# Handling Imbalanced Data in Intrusion Detection Systems using Generative Adversarial Networks

Ly Vu, Quang Uy Nguyen
Le Quy Don Technical University, Hanoi, Vietnam
Correspondence: Ly Vu, vu.ly@lqdtu.edu.vn

Communication: received 24 October 2019, revised 6 February 2020, accepted 30 March 2020
Digital Object Identifier: 10.32913/mic-ict-research.v2020.n1.894

**Abstract: Machine learning-based intrusion detection has become more popular in the research community thanks to its capability in discovering unknown attacks. To develop a good detection model for an intrusion detection system (IDS) using machine learning, a great number of attack and normal data samples are required in the learning process. While normal data can be relatively easy to collect, attack data is much rarer and harder to gather. Subsequently, IDS datasets are often dominated by normal data and machine learning models trained on those imbalanced datasets are ineffective in detecting attacks. In this paper, we propose a novel solution to this problem by using generative adversarial networks to generate synthesized attack data for IDS. The synthesized attacks are merged with the original data to form the augmented dataset. Three popular machine learning techniques are trained on the augmented dataset. The experiments conducted on the three common IDS datasets and one our own dataset show that machine learning algorithms achieve better performance when trained on the augmented dataset of the generative adversarial networks compared to those trained on the original dataset and other sampling techniques. The visualization technique was also used to analyze the properties of the synthesized data of the generative adversarial networks and the others.**

**Keywords:** *Generative adversarial networks, intrusion detection system, synthesized attack, imbalanced dataset, sampling technique.*

## I. Introduction

Communication networks and information systems have become a very important factor in almost every facet of our daily lives [1, 2]. The rapid development of Internet services and communication networks has revolutionized our perspective of the world. Nevertheless, this also resulted in the information systems being very vulnerable to one or more types of cyber attacks. The security of communication networks and information systems is therefore an increasing concern for cyber security officers, network administrators and end-users. Among several approaches to protect information systems, Intrusion Detection System (IDS) plays a crucial role for early detecting of security violation [1].

IDS monitors the network traffic to find any abnormal activity. There are three popular methods for analyzing the network traffic to detect intrusive behaviors: statistical-based, machine learning-based, and knowledge-based methods [3]. Among these, machine learning-based methods have received a great attention and achieved remarkable success recently [1, 4–15] even for encrypted network traffic [16, 17]. To train a good detection model for IDS in the real world, a considerable number of attack and normal data samples are required to be gathered. Compared to normal data, intrusive data is more scarce and more expensive to collect. Thus, the collected network traffic datasets for intrusion detection are often imbalanced. Subsequently, the ability to handle imbalanced data is essential for machine learning algorithms to achieve a good accuracy in intrusion detection.

The imbalance of class samples can reduce the performance of a machine learning based IDS [18]. In this paper, we propose a novel approach to tackle the imbalanced problem of IDS datasets by using generative adversarial networks (GANs) to generate synthesized attack data. The synthesized data is then combined with the original data to form the augmented training data. Three conventional classifiers including decision tree (DT), random forest (RF), and support vector machine (SVM) are trained on the augmented dataset. The experiments were conducted on three popular IDS datasets, i.e., NSL-KDD [19], UNSW-NB15 [20], CICIDS2017 [21], and one our own dataset (i.e., RAWDATA). The results showed that machine learning algorithms achieve better performance when trained on the augmented dataset of GANs compared to those trained on the original dataset and the datasets generated by some popular sampling techniques.

The main contribution of this paper is the demonstration of the usefulness of GANs in addressing the imbalance problem in IDS datasets. Compared to [22] where we originally proposed this technique, here we present a better version of using GANs for generating synthesized data and examine them in detail on a wide range of benchmarks. The rest of the paper is organized as follows. Section II briefly reviews the previous works in applying machine learning to IDS and the methods for dealing with imbalanced datasets. Section III presents the fundamental background of our paper. The proposed method is then described in Section IV. The tested datasets and experimental settings are provided in Section V. Section VI presents experimental results and the analysis. The conclusions and future work are discussed in Section VII.

## II. Related Work

This section presents a brief review of using machine learning in IDS and the techniques for addressing imbalanced data.

### 1. Machine learning for Intrusion Detection

Machine learning for intrusion detection has received a considerable attention in the research community [1, 4–9, 11, 12, 23, 24]. For a comprehensive review and analysis of different machine learning techniques in IDS, the readers are recommended to see [25]. In this paper, we shortly summarize recent and related research to our topic. Usually, machine learning is applied to intrusion detection to automatically build a detection model based on the training dataset. Machine learning techniques for IDS can be divided into two categories: a single and a hybrid method. The single method attempts to use only one machine learning technique to find the model which is then used to recognize whether the incoming access is a normal access or an attack [1]. The popular single learning algorithms used in the IDS include SVM, K-mean, Artificial Neural Network (ANN), RF, and DT [9–12, 26].

The hybrid method aims to combine two or more learning techniques to enhance the performance of the systems. There are several ways in which different algorithms can be hybridized. The first method is by cascading different classifiers. For example, Hussain et al. [4] proposed a two stage hybrid method using SVM as an anomaly detection in the first stage, and ANN as a misuse detection in the second stage to enhance accuracy of IDS. Aburomman et al. [23] proposed an ensemble classifiers by using the weighted majority algorithm (WMA) approach of particle swarm optimization with the SVM and K-Nearest Neighbor algorithm to enhance the accuracy of IDS. Aburomman et al. [24] also compared the effectiveness of various ensemble techniques (Boosting, Voting, and Stacking) for IDS. Other methods are based on re-sampling data samples and then taking a majority vote of the resulting weak learners [27]. Several other approaches [28, 29] aimed to collect datasets to improve the accuracy of IDS.

Recently, deep learning has been applied to improve the accuracy of IDS [13, 14, 30, 31]. Malaiya et al. [13] used Convolutional Neural Network (CNN) for detecting abnormal behaviours in the network. They used the 1D-feature of the network traffic in IDS datasets as the input of CNN to enhance the accuracy of intrusion detection. Salama et al. [30] proposed a method for the anomaly intrusion detection scheme using Restricted Bolzman Machine (RBM)-based Deep Believe Network (DBN). In their work, DBN is used as a feature reduction method and SVM is utilized as a classifier. Kim et al. [31] showed that Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM) were effective for IDS problems. Kwon et al. [14] compared the accuracy of some popular deep learning models including RBN, RNN, AutoEncoder, and DBN in intrusion detection. They also proposed a Fully Connected Model (FCM) for intrusion detection. In FCM, the number of neurons in each layer is equal to the number of data features. Rodda [15] showed that the multi-layer perceptron is better than radial basis function networks for designing network intrusion detection system. Moreover, Benlenko [32] presented generative adversarial artificial neural networks to detect security intrusions in the large-scale networks of cyber Internet of Thing devices.

Overall, deep learning has often been used to extract meaningful features from intrusion data sets. In this paper, we propose a new application of deep learning to IDS. More specifically, we use a generative model (i.e., GAN) to generate synthesized attacks to handle the imbalanced problem of IDS datasets. The detailed description of our method will be presented in Section IV.

### 2. Handling Imbalanced Data in Machine Learning

Imbalanced data typically refers to a problem where the number of observations belonging to one class is significantly lower than those belonging to the other classes [33]. In this situation, the predictive model developed using conventional machine learning algorithms could be biased and inaccurate [33]. The reason is that machine learning algorithms are usually designed to improve the accuracy by reducing the error. Thus, they do not take into account the class distribution/proportion or the balance of classes. In order to improve the accuracy of machine learning in imbalanced datasets, several techniques have been proposed [33].

