracy. The macro-averaging is an arithmetic mean of the recall score for each class, and it would consider the recall of each class fairly. In this paper, we use the macro-averaged recall score for evaluation. The recall Recall_M used for evaluating multiclass classification are expressed as follows:

$$\mathrm{Recall}_M = \frac{1}{s} \sum_{i \in [s]} \mathrm{Recall}_i \tag{11}$$

where $\mathrm{Recall}_i = |\mathcal{D}_i \cap \mathcal{E}_i| / |\mathcal{D}_i|$. Note that the Recall_M score is also known as the *balanced accuracy* [2].

In the following section, we evaluate the classification results based on accuracy and recall scores. When evaluating imbalanced datasets, the accuracy tends to become high enough because the classifier may classify most of the samples as a majority class. If the samples in minority classes are misclassified, it does not affect the accuracy significantly. As mentioned at the top of this section, our goal is to classify the minority-class samples correctly. In this sense, we use the recall scores to see how many samples a model correctly classifies as their original classes.

5.3 Experimental Results

First, we explore the weighting function and parameter ξ with which we can obtain the best classification performance for each model. Then, we pick up the best classification results over the explored parameters for each model.

Table 4. CICIDS2017 multiclass classification results.

Dataset	Model	Parameters	Accuracy_M	Recall_M
DoS				
	Normal		0.997	0.961
	B.B.		0.998	0.964
	SMOTE		0.998	0.964
	ADASYN		0.995	0.994
	WPGD	$g_{\ln}, \xi = 0.01$	0.997	0.964
	UT-wPGD	$g_{\ln m}, \xi = 0.05$	0.997	**0.997**
	T-wPGD	$g_{4\text{thr}}, \xi = 0.01$	0.996	**0.997**
Patator				
	Normal		0.999	0.992
	B.B.		0.999	**0.997**
	SMOTE		0.999	0.996
	ADASYN		0.992	0.867
	WPGD	$g_{\ln}, \xi = 0.01$	0.999	**0.997**
	UT-wPGD	$g_{\text{sqrt}}, \xi = 0.05$	0.997	**0.997**
	T-wPGD	$g_{4\text{thr}}, \xi = 0.05$	0.998	**0.997**
WebAttack				
	Normal		0.994	0.502
	B.B.		0.992	0.709
	SMOTE		0.991	0.758
	ADASYN		0.990	0.721
	WPGD	$g_{\ln}, \xi = 0.01$	0.992	0.723
	UT-wPGD	$g_{\ln}, \xi = 0.005$	0.978	0.774
	T-wPGD	$g_{4\text{thr}}, \xi = 0.005$	0.989	**0.775**

Binary Classification Results. Table 3 shows the parameters and results of binary classification. The 'Parameters' column shows the weighting function (see Table 2) and the parameter ξ used for each PGD model. We show the two metrics: Accuracy and Recall as shown in the Eqs. (8) and (9) for binary classification. As shown in Table 3, we use Bot and Infiltration datasets to evaluate binary classification. They contain benign and attacked samples, and attacked samples are in the minority class. From the results in Table 3, the Normal model obtains the best accuracy in all the models in both Bot and Infiltration datasets. However, a high accuracy score in imbalanced classification means that a classifier correctly classifies most of the majority class samples, not the minority class samples. Actually, the recall scores of both Bot and Infiltration datasets are the lowest in all the models. In contrast, our proposed method, the **UT-wPGD** and **T-wPGD** models obtain the first and second highest recall scores in all the models for each dataset. It should be noted that the **B.B.** model itself improves the recall score compared to the Normal model. Combining the **B.B.** model and weighted PGD-based perturbation could further improve the recall score. Our proposed method successfully works in the two datasets.

Multiclass Classification Results. Table 4 shows the results of multiclass classification. The evaluation metrics used in this table are as shown in the Eqs. (10) and (11). To see the detailed classification results in each dataset, Table 5 shows the number of correctly classified samples for each class. In multiclass classification, we evaluate the classification results based on the Accuracy_M and Recall_M scores. In the multiclass classification, we aim to maximize the recall scores of the minority classes; therefore a high Recall_M score is the desired result.

