signals

MDPI

Article

A Comparative Analysis of the TDCGAN Model for Data Balancing and Intrusion Detection

Mohammad Jamoos [1,2], Antonio M. Mora [1], Mohammad AlKhanafseh [3] and Ola Surakhi [4,*]

[1] Department of Signal Theory, Telematics and Communications, University of Granada, 18012 Granada, Spain; jamoos@staff.alquds.edu (M.J.); amorag@ugr.es (A.M.M.)
[2] Department of Computer Science, School of Science and Technology, Al-Quds University, Jerusalem P.O. Box 51000, Palestine
[3] Department of Computer Science, Birzeit University, West Bank P.O. Box 14, Palestine; malkhanafseh@birzeit.edu
[4] Cybersecurity Department, American University of Madaba, Madaba 11821, Jordan
* Correspondence: o.surakhi@aum.edu.jo

Abstract: Due to the escalating network throughput and security risks, the exploration of intrusion detection systems (IDSs) has garnered significant attention within the computer science field. The majority of modern IDSs are constructed using deep learning techniques. Nevertheless, these IDSs still have shortcomings where most datasets used for IDS lies in their high imbalance, where the volume of samples representing normal traffic significantly outweighs those representing attack traffic. This imbalance issue restricts the performance of deep learning classifiers for minority classes, as it can bias the classifier in favor of the majority class. To address this challenge, many solutions are proposed in the literature. TDCGAN is an innovative Generative Adversarial Network (GAN) based on a model-driven approach used to address imbalanced data in the IDS dataset. This paper investigates the performance of TDCGAN by employing it to balance data across four benchmark IDS datasets which are CIC-IDS2017, CSE-CIC-IDS2018, KDD-cup 99, and BOT-IOT. Next, four machine learning methods are employed to classify the data, both on the imbalanced dataset and on the balanced dataset. A comparison is then conducted between the results obtained from each to identify the impact of having an imbalanced dataset on classification accuracy. The results demonstrated a notable enhancement in the classification accuracy for each classifier after the implementation of the TDCGAN model for data balancing.

Keywords: data balancing; deep learning; generative adversarial network; intrusion detection systems; security; TDCGAN

Citation: Jamoos, M.; Mora, A.M.; AlKhanafseh, M.; Surakhi, O. A Comparative Analysis of the TDCGAN Model for Data Balancing and Intrusion Detection. *Signals* **2024**, *5*, 580–596. https://doi.org/10.3390/signals5030032

Academic Editor: Jesús Brezmes

Received: 11 July 2024
Revised: 15 August 2024
Accepted: 4 September 2024
Published: 12 September 2024

1. Introduction

The rapid advancement of information technologies, such as smart devices, the Internet of Things (IoT), cloud computing, and big data, has led to an unprecedented surge in the number of devices connected to the Internet. Consequently, networks are expanding, becoming larger and more complex. However, this expansion also amplifies the risk of cyberattacks, as the sheer scale of networks makes them increasingly challenging to monitor effectively. Developing effective Network Intrusion Detection Systems (IDS) is essential for detecting cyberattacks. This requires the efficient and swift analysis of network traffic flow data, referred to as cybersecurity data [1].

The challenge with much of the network traffic flow data or cybersecurity data lies in its inherent imbalance. Usually, there is a substantial surplus of normal traffic data compared to instances of attacks. This remains consistent even with well-established benchmark datasets. Dealing with imbalanced data presents a significant hurdle for both machine learning and deep learning algorithms as many of these datasets require multi-class classification, which further complicates the task [2,3].

Machine learning methods rely on historical data for training, and they can be significantly impacted by imbalanced data proportions. In cases where training data are extremely imbalanced, with one class greatly outnumbering the others, the majority data (the class or classes with larger proportions) will exert a stronger influence on the machine learning model compared to minority data (the class or classes with lesser proportions). Consequently, the machine learning model tends to perform well in recognizing majority data, but may struggle in accurately identifying minority data under such circumstances [4–6].

To address the challenge of imbalanced datasets, oversampling techniques are commonly utilized. Conventional techniques utilize interpolation to generate samples near existing data points. Examples include the Synthetic Minority Oversampling Technique (SMOTE) and the Adaptive Synthetic Sampling Approach (ADASYN) [7]. However, a recent development known as the generative adversarial network (GAN) presents a novel approach to sample generation. GANs, short for Generative Adversarial Networks, are deep neural network models consisting of a generator and a discriminator. The generator is responsible for producing realistic data, while the discriminator's task is to distinguish between generated data and real ones. GAN allows the generator to efficiently learn data characteristics by participating in a competitive interaction with the discriminator