Perhaps, the most popular technique for dealing with imbalanced data is sampling. Sampling techniques usually consider the ratio of sample classes. Random undersampling [34] aims to downsize the majority classes by removing observations until the dataset is balanced. Random oversampling [34] decreases the level of class imbalance by copying the minority class until the classes have equal samples. The downsize of oversampling is the increase in the risk of overfitting [34]. Synthetic Minority Oversampling Technique (SMOTE) [35] generates the samples of the minor class by extrapolating and interpolating minor samples from the neighborhood ones. An extension of SMOTE is SMOTE-SVM [36] that synthesizes samples located in the borderline between classes by only generating the samples for the support vectors of a SVM model trained on the original dataset.

BalanceCascade [27] combines a sampling technique with a classifier to discover the distribution of the majority and minority class. This method develops an ensemble model of classifiers to systematically select the major samples to remove. A number of sub-datasets of balanced rate between minority and majority are created by undersampling the major class. Next, an ensemble of classifiers are trained on these sub-datasets. Finally, the samples in the major class will be removed if they are classified correctly by the ensemble model.

Some other techniques use the distance between data samples to remove the noisy or borderline samples of each class. Tomek link [37] removes samples from the major class that are close to the minor region in order to return a dataset that presents a better separation between the two classes. TomekSMOTE is a hybrid technique that uses SMOTE to oversample the minor class and apply Tomek to remove the noisy samples generated by SMOTE.

Although TomekSMOTE and SMOTE-SVM have been popularly used and their effectiveness has been well-evidenced [33, 37, 38], the shortcoming of these methods is that the distribution of the original data can be lost. In other words, the generated data samples may have a very different distribution from the original data. In this paper, we propose a method for oversampling data by using a generative deep learning model. We hypothesize that the synthesized data by the generative model will preserve the distribution of the original data better than the traditional sampling techniques. Subsequently, the augmented dataset of deep learning method will help to improve the accuracy of classification algorithms. In [39], the author also synthesized the data samples to improve the quality of IDS datasets by using a variety of Generative Adversarial Network (GAN). Their model is called SynGAN. which uses an extension of GAN, i.e., Wasserstein GAN (WGAN) [40] to synthesize data samples. The difference of SynGAN compared with our model is that SynGAN is operated in an unsupervised manner, and thus, it does not use the label information for training. Conversely, our proposed method, i.e., ACGAN-SVM uses the label information during the training process, thereby improving the accuracy of IDS.

## III. Background

This section describes in detail Generative Adversarial Network and its extension Auxiliary Classifier Generative Adversarial Network.

### 1. Generative Adversarial Network

Generative Adversarial Network (GAN) [41] is a recent deep learning method for generating synthesized data. GAN composes of two neuron networks: a Generator (G) and a Discriminator (D). The input of the generator is noise and it outputs a generated sample. The input of the discriminator includes two sources: a generated sample of the generator and a training data sample. The discriminator attempts to differentiate between a real data sample and a fake sample generated by the generator. These two networks are trained continuously, where the generator learns to generate more realistic samples, and the discriminator aims to separate the generated data from the real data. Usually, the competition between two networks will result in the generated samples being difficult to distinguish from the real samples.

Let $z$ (random noise) be the input of G and its output is a synthesized sample $X_{fake} = G(z)$. D takes as input either a real $(x)$ or a fake sample $(G(z))$, and its output is a value that presents the probability of being real of the input sample. In other words, D is trained to increase the probability of the real data and to decrease the probability of the generated data by maximizing (1). Conversely, G is trained to increase the probability of the fake data being rated as the real data by minimizing the second term in this equation.

$$L = E[\log D(x)] + E[\log(1 - D(G(z)))]. \qquad (1)$$

One appealing property of GAN is that it is a fully unsupervised approach. Therefore, we can train a GAN in an unsupervised manner and then use its generator and discriminator as feature extractors for supervised tasks. However, GAN is more unstable to train because we have to train two networks in an opposite way from a single back-propagation. Subsequently, the resulting generator may produce nonsensical outputs.

**Algorithm 1:** Algorithm of training G.

```
1 Inputs: Random noise $z$, Class label $c$ .
2 Outputs: Trained generator $G$.
3 begin
4     Set input value of $G$ is $z$ concatenate $c$;
5     Train $G$ by maximizing $(L_C - L_S)$;
6     return $G$.
7 end
```

**Algorithm 2:** Algorithm of training D.

```
1 Inputs: Real data sample from original dataset $X$,
    Class label $c$.
2 Outputs: Trained discriminator $D$.
3 begin
4     Generate random noise $z$;
5     $X' = G(z, c)$;
6     Set input value of $D$ is $X$ or $X'$ concatenate $c$ ;
7     Train $D$ by maximizing $(L_C + L_S)$;
8     return $D$.
9 end
```

### 2. Auxiliary Classifier Generative Adversarial Network

Auxiliary Classifier Generative Adversarial Network (ACGAN) [42] is an extension of GAN by using the class label in the training process. ACGAN also includes two neural networks operating in a contrary way: a Generator ($G$) and a Discriminator ($D$). The input of $G$ in ACGAN includes a random noise $z$ and a class label $c$ instead of only random noise $z$ as in the GAN model. Therefore, the synthesized sample of $G$ in ACGAN is $X_{fake} = G(c, z)$, instead of $X_{fake} = G(z)$. In other words, ACGAN can generate data samples for a desired class label. The objective function of ACGAN has two parts (in (2) and (3)): the log-likelihood of the correct data, $L_S$, and the log-likelihood of the correct class, $L_C$. $D$ is trained to maximize $L_C + L_S$ and $G$ is trained to maximize $L_C - L_S$.

$$L_S = E[\log D(x)] + E[\log(1 - D(G(z)))]. \quad (2)$$

$$L_C = E[\log D(x, c)] + E[\log(1 - D(G(z, c)))]. \quad (3)$$

Algorithms 1 and 2 describe the process of training $G$ and $D$ networks in the ACGAN model. $G$ is trained to generate fake samples that are similar to real samples. Conversely, the objective of training $D$ is to increase the difference between a real sample and a fake sample of the same class. These two networks are trained simultaneously until achieving a Nash equilibrium [43].

Similar to GAN, the training process in ACGAN using gradient descent algorithm may not be converged. Since two cost functions are updated independently, there is no guarantee to enhance the training error for both neural networks in ACGAN [43]. In this paper, we investigated various structures of $G$ and $D$ to determine the structure that achieve good performance when trained on the network traffic datasets.

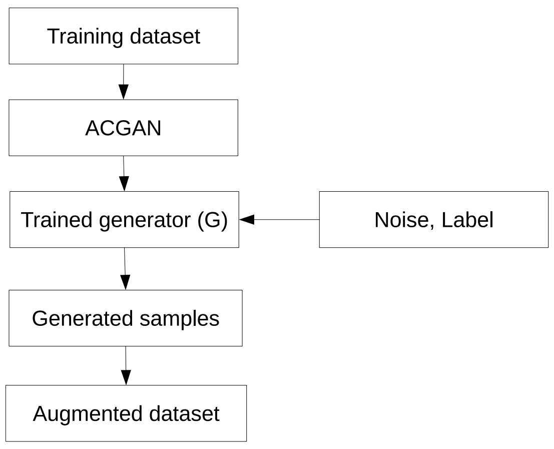

Figure 1. Process of using ACGAN to create the augmented datatset.

## IV. Methods

This section describes two approaches for generating synthesized attacks in IDS. The first approach uses an ACGAN network to generate the synthesized data. The second approach uses a SVM model to remove the noisy samples that is generated when using the ACGAN model.