In the DoS dataset, the **UT-wPGD** and the **T-wPGD** models obtain the highest Recall_M scores. As shown in Table 5, the **UT-wPGD** and **T-wPGD** models correctly classify more samples of DoS GoldenEye, DoS Slowhttptest, DoS slowloris, and Heartbleed classes than the Normal model. These classes include fewer samples than the other classes. From the viewpoint of classifying minority class samples (especially the bottom three classes), the **T-wPGD** model outperforms the other models.

In the Patator dataset, the difference between the models is slight. However, our proposed models successfully obtain equal to or better classification results than the existing methods.

In the WebAttack dataset, the **T-wPGD** model outperforms other models. This is because the **T-wPGD** model successfully classifies the minority classes with good balance. In particular, the 'Web Attack Brute Force' and 'Web Attack XSS' classes have similar features, and most of the classifiers are prone to misclassify them. A recent study [19] has also shown that the accuracy and recall score of 'Web Attack XSS' is not so high when those of 'Web Attack Brute Force.' Moreover, detecting 'Web Attack Sql Injection' is also difficult because it contains a tiny number of samples. In this paper, as shown in Table 5, the numbers of correctly classified samples in the 'Web Attack Brute Force' and 'Web Attack XSS' classes are unstable. Actually, the 'Web Attack Brute Force' class contains 293 test samples, and the 'Web Attack XSS' class contains 128 test samples. The Normal model could not consider the class distribution, and therefore classification borderline is not well constructed. The **UT-wPGD** model adds perturbation toward the direction where the loss is most increased. In other words, the model cannot control the direction of the perturbation directly. Consequently, the perturbed sample might invade another smaller class, whereas we expect to invade a larger class. In contrast, the **T-wPGD** model controls the direction of the perturbation directly, decreasing the loss as much as possible, as mentioned in Sect. 4.2 with the Eq. (6). As a result, the perturbed samples would not invade another class any more than necessary. Therefore, the **T-wPGD** model successfully classifies them considering class weights, and the Recall_M score becomes the best in all the models.

In conclusion, the **UT-wPGD** and **T-wPGD** models successfully classify the real-world imbalanced datasets. When the dataset is a binary-class set, the **UT-wPGD** model would be more effective due to the label leaking effect. Otherwise, the **T-wPGD** model would effectively work considering the class weights.

Table 5. CICIDS2017 correctly classified samples.

Dataset	Class	Normal	UT-wPGD	T-wPGD
DoS				
	Benign	87,714	87,760	87,594
	DoS Hulk	46,120	46,059	46,094
	DoS GoldenEye	2,008	2,024	2,023
	DoS slowloris	1,141	1,148	1,148
	DoS Slowhttptest	1,109	1,112	1,114
	Heartbleed	4	5	5
Patator				
	Benign	86,305	86,097	86,144
	FTP-Patator	1,585	1,588	1,588
	SSH-Patator	1,235	1,254	1,255
WebAttacks				
	Benign	33,569	33,163	33,466
	Web Attack Brute Force	291	35	122
	Web Attack XSS	2	127	114
	Web Attack Sql Injection	0	5	4

t-SNE Analysis. In order to analyze the class distribution obtained by the classifier, we visualize the test samples and their classes with the t-distributed Stochastic Neighbor Embedding (t-SNE) method. The t-SNE method is commonly used to reduce the dimension of the dataset. Specifically, it is useful to visualize the distribution of high-dimensional data as a two-dimensional scatter plot. Using the scatter plot obtained with the t-SNE method, we visually analyze the data distribution calculated by the classifier.

To visualize the classification results of binary classification, Fig. 4 shows the t-SNE plots of the results using the Infiltration dataset. The plot area has been enlarged to make it easy to see the notable areas. The blue points belong to the 'Benign' class (the majority class), and the red points belong to the 'Bot' class (the minority class). The depth of the color shows the probability of being classified into the color's class, with a deep color indicating a high probability.