### 1. Generating Synthesized Attacks using ACGAN

The flow of using ACGAN to create augmented dataset is described in Fig. 1. First, the original dataset is divided into two parts: a training set and a testing set. The ACGAN model is trained on the training set. After training, the generator ($G$) is used to generate synthesized data samples for the minor (attack) classes. The synthesized samples are then combined with the training dataset to form the augmented dataset. Three supervised classification algorithms (i.e., SVM, DT, and RF) are then trained on the augmented dataset. Finally, the supervised classification algorithms (after being trained on the augmented dataset) are tested on the testing set.

### 2. Generating Synthesized Attacks using ACGAN-SVM

The second method for generating artificial data is ACGAN-SVM. ACGAN-SVM attempts to produce samples that are near the borderline area defined by the SVM model. However, unlike SMOTE-SVM that produces data samples by extrapolating or interpolating from the original samples [36], ACGAN-SVM uses a generative model to generate new samples. We hypothesize that the samples generated by ACGAN-SVM will preserve the underlying distribution of the original dataset better than SMOTE-SVM. Subsequently, the augmented dataset of ACGAN-SVM will improve the performance of classification algorithms.

**Algorithm 3:** ACGAN-SVM algorithm for over-sampling.

```
1  Inputs: original training set $X$, generated data
     samples $X'$, number of nearest neighbors $m$, Euclid
     distance between vector $x$ and vector $y$ d($x$,$y$).
2  Outputs: new sampling set $X_{new}$.
3  begin
4      Training ACGAN on $X$ to have trained
         Generator $G$;
5      Set $X'$ contains minority samples generated by
         $G$ network;
6      Train SVM model on $X$ to have the set of
         support vectors $SVs$;
7      foreach $sv_i \in SVs$ do
8          Compute $m$ nearest neighbors in $X$;
9          Compute $d_i$ that is average of Euclid
             distance from $m$ nearest neighbors to $sv_i$;
10     end
11     foreach $x_j \in X'$ do
12         if $d(x_j, sv_i) \leq d_i$ then
13             $X_{new} = X \cup \{x_j\}$;
14         end
15     end
16     return $X_{new}$.
17 end
```

Algorithm 3 presents the detailed description of using ACGAN-SVM for generating synthesized data. The technique is divided into two main phases, i.e., generation and selection. In the generation phase, the ACGAN network is trained on the training dataset $X$. After that, the generator network ($G$) of ACGAN is used to generate synthesized samples $X'$. In the selection phase, the SVM model is trained on the training dataset $X$ and the set of support vectors of this model is called $SV_s$. For each support vector $sv_i \subset SV_s$, we calculate the average Euclidean distance $d_i$ of $m$ nearest neighbor samples of $sv_i$ in $X$ to $sv_i$. If a generated sample $x_j \subset X'$ has the distance to $sv_i$ smaller than $d_i$, this sample is kept. Conversely, if the distance from $x_i$ to $sv_i$ is greater than $d_i$, the sample is removed. The algorithm will stop when the augmented dataset is balanced for every class. The augmented dataset is then used to train three classification algorithms as in ACGAN.

## V. Experimental Settings

This section presents the datasets used in the experiments and the parameter's setting for the tested algorithms.

### 1. Datasets

In order to test the effectiveness of the proposed method we used three well-known network intrusion detection datasets, i.e., NSL-KDD, UNSW-NB15, CICIDS2017 and one our own dataset, i.e., RAWDATA.

**NSL-KDD** is an IDS dataset [19] which is used to solve some intrinsic drawbacks of the KDD'99 dataset. The NSL-KDD dataset contains 148517 records divided into the training set (125973 data samples) and the testing set (22544 data samples). Each sample has 41 features and is labeled either as a type of attack or normal data. The training set contains 24 attack types, and the testing set includes 14 more types of attack. The simulated attack samples belong to one of four categories: DoS, R2L, U2R, and Probing.

**UNSW-NB15** is created by utilizing the synthetic environment in the Cyber Range Lab of the Australian Center of Cyber Security (ACCS) [20]. The number of records in the training set and the testing set are 175341 and 82332, respectively. The data samples are labeled as normal or one of nine attack types. Nine categories of attacks includes Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. Each data sample has 49 features which were generated by using Argus, Bro-IDS tools and twelve other algorithms to analyze characteristics of network packets.

**CICIDS2017** is an IDS dataset developed by Canadian Institute for Cybersecurity. Data samples in CICIDS2017 include benign samples and some recent common attacks such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Port Scan, Infiltration, Botnet and [21]. We prepossessed data by dropping some attributes that do not represent the characteristic of a network flow. The removed attributes include FlowID, Source IP address, Destination IP address and time stamp. The resulting network flow is represented by 78 attributes. We also removed Heartbleed and Infiltration attacks from the dataset since they have too few samples for any algorithm to learn[1]. The final dataset used in our experiments has four categories of attacks: DoS, Brute Force, Web Attack, and Botnet and is randomly divided into the training dataset (266028 records) and the testing dataset (186213 records).

[1]Infiltration and Heartbleed have only 3 and 1 samples, respectively.

**RAWDATA** is an IDS dataset which we collected at our Computer Network Lab at Le Quy Don Technical University. First, we set up a network environment and captured three kinds of traffic including the normal traffic, the Structured Query Language (SQL) injection traffic, and the Cross-Site Scripting (XSS) traffic using the Winpcap library [44]. We used the Sqlmap tool [45] to simulate the SQL injection traffic and the XSS attack traffic was generated manually. Then, the collected packets were extracted by the Scapy library [46] in the Python language. Each packet was extracted by 500[2] raw bytes as features. Finally, we labeled the normal traffic as 0, the SQL injection traffic as 1, and the XSS traffic as 2. The final dataset was randomly divided into the training set (80% number of samples) and a testing set (20% number of samples). The processed dataset is available for download[3].

## 2. Parameters Settings

Since it is often difficult to guarantee a good convergence when training ACGAN, we have conducted a number of experiments to tune the hyper-parameters of ACGAN. The best values of hyper-parameters in ACGAN are calibrated and presented in Table I. For ACGAN-SVM, we used the same structure as ACGAN in Table I. Moreover, the SVM model with the rbf kernel and gama parameter of 0.01 was trained on the original data. For both ACGAN and ACGAN-SVM, we used the ReLu activation function for all hidden layers except the last layer where we used the Sigmoid activation function. The Adam optimization [47] with the learning rate of $10^{-3}$ was used to train $G$ and $D$ networks in ACGAN and ACGAN-SVM.

TABLE I
PARAMETERS SELECTION FOR $G$ AND $D$ NETWORKS IN ACGAN AND ACGAN-SVM.

| Dataset | Number of layers | Number of Neurons | Batch size |
|---|---|---|---|
| NSL-KDD | 5 | 160 | 128 |
| UNSW-NB15 | 5 | 160 | 64 |
| CICIDS2017 | 5 | 160 | 100 |
| RAWDATA | 5 | 160 | 50 |

After using ACGAN and ACGAN-SVM to generate the synthesized data, three popular classification algorithms, i.e., SVM, DT, and RF are trained on the augmented datasets. We used the implementation of these algorithms in a popular machine learning packet in Python, Scikit learn [48]. In order to lessen the impact of experimental parameters to the performance of the classifiers, we used the grid search technique for each algorithm. The range of values for the important parameters tuned by the grid search technique are presented in Table II.

[2]This number is a popular size of packet for our data chosen by a statistical method.
[3]https://github.com/vuthily/data/rawdata.