Figure 4(a) illustrates the distribution of each class in the training dataset. As shown in the figure, there are quite a few attacked samples, and the samples are intricately distributed, with some majority and minority class samples overlapping. Figure 4(b) illustrates the distribution of each class in the test dataset. There are fewer red points (= minority class samples) compared to the training dataset. Figures 4(c)–(f) illustrate the classification results of the test dataset. Figure 4(c) illustrates the classification result by the Normal model. In the figure, the sample circled in red in (b) is misclassified as the majority class. Figure 4(d) is obtained by ADASYN. Compared to the Normal model, minority class samples are correctly classified. Figure 4(e) is obtained by the **UT-wPGD** classifier. Due to the small perturbation of our proposed method, the classification area of

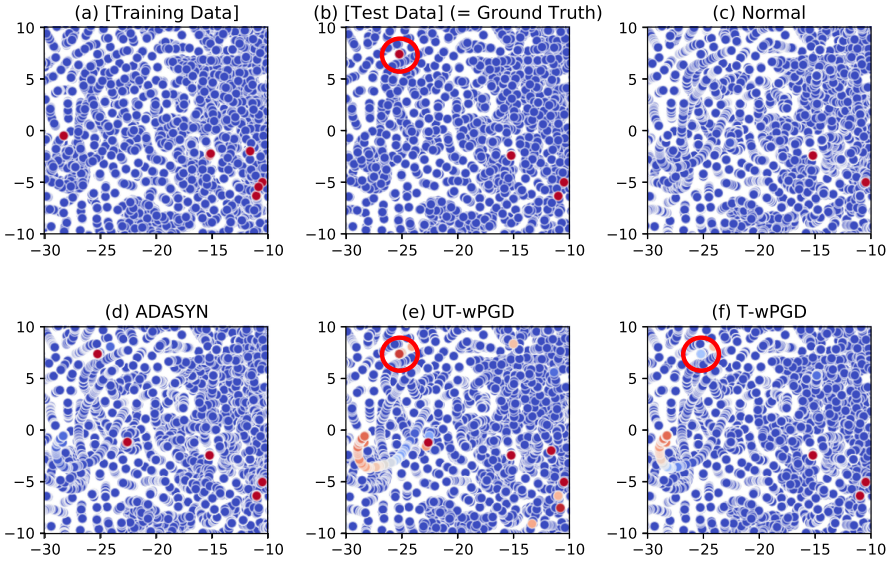

Fig. 4. The t-SNE plots of the classification results (Infiltration dataset). The blue points are the majority class samples, and the red ones are the minority class samples. (Color figure online)

the minority class is expanded. However, more majority class samples are misclassified as the minority class. The misclassified samples are distributed around the minority class samples in the training dataset. Therefore, the **UT-wPGD** classifier expands the classification area of the minority class, but it seems too large. Figure 4(f) is obtained by the **T-wPGD** classifier. Because the perturbation direction is targeted toward the other class, the classification area is not expanded compared to the **UT-wPGD** classifier. However, the sample circled in red is misclassified as the majority class, whereas the **UT-wPGD** classifier successfully classifies. From the results above, our proposed models **UT-wPGD** and **T-wPGD** successfully work for the binary classification. In particular, the **UT-wPGD** model expands more classification area than the **T-wPGD**. As mentioned in Sect. 4.2, untargeted adversarial training would cause *label leaking* effect [17]. We can observe this effect from the results, and the **UT-wPGD** model performs more effectively.

To visualize the classification results of multiclass classification, we select the WebAttack dataset. This dataset is difficult to classify, as shown in Table 5, because the samples of the Brute Force and XSS classes have similar features. Figure 5 shows the t-SNE plots of the results using the WebAttack dataset. There are four colors: blue, orange, green, and red. Different from Fig. 4, the depth of the face color is fixed. The blue points are the majority class, and the others are the minority classes. The green (resp. yellow or red) samples correspond to the Brute Force class (resp. XSS or Sql Injection). The number of red samples is quite small compared to the other classes. It should be noted that the green

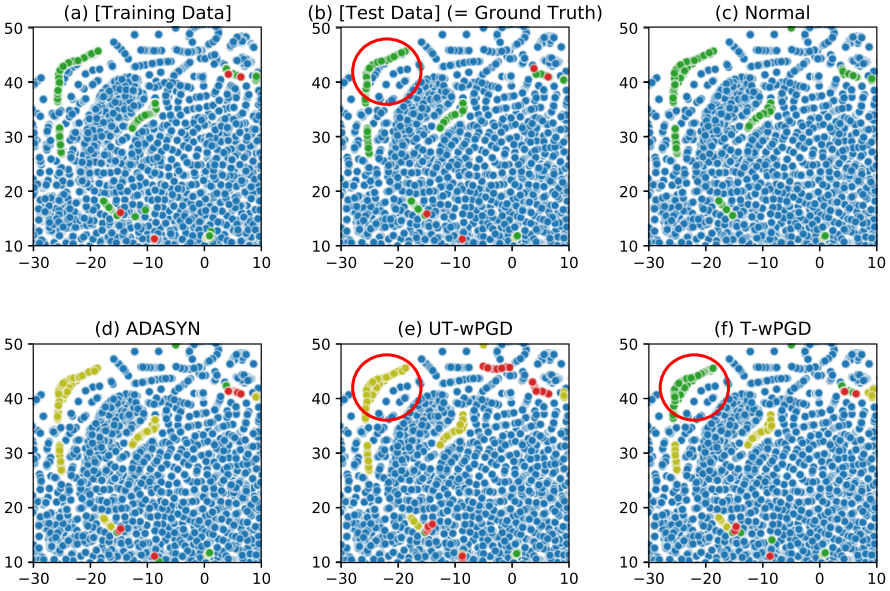

Fig. 5. The t-SNE plots of the classification results (WebAttack dataset). (Color figure online)

and yellow class samples are overlapped because their features are quite similar. In the figures, green samples are on top of yellow samples.

Similar to Fig. 4, Fig. 5(a) illustrates the distribution of the training dataset, and Fig. 5(b) illustrates the distribution of the test dataset. Figures 5(c)–(f) illustrate the classification results of the test dataset. Figure 5(c) illustrates the classification result by the Normal model, which fails to classify red class samples. Figure 5(d) is obtained by the ADASYN method. Compared to the Normal model, minority class samples are correctly classified, and it seems to have a similar distribution to the test dataset. Figure 5(e) is obtained by the **UT-wPGD** classifier. Although the classification areas of the minority classes are successfully expanded, some samples in the red circle are misclassified. Figure 5(f) is obtained by the **T-wPGD** classifier. Ideally, green and yellow samples should be distributed in the plot area. However, ADASYN and the **UT-wPGD** model could not classify green samples. In contrast, the **T-wPGD** classifier successfully classifies several green samples, as shown in the red circle in the figure.

From the discussion in this section, we find out the following results. Firstly, balancing class distribution in a minibatch successfully expands the classification area of the minority class. Secondly, our proposed method outperforms conventional over-sampling methods such as SMOTE and ADASYN. The SMOTE and ADASYN models can certainly contribute to improving the recall score of the minority class. However, those models might strongly misclassify the majority class samples in the neighborhood of the minority class. Compared to the SMOTE and ADASYN models, the **UT-wPGD** and **T-wPGD** expand the

classification area of the minority class, but the misclassified samples have low probability. Thirdly, the **T-wPGD** model tends to classify higher probability near the genuine minority-class samples than the **UT-wPGD** model. Although the targeted and untargeted perturbation have similar meanings in binary classification, they result in different classification areas. While the **UT-wPGD** model obtains the best results in the experiment, the **T-wPGD** model is more stable in classification.