TABLE II
PARAMETER'S RANGE OF THE GRID SEARCH FOR CLASSIFIERS.

| Classifiers | Parameters |
|---|---|
| SVM | $kernel = rbf\,; gama = 0.001, 0.01, 0.1, 1.0$ |
| DT | $max - depth = 5, 6, 7, 8, 9, 10, 50, 100$ |
| RF | $n - estimators = 20, 40, 80, 150$ |

For each dataset, we conducted two sets of experiments. In the first set, the traffic datasets were adapted to form two-class classification problems. In other words, the class label has two values: "Normal"("Benign") or "Attack". In the second set, the datasets were processed to form multi-class classification problems. In this case, NSL-KDD and CICIDS2017 datasets have five classes, UNSW-NB15 has ten classes, and RAWDATA has three classes. The number of samples in each class for binary and multiclass problems are described in Table III. It can be seen that these datasets are mostly balanced in the binary classification problem. However, in the multiclass classification problem, both datasets are highly imbalanced. The number of samples of some classes such as U2L in NSL-KDD and Worms, Shellcode in UNSW-NB15, and Brute Force, Botnet in CICIDS2017, SQL injection in RAWDATA are much less than the number of samples of normal class and some other attack classes.

We compared the performance of the classification algorithms when they are trained on the augmented datasets of WGAN [40], ACGAN [42], and ACGAN-SVM with the version trained on the original data and the synthesized data of three traditional sampling approaches: SMOTE-SVM [36], BalanceCasscade [27], TomekSMOTE [37]. The source code of all tested methods are available for download[4]. All techniques were implemented in Python, Scikit-learn machine learning library [48] and Tensorflow deep learning framework [49]. Moreover, the same computing platform (Operating system: Ubuntu 16.04 (64 bit), Intel(R) Core(TM) i5-5200U CPU, 2 cores and 4GB RAM memory) was used in every experiment in this paper.

## 3. Evaluation Metrics

Three popular performance metrics in a classification problem were used to measure the effectiveness of our method. The reported metrics include precision score, recall score, and F1 score [50]. Equation (4) and (5) present precision and recall score for one class. The final values of these metrics are the average over all classes.

[4]https://github.com/vuthily/acgansvm.

TABLE III
NUMBER OF CLASS SAMPLES IN EACH TRAINING DATASET.

| NSL-KDD | | UNSW-NB15 | | CICIDS2017 | | RAWDATA | |
|---|---|---|---|---|---|---|---|
| Classes | Number | Classes | Number | Classes | Number | Classes | Number |
| Normal | 67373 | Normal | 37000 | Benign | 219110 | Benign | 6503 |
| Attack | 58630 | Attack | 45332 | Attack | 46928 | Attack | 1528 |
| DoS | 45927 | Generic | 18871 | DoS | 29447 | SQL injection | 526 |
| U2L | 52 | Exploits | 11132 | Port Scan | 15893 | XSS | 1002 |
| R2L | 995 | Fuzzers | 6062 | Brute Force | 1392 | | |
| Probing | 11656 | DoS | 4089 | Botnet | 196 | | |
| | | Reconna-issance | 3496 | | | | |
| | | Analysis | 677 | | | | |
| | | Backdoor | 583 | | | | |
| | | Shellcode | 378 | | | | |
| | | Worms | 44 | | | | |

TABLE IV
RESULT OF DT, RF, AND SVM ON TWO-CLASS CLASSIFICATION FOR NSL-KDD, UNSW-NB15, CICIDS2017, AND RAWDATA DATASETS.

| **Algorithm** | **Augmented datasets** | **NSL-KDD** | | | **UNSW-NB15** | | | **CICIDS2017** | | | **RAWDATA** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** |
| **DT** | ORIGINAL | 0.92 | 0.90 | 0.91 | **0.84** | 0.74 | 0.80 | 0.95 | 0.95 | 0.94 | 0.97 | 0.95 | 0.96 |
| | SOMTE-SVM | 0.91 | 0.90 | 0.90 | 0.82 | 0.72 | 0.81 | 0.93 | 0.89 | 0.90 | 0.97 | 0.98 | 0.97 |
| | BalanceCascade | 0.92 | 0.91 | **0.92** | 0.81 | 0.70 | 0.82 | 0.94 | 0.92 | 0.93 | 0.96 | 0.96 | 0.95 |
| | TomekSMOTE | 0.92 | 0.92 | **0.92** | 0.81 | 0.74 | 0.83 | 0.94 | 0.92 | 0.93 | 0.97 | 0.95 | 0.96 |
| | WGAN | 0.92 | 0.90 | 0.90 | 0.80 | 0.78 | 0.76 | 0.93 | 0.92 | 0.92 | 0.97 | 0.98 | 0.97 |
| | ACGAN | 0.91 | 0.91 | 0.91 | 0.81 | 0.81 | 0.82 | 0.95 | 0.96 | 0.95 | 0.98 | 0.98 | 0.98 |
| | ACGAN-SVM | **0.93** | **0.93** | **0.92** | 0.82 | **0.84** | **0.84** | **0.97** | **0.98** | **0.96** | **0.99** | **0.99** | **0.99** |
| **RF** | ORIGINAL | **0.87** | 0.83 | 0.83 | 0.91 | 0.88 | 0.88 | 0.98 | 0.98 | 0.98 | 0.97 | 0.97 | 0.97 |
| | SMOTE-SVM | **0.87** | 0.83 | 0.83 | 0.88 | 0.82 | 0.83 | **0.99** | **0.99** | **0.99** | 0.98 | 0.97 | 0.98 |
| | BalanceCascade | **0.87** | 0.83 | 0.83 | 0.91 | 0.89 | **0.89** | **0.99** | **0.99** | **0.99** | 0.97 | 0.97 | 0.97 |
| | TomekSMOTE | 0.86 | 0.84 | 0.84 | 0.91 | 0.90 | **0.89** | **0.99** | **0.99** | **0.99** | 0.97 | 0.96 | 0.97 |
| | WGAN | 0.85 | 0.86 | 0.84 | 0.89 | 0.87 | 0.88 | 0.97 | 0.95 | 0.95 | 0.98 | 0.98 | 0.98 |
| | ACGAN | **0.87** | 0.84 | 0.84 | 0.91 | 0.88 | **0.89** | **0.99** | **0.99** | 0.98 | **0.99** | 0.97 | 0.98 |
| | ACGAN-SVM | **0.87** | **0.86** | **0.85** | **0.92** | **0.90** | **0.89** | **0.99** | **0.99** | **0.99** | **0.99** | **0.99** | **0.99** |
| **SVM** | ORIGINAL | **0.85** | 0.82 | 0.83 | **0.87** | **0.87** | 0.87 | 0.96 | 0.96 | 0.96 | 0.96 | 0.95 | 0.96 |
| | SMOTE-SVM | **0.85** | **0.83** | 0.83 | 0.84 | 0.76 | 0.77 | 0.93 | 0.91 | 0.91 | 0.97 | 0.97 | 0.97 |
| | BalanceCascade | **0.85** | **0.83** | 0.83 | 0.84 | 0.76 | 0.76 | 0.92 | 0.90 | 0.91 | 0.97 | 0.96 | 0.97 |
| | TomekSMOTE | **0.85** | 0.82 | 0.83 | 0.85 | 0.77 | 0.78 | 0.92 | 0.90 | 0.91 | 0. | 0. | 0. |
| | WGAN | **0.86** | 0.84 | **0.85** | 0.85 | 0.83 | 0.85 | **0.98** | 0.96 | 0.96 | 0.97 | 0.98 | 0.97 |
| | ACGAN | 0.85 | 0.82 | 0.83 | **0.87** | 0.85 | 0.87 | 0.97 | **0.98** | 0.97 | 0.97 | **0.99** | 0.97 |
| | ACGAN-SVM | 0.85 | **0.83** | 0.84 | **0.87** | 0.86 | **0.88** | **0.98** | 0.97 | **0.98** | **0.98** | 0.97 | **0.98** |

$$Precision = \frac{TP}{TP + FP}. \tag{4}$$

$$Recall = \frac{TP}{TP + FN}. \tag{5}$$

In (4) and (5), *TP* and *FP* are the number of correct and incorrect predicted samples for class *i*, respectively, and *FN* is the number of incorrect predicted samples of the rest of the classes. Although, precision and recall are very intuitive and easy to implement, they make no distinction between classes. Therefore, they are not suitable to measure a classifier performance in imbalanced datasets.

F1-score (in (6)) aims to overcome the limitation of precision and recall score. F1-score is calculated as the mean [50] of precision and recall. F1-score is often considered as a reliable metric to evaluate the performance of classification algorithms in imbalanced datasets. Therefore, we will use this metric for comparing various techniques in this paper. Precision and recall are used as reference only.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{6}$$

## VI. RESULTS AND DISCUSSION

This section compares the accuracy of SVM, DT, and RF when they are trained on the augmented datasets of WGAN, ACGAN, and ACGAN-SVM with those trained on the original dataset and the datasets generated by three traditional sampling approaches: SMOTE-SVM [36], BalanceCascade [27], TomekSMOTE [37]. After that, the computational time for generating the augmented dataset of the sampling approaches is analyzed. Finally, we analyze the quality of the synthesized data and visualize the borderline samples of ACGAN, ACGAN-SVM, and the other oversampling techniques.

### 1. Accuracy of classification algorithms with synthesized attacks

The accuracy of classification algorithms on the binary and muticlass classification problems is presented in this subsection. Table IV presents the precision, recall, and F1-score of three classifiers on the binary classification problem. In this table and Table V, the best value in each configuration is printed bold face. It can be observed that all techniques for handling imbalanced data only slightly improve the accuracy of classifiers. The F1-score of SVM, DT and RF when trained on the augmented datasets of ACGAN, ACGAN-SVM and three traditional approaches is only slightly greater than that trained on the original version. The reason could be that, on the binary classification problem, the training data is mostly balanced. Therefore, using techniques for addressing imbalanced data did not help classification algorithms achieve a significant improvement.

Among the tested sampling techniques, we can see that ACGAN-SVM outperforms all others. The F1-score of the classifiers trained on the datasets of ACGAN-SVM is often higher than those trained on the datasets of the others. Moreover, the margin of the improvement of ACGAN-SVM over the original version is always greater than the margin of the improvement of the other sampling techniques. For example, using the augmented dataset of ACGAN-SVM for training, the F1-scores of DT, RF, and SVM are increased from 0.80, 0.88, and 0.87 to 0.84, 0.89, and 0.88 on UNSW-NB15 compared to using the original datatset. Another deep neural network based technique, i.e., WGAN, also provides the augmented dataset with the high accuracy of classifiers. However, it is still lower than those of the ACGAN based models. The reason is that training WGAN does not use the label of data as training the ACGAN models. For three traditional sampling techniques, the table shows that TomekSMOTE is often better than BalanceCascade and SOMTE-SVM. However, the accuracy of the classifiers trained on the datasets of TomekSMOTE is mostly equal to the accuracy of the classifiers trained on the original data.

Table V presents the F1-score, precision and recall of DT, RF, and SVM on the multiclass classification problem. It can be seen that the accuracy of the classifiers is improved considerably when trained on the datasets generated by all sampling techniques. Among three traditional sampling techniques, TomekSMOTE still achieves better results than SMOTE-SVM and BalanceCascade. Specifically, in UNSW-NB15, the F1-score of the classifiers that trained on the datasets of TomekSMOTE is often much better than that trained on the original dataset. Specifically, F1-scores of DT, RF, and SVM are improved from 0.42, 0.41, and 0.41 to 0.60, 0.72, and 0.60 when trained with the augmented datasets of TomekSMOTE compared to those trained on the original dataset. On three other datasets, i.e., NSL-KDD, CICIDS2017, and RAWDATA, the F1 scores of classifiers trained on the augmented dataset generated by TomekSMOTE are mostly equal to those trained on the original data.

The most impressive results in Table V are obtained by our proposed methods. The table shows that ACGAN and ACGAN-SVM are often considerably better than other techniques. Compared to training on the original dataset, the F1-scores of DT, RF, and SVM on UNSW-NB15 with ACGAN are increased from 0.42, 0.41, and 0.41 to 0.63, 0.73, and 0.62, respectively. The F1-score of ACGAN-SVM is often the highest value among all tested techniques. The improvement margin of ACGAN-SVM over the original version is often greater than that of ACGAN and TomekSMOTE. For example, the F1-scores of DT, RF, and SVM on UNSW-NB15 with ACGAN-SVM are increased to 0.67, 0.74, and 0.63 compared to those of the original dataset. On the CICIDS2017 dataset, only ACGAN-SVM achieves better performance than the original data. Additionally, on the RAWDATA, because the imbalance rate is relatively low, the classification accuracy of handling imbalance techniques is slightly improved compared with those of the original data.

The results in Table V give evidence for the benefit of using ACGAN and ACGAN-SVM to improve the machine learning performance in IDSs when the training datasets are highly imbalanced. In other words, using ACGAN and ACGAN-SVM to generate synthesized attacks enhances the quality of the IDS training datasets. This subsequently improves the effectiveness of the machine learning algorithms when they are trained on the augmented datasets of ACGAN and ACGAN-SVM. The reason for the better performance of ACGAN and ACGAN-SVM compared to others could be that the synthesized samples of the traditional sampling techniques may not fully follow the original data distribution and that this problem is mitigated in the generative models (see Subsection VI.3). Moreover, using SVM to remove unimportant samples which do not contribute much to the classification algorithms helps ACGAN-SVM to further improve the performance of classifiers over using ACGAN.

TABLE V
RESULT OF DT, RF, AND SVM ON MULTICLASS CLASSIFICATION FOR NSL-KDD, UNSW-NB15, CICIDS2017, AND RAWDATA DATASETS .

| **Algorithm** | **Augmented datasets** | **NSL-KDD** | | | **UNSW-NB15** | | | **CICIDS2017** | | | **RAWDATA** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** |
| **DT** | ORIGINAL | 0.87 | 0.83 | 0.82 | 0.51 | 0.50 | 0.42 | 0.97 | 0.97 | 0.97 | 0.97 | 0.95 | 0.95 |
| | SMOTE-SVM | 0.85 | 0.80 | 0.82 | 0.49 | 0.47 | 0.46 | 0.92 | 0.85 | 0.87 | **0.98** | 0.95 | 0.96 |
| | BalanceCascade | 0.80 | 0.81 | 0.79 | 0.63 | 0.61 | 0.58 | 0.95 | 0.94 | 0.94 | 0.97 | 0.95 | 0.96 |
| | TomekSMOTE | 0.86 | 0.82 | 0.82 | 0.65 | 0.62 | 0.60 | 0.94 | 0.93 | 0.93 | 0.95 | **0.97** | 0.96 |
| | WGAN | 0.83 | 0.80 | 0.78 | 0.64 | 0.60 | 0.61 | 0.96 | 0.95 | 0.94 | 0.96 | 0.96 | 0.96 |
| | ACGAN | 0.89 | **0.85** | 0.83 | 0.66 | 0.61 | 0.63 | 0.98 | 0.97 | 0.97 | 0.96 | 0.96 | 0.96 |
| | ACGAN-SVM | **0.90** | 0.84 | **0.84** | **0.67** | **0.63** | **0.67** | **0.98** | **0.98** | **0.98** | 0.97 | **0.97** | **0.97** |
| **RF** | ORIGINAL | 0.81 | 0.76 | 0.72 | 0.46 | 0.54 | 0.41 | **0.99** | 0.99 | 0.98 | 0.97 | 0.96 | 0.96 |
| | SMOTE-SVM | 0.82 | 0.78 | 0.74 | 0.74 | 0.70 | 0.71 | **0.99** | 0.95 | **0.99** | 0.98 | **0.98** | **0.98** |
| | BalanceCascade | 0.83 | 0.79 | 0.77 | 0.82 | 0.67 | 0.70 | 0.98 | 0.98 | 0.98 | 0.97 | **0.98** | 0.97 |
| | TomekSMOTE | 0.88 | 0.80 | 0.77 | **0.84** | 0.69 | 0.72 | **0.99** | **0.99** | **0.99** | 0.97 | **0.98** | 0.97 |
| | WGAN | 0.81 | 0.76 | 0.84 | 0.69 | 0.70 | 0.72 | 0.96 | 0.97 | 0.96 | 0.98 | **0.98** | **0.98** |
| | ACGAN | 0.84 | 0.79 | 0.78 | 0.69 | 0.72 | 0.73 | **0.99** | **0.99** | **0.99** | 0.98 | 0.97 | **0.98** |
| | ACGAN-SVM | **0.84** | **0.82** | **0.80** | 0.83 | **0.76** | **0.74** | **0.99** | **0.99** | **0.99** | **0.99** | **0.98** | **0.98** |
| **SVM** | ORIGINAL | 0.67 | 0.74 | 0.69 | 0.46 | 0.55 | 0.41 | 0.93 | 0.92 | 0.92 | 0.94 | 0.94 | 0.94 |
| | SMOTE-SVM | 0.81 | 0.77 | 0.75 | 0.65 | 0.52 | 0.54 | 0.92 | 0.84 | 0.86 | **0.96** | 0.94 | 0.95 |
| | BalanceCascade | 0.70 | 0.69 | 0.63 | 0.75 | 0.61 | 0.59 | 0.93 | 0.81 | 0.85 | 0.95 | 0.94 | 0.95 |
| | TomekSMOTE | 0.81 | 0.78 | 0.68 | 0.71 | 0.63 | 0.60 | **0.94** | 0.85 | 0.88 | 0.94 | 0.95 | 0.95 |
| | WGAN | **0.83** | **0.80** | **0.84** | 0.75 | 0.63 | 0.62 | **0.94** | 0.93 | 0.92 | **0.96** | 0.95 | **0.96** |
| | ACGAN | 0.81 | 0.74 | 0.76 | 0.76 | 0.65 | 0.62 | **0.94** | 0.94 | 0.93 | **0.96** | **0.96** | **0.96** |
| | ACGAN-SVM | **0.83** | 0.79 | 0.78 | **0.78** | **0.66** | **0.63** | **0.94** | **0.95** | **0.95** | **0.96** | **0.96** | **0.96** |

TABLE VI
COMPUTATIONAL TIME (IN SECONDS) TO SYNTHESIZE DATA OF SAMPLING TECHNIQUES.

| **Methods** | **NSL-KDD** | | **UNSW-NB15** | | **CICIDS2017** | | **RAWDATA** | |
|---|---|---|---|---|---|---|---|---|
| | b-classes | m-classes | b-classes | m-classes | b-classes | m-classes | b-classes | m-classes |
| SMOTE-SVM | 280 | 2266 | 645 | 4026 | 3310 | 3439 | 312 | 155 |
| BalanceCascade | 256 | 5 | 33 | 3 | 413 | 4 | 232 | 137 |
| TomekSMOTE | 386 | 1651 | 88 | 503 | 1251 | 5946 | 342 | 901 |
| WGAN | 5700 | 5287 | 6230 | 5893 | 5902 | 6101 | 6003 | 6101 |
| ACGAN | 7200 | 6884 | 7508 | 6280 | 6285 | 6348 | 5974 | 6048 |
| ACGAN-SVM | 7935 | 7725 | 8276 | 7198 | 7304 | 7122 | 7502 | 7107 |

## 2. Computational Time of Sampling Approaches

This subsection compares the computational time of the sampling methods. The computational time in seconds of five sampling techniques to generate the augmented datasets is showed in Table VI. It can be seen that the computational time of BalanceCascade and TomekSMOTE are relatively small while these values of others are often much higher. The running time of ACGAN-SVM is always greater than the time of all other approaches. The reason is that this method needs to train both the ACGAN model and the SVM model before it is used to create the synthesize data. However, the sampling phase can be considered as the pre-processing data step. When applying a machine learning model to a real world problem like IDS, the computational time for prediction is often the most desirable and the sampling step does not affect the prediction time of the classifiers.

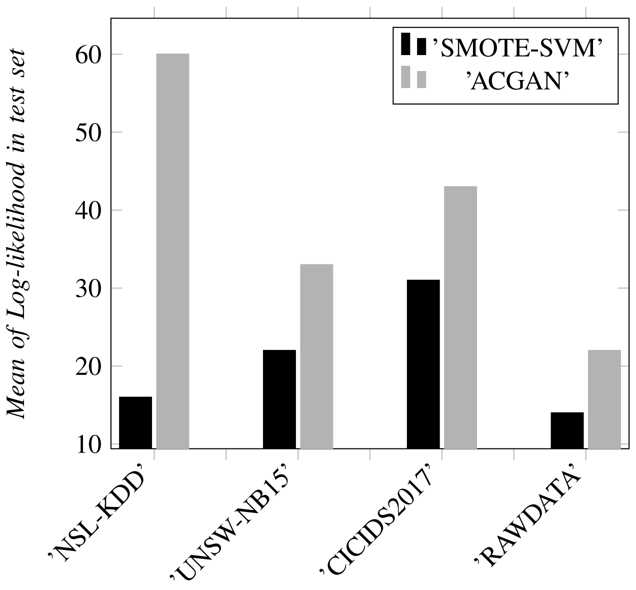

Figure 2. Parzen window log-likelihood of test set.

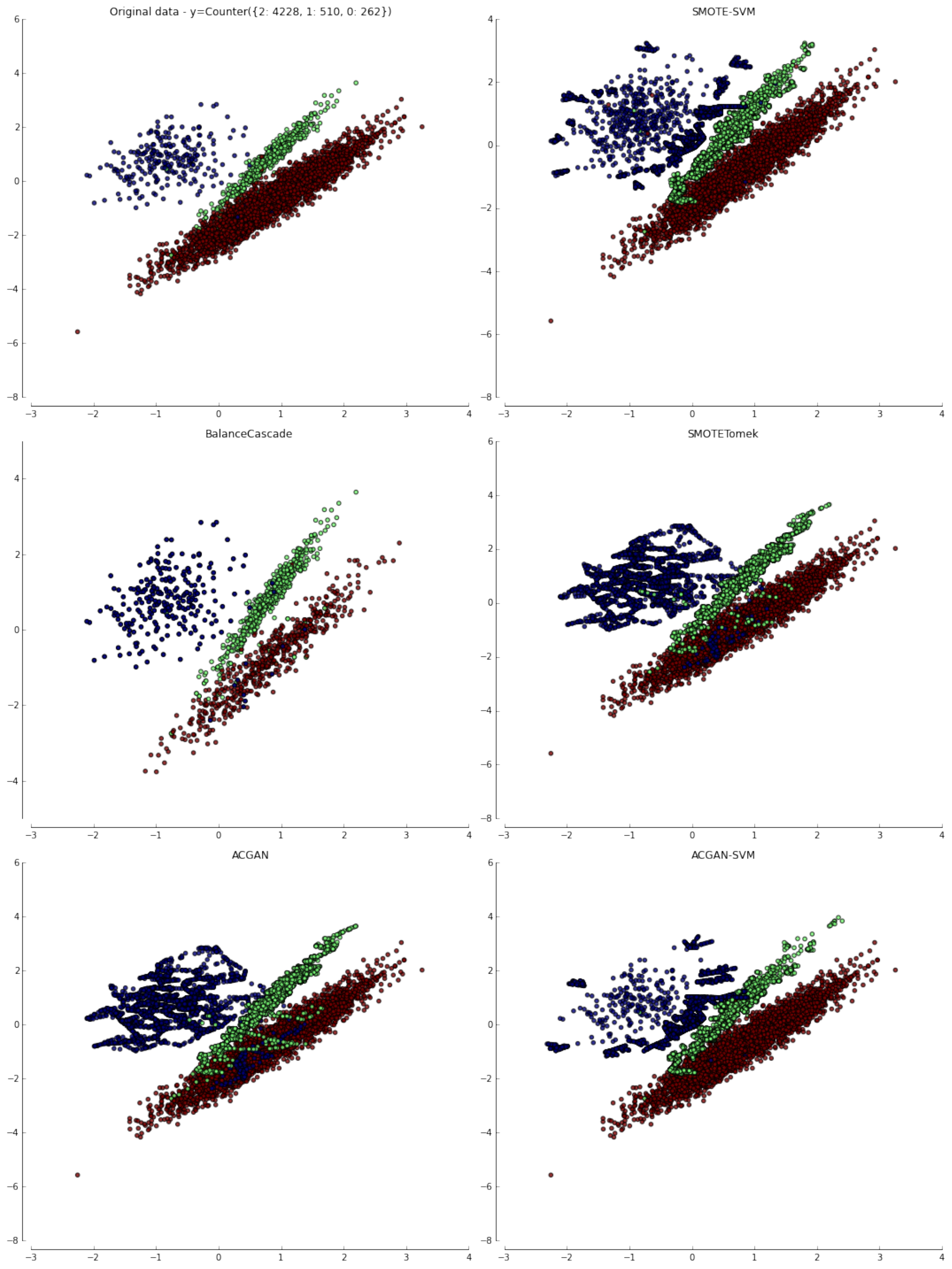

Figure 3. Illustration of oversampling techniques.

## 3. Analysis of Generated Data

This subsection qualitatively evaluates the generated data of SMOTE-SVM and ACGAN by measuring whether this data converges to the true data distribution. This experiment is applied to assess generated data techniques by oversampling techniques. Thus, we just assess two sampling techniques such as SMOTE-SVM and ACGAN. We applied the procedure of Goodfellow et al. [41] to assess the quality of generative models. For each dataset, we use a Gaussian Parzen window to fit datasets. Then, the log-likelihood of each generative model under the true distribution is reported. Experimental results are shown in Fig. 2 where a greater value presents a better model.

It can be seen that the log-likelihood of ACGAN is always higher than the value of SMOTE-SVM on three tested datasets. The reason is that ACGAN are trained to learn the true distribution of the original data while SMOTE-SVM does not do so. The figure evidences that the generated data of ACGAN correlates more strongly to the original data than the data of SMOTE-SVM. This result supports for the better performance of machine learning algorithms when they are trained on the augmented datasets of ACGAN compared to those trained on the augmented datasets of SMOTE-SVM.

### 4. Borderline Samples Visualization

This subsection visualizes the data synthesized by ACGAN, ACGAN-SVM, and other sampling techniques. We created a random dataset of 5000 samples with two features. The dataset includes three classes: two minor classes and one major class. The ratio of samples in each class is 0.05 : 0.10 : 0.85. This dataset is visualized on the top left of Fig. 3. In this figure, the red dots, blue dots, and green dots are the data samples of the major class, the first and the second minor classes, respectively. Using ACGAN, we generated 3966 samples for the first minor class and 3718 samples for the second minor class. Totally, 7684 samples of two minor classes were generated to form a balanced dataset. Similarly, SMOTE-SVM and ACGAN-SVM were used to generate 7684 samples for two minor classes. BalanceCascade reduced the number of samples of the major class to 524 and oversampled the most minor class to 524 to balance with the middle class. For TomekSMOTE, we oversampled two minor classes to have 4163 and 4158 samples, respectively. After that, 117 samples in the major class is reduced to form the augmented dataset. The generated datasets of all sampling techniques are presented in Fig. 3.

It can be seen from Fig. 3 that BalanceCascade is only the technique that reduces the number of data samples. All other techniques generate synthesized data from the original data. Among four oversampling techniques, we can see that both ACGAN and TomekSMOTE generate many samples that are in the middle area of the minor classes. Conversely, SMOTE-SVM and ACGAN-SVM only generate the samples that are near the borderline between classes. In the context of sampling techniques, the samples that are near the borderline between classes often contribute more significantly to the effectiveness of the classifiers than the samples located in the center area.

Moreover, Fig. 3 also shows that TomekSMOTE and ACGAN sometimes create the samples that are overlapped with the samples of other classes. Subsequently, these samples will be difficult for the classifiers to separate correctly. Conversely, the generated samples of SMOTE-SVM and ACGAN-SVM are not overlapped with other classes. This evidences that using SVM helps to remove the noisy samples generated by ACGAN and SMOTE. This provides partial explanation for the better performance of ACGAN-SVM compared to ACGAN and others. Moreover, the superior performance of ACGAN-SVM and ACGAN to other sampling techniques could be that the generated samples of ACGAN and ACGAN-SVM are better follow the original distribution than the traditional techniques as analyzed in Subsection VI.3.

Overall, the results in this section show that Generative Adversarial Networks can generate the meaningful samples for imbalanced IDS datasets. The classification algorithms that are trained on datasets augmented by ACGAN and particularly ACGAN-SVM are often better than those trained on the original dataset and the datasets obtained by using some popular sampling techniques.

## VII. Summary

In this paper, we proposed a novel approach based on generative adversarial networks for addressing imbalanced datasets in IDS. Specifically, we proposed two techniques based on ACGAN and ACGAN-SVM to generate samples for the attack classes in IDS. The augmented datasets of ACGAN and ACGAN-SVM are then used as the training dataset for three popular classification algorithms, SVM, DT, and RF. The experiments were conducted on three common IDS datasets: NSL-KDD, UNSW-NB15, and CICIDS2017 and one our own dataset, i.e., RAWDATA. The results show that the augmented datasets of ACGAN and ACGAN-SVM help machine learning to enhance accuracy on the imbalanced datasets although the training processes of ACGAN and ACGAN-SVM are often slower than the traditional sampling approaches. We analysed the quality of the generated data of ACGAN and SMOTE-SVM and visualized the borderline synthesized samples of five tested sampling techniques. The visualization partially explains the better performance of ACGAN and particularly ACGAN-SVM compared to others.

There are a number of research areas for future work that arise from this paper. First, we would like to examine the effectiveness of other deep learning generative models such as auto-encoder in generating the synthesized attacks for IDS. Second, the visualization technique has shed some light on the superior performance of ACGAN-SVM to other sampling techniques. However, this technique did not completely explain why ACGAN is also often better than SMOTE-SVM. We hypothesize that the generated data of ACGAN is better follow the original distribution than SMOTE-SVM. In the future, we will study the method to quantify the distribution of the synthesized samples of these sampling techniques [41]. Last but not least, we want to extend this approach to other problems in security and in other areas such as anomaly detection.

## Acknowledgement

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2019.05.

## References

[1] M. Albayati and B. Issac, "Analysis of intelligent classifiers and enhancing the detection accuracy for intrusion detection system," *Int. J. Comput. Intell. Syst.*, vol. 8, pp. 841–853, 2015.

[2] G. T. Nguyen, B. M. Nguyen, D. Tran, and L. Hluchý, "A heuristics approach to mine behavioural data logs in mobile malware detection system," *Data Knowl. Eng.*, vol. 115, pp. 129–151, 2018.

[3] X. Jing, Z. Yan, and W. Pedrycz, "Security data collection and data analytics in the internet: A survey," *IEEE Communications Surveys & Tutorials*, vol. Accepted Manuscript, 2018.

[4] J. Hussain, S. Lalmuanawma, and L. Chhakchhuak, "A two-stage hybrid classification technique for network intrusion detection system," *Int. J. Comput. Intell. Syst.*, vol. 9, pp. 863–875, 2016.

[5] D. A. Effendy, K. Kusrini, and S. Sudarmawan, "Classification of intrusion detection system (ids) based on computer network," *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, pp. 90–94, 2017.

[6] B. Xu, S. Chen, H. Zhang, and T. Wu, "Incremental k-nn svm method in intrusion detection," in *International Conference in Software Engineering and Service Science (ICSESS)*, 2017, pp. 712–717.

[7] A. Hadri, K. Chougdali, and R. Touahni, "Intrusion detection system using pca and fuzzy pca techniques," *2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, pp. 1–7, 2016.

[8] A. R. Syarif and W. Gata, "Intrusion detection system using hybrid binary pso and k-nearest neighborhood algorithm," in *IEEE conference in Information and Communication Technology and System (ICTS)*, 2017, pp. 181–186.

[9] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. C. Atkinson, and X. J. A. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," *CoRR*, vol. abs/1806.03517, 2018.

[10] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system," *Expert Syst. Appl.*, vol. 67, pp. 296–303, 2017.

[11] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Syst. Appl.*, vol. 42, pp. 2670–2679, 2015.

[12] B. W. Masduki, K. Ramli, F. A. Saputra, and D. Sugiarto, "Study on implementation of machine learning methods combination for improving attacks detection accuracy on intrusion detection system (ids)," *2015 International Conference on Quality in Research (QiR)*, pp. 56–64, 2015.

[13] R. K. Malaiya, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim, "An empirical evaluation of deep learning for network anomaly detection," in *2018 International Conference on Computing, Networking and Communications, ICNC*, 2018, pp. 893–898.

[14] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, pp. 1–13, 2017.

[15] S. Rodda, "Network intrusion detection systems using neural networks," in *Information Systems Design and Intelligent Applications*. Springer, 2018, pp. 903–908.

[16] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE communications magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[17] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54 024–54 033, 2019.

[18] S. Rodda and U. S. R. Erothi, "Class imbalance problem in the network intrusion detection systems," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016, pp. 2685–2688.

[19] M. Tavallaee, E. Bagheri, W. Lu, , and A. A. Ghorbani, "Nsl-kdd data set for network-based intrusion detection systems," http://nsl.cs.unb.ca/NSL-KDD/, 2009, accessed: 2018-04-10.

[20] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference, MilCIS 2015, Canberra, Australia, November 10-12, 2015*, 2015, pp. 1–6.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116.

[22] L. Vu, B. C. Thanh, and Q. U. Nguyen, "A deep learning based method for handling imbalanced problem in network traffic classification," in *Proceeding of Symposium on Information and Communication Technology*, 2017, pp. 333–339.

[23] A. A. Aburomman and M. B. I. Reaz, "A novel svm-knn-pso ensemble method for intrusion detection system," *Appl. Soft Comput.*, vol. 38, pp. 360–372, 2016.

[24] A. AminAburomman and M. B. IbneReaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers," *Computers & Security*, vol. 65, pp. 135–152, 2017.

[25] E. Hodo, X. J. A. Bellekens, A. Hamilton, C. Tachtatzis, and R. C. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," *CoRR*, vol. abs/1701.02145, 2017.

[26] Akashdeep, I. Manzoor, and N. Kumar, "A feature reduced intrusion detection system using ANN classifier," *Expert Syst. Appl.*, vol. 88, pp. 249–257, 2017.

[27] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *Sixth International Conference on Data Mining (ICDM'06)*, pp. 965–969, 2006.

[28] C. G. Cordero, E. Vasilomanolakis, A. Wainakh, M. Mühlhäuser, and S. Nadjm-Tehrani, "On generating network traffic datasets with synthetic attacks for intrusion detection," *arXiv preprint arXiv:1905.00304*, 2019.

[29] L. Arnaboldi and C. Morisset, "Generating synthetic data for real world detection of dos attacks in the iot," in *Federation of International Conferences on Software Technologies: Ap-*

*plications and Foundations*. Springer, 2018, pp. 130–145.

[30] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid intelligent intrusion detection scheme," *Soft Computer Industry Application*, pp. 193–303, 2011.

[31] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *International Conference on Platform Technology and Service (PlatCon)*, 2016, pp. 1–5.

[32] V. Belenko, V. Chernenko, M. Kalinin, and V. Krundyshev, "Evaluation of gan applicability for intrusion detection in self-organizing networks of cyber physical systems," in *2018 International Russian Automation Conference (RusAutoCon)*. IEEE, 2018, pp. 1–7.

[33] A. D. Pozzolo, O. Caelen, S. Waterschoot, and G. Bontempi, "Racing for unbalanced methods selection," in *Intelligent Data Engineering and Automated Learning - IDEAL 2013*, 2013, pp. 24–31.

[34] C. Drummond and R. Holte, "C4.5 class imbalance, and cost sensitivity: why under-sampling beats over-sampling," *Workshop on Learning from Imbalanced Data Sets II*, vol. 11, pp. 1–8, 2003.

[35] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[36] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *International Journal of Knowledge Engineering and Soft Data Paradigms (IJKESDP)*, vol. 3, no. 1, pp. 4–21, 2011.

[37] A. Namvar, M. Siami, F. Rabhi, and M. Naderpour, "Credit risk prediction in an imbalanced social lending environment," *Int. J. Comput. Intell. Syst.*, vol. 11, no. 1, pp. 925–935, 2018.

[38] Q. Wang, Z. Luo, J. Huang, Y. Feng, and Z. Liu, "A novel ensemble method for imbalanced data learning: Bagging of extrapolation-smote SVM," *Comp. Int. and Neurosc.*, vol. 2017, pp. 1 827 016:1–1 827 016:11, 2017.

[39] J. Charlier, A. Singh, G. Ormazabal, R. State, and H. Schulzrinne, "Syngan: Towards generating synthetic network attacks using gans," *arXiv preprint arXiv:1908.09899*, 2019.

[40] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[41] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," in *Conference on Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[42] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 2642–2651.

[43] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 2226–2234.

[44] Winpcap, "The industry standard windows packet library," https://www.winpcap.org/default.htm, online; accessed 20 December 2019.

[45] SQL map, "Automatic SQL injection and database takeover tool," http://sqlmap.org/, online; accessed 20 December 2019.

[46] Scapy library, "Packet crafting for Python2 and Python3," https://scapy.net/, online; accessed 20 December 2019.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[49] G. Research, "Tensorflow tutorial," https://www.tensorflow.org/, 2015, accessed: 2018-04-24.

[50] D. M. W. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine learning Technologies*, vol. 2, pp. 37–63, 2011.

**Ly Vu** received her B.Sc. and M.Sc. degrees in computer science from Le Quy Don Technical University (LQDTU), Vietnam and Inha University, Korea, respectively. She is currently pursuing the Ph.D. degree with LQDTU. She was a Lecturer with Le Quy Don Technical University. Her research interests include data mining, machine learning, deep learning, network security.

**Quang Uy Nguyen** received B.Sc. and M.Sc. degree in computer science from Le Quy Don Technical University (LQDTU)), Vietnam and the PhD degree at University College Dublin, Ireland. Currently, he is a senior lecturer at LQDTU and the director of Machine Learning and Applications research group at LQDTU. His research interest includes Machine Learning, Computer Vision, Information Security, Evolutionary Algorithms and Genetic Programming.