6 Conclusion

In this paper, we propose an imbalanced training method leveraging an adversarial training algorithm, which is useful for security tasks such as an IDS. In order to realize robust learning for imbalanced datasets, we leverage weighted adversarial training that makes it possible to spread or shrink the classification area according to the weighting vector that is determined by the importance of each class. We perform several experiments with real-world imbalanced datasets. The experimental results demonstrate that our proposed method successfully enhances the robustness of the classifier for imbalanced training datasets, and effectively increases the classification performance for a security task. Furthermore, we visually analyze the classification area of our proposed method.

While our proposed method gives a significant contribution to imbalanced learning, some points are still remained to be further considered. In order to appropriately set the maximum perturbation size ϵ'_i for each class i, optimizing the function $g(x)$ is one of the most challenging problems, and this will be our future work.

References

1. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: a survey. IEEE Access **6**(AUGUST), 14410–14430 (2018)
2. Brodersen, K.H., Ong, C.S., Stephan, K.E., Buhmann, J.M.: The balanced accuracy and its posterior distribution. In: International Conference on Pattern Recognition (ICPR), pp. 3121–3124 (2010)
3. Cao, K., Wei, C., Gaidon, A., Aréchiga, N., Ma, T.: Learning imbalanced datasets with label-distribution-aware margin loss. In: Neural Information Processing Systems, pp. 1565–1576 (2019)
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
5. Douzas, G., Bação, F.: Effective data generation for imbalanced learning using conditional generative adversarial networks. Expert Syst. Appl. **91**, 464–471 (2018)
6. Elkan, C.: The foundations of cost-sensitive learning. In: International Joint Conference on Artificial Intelligence, pp. 973–978 (2001)
7. Forman, G.: An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. **3**(Mar), 1289–1305 (2003)
8. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (ICLR) (2015)

9. Goswami, G., Ratha, N., Agarwal, A., Singh, R., Vatsa, M.: Unravelling robustness of deep learning based face recognition against adversarial attacks. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, pp. 6829–6836 (2018)
10. Hayat, M., Khan, S., Zamir, W., Shen, J., Shao, L.: Max-margin class imbalanced learning with gaussian affinity (2019). http://arxiv.org/abs/1901.07711v1
11. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322–1328 (2008)
12. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009)
13. Kaur, H., Pannu, H.S., Malhi, A.K.: A systematic review on imbalanced data challenges in machine learning - applications and solutions. ACM Comput. Surv. **52**(4), 79:1–79:36 (2019)
14. Khan, S.H., Hayat, M., Bennamoun, M., Sohel, F., Togneri, R.: Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Trans. Neural Netw. Learn. Syst. **29**(8), 3573–3587 (2017)
15. Kim, J., Jeong, J., Shin, J.: M2m: imbalanced classification via major-to-minor translation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13893–13902 (2020)
16. Kukar, M., Kononenko, I.: Cost-sensitive learning with neural networks. In: Proceedings of the 13th European Conference on Artificial Intelligence, pp. 445–449 (1998)
17. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. In: International Conference on Learning Representations (ICLR) (2017)
18. Lee, J., Park, K.: Gan-based imbalanced data intrusion detection system. Personal Ubiquit. Comput. **25**(1), 121–128 (2021)
19. Maseer, Z.K., Yusof, R., Bahaman, N., Mostafa, S.A., Foozy, C.F.M.: Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. IEEE Access **9**, 22351–22370 (2021)
20. Montahaei, E., Ghorbani, M., Baghshah, M.S., Rabiee, H.R.: Adversarial classifier for imbalanced problems (2018). http://arxiv.org/abs/1811.08812v1
21. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: International Conference on Information Systems Security and Privacy (ICISSP), pp. 108–116 (2018)
22. Szegedy, C., et al.: Intriguing properties of neural networks. In: International Conference on Learning Representations (ICLR) (2014)
23. Terzi, M., Susto, G.A., Chaudhari, P.: Directional adversarial training for cost sensitive deep learning classification applications. Eng. Appl. Artif. Intell. **91**, 103550 (2020)
24. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: Ensemble adversarial training: attacks and defenses. In: International Conference on Learning Representations (ICLR) (2018)
25. Vivek, B.S., Mopuri, K.R., Babu, R.V.: Gray-box adversarial training. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 213–228 (2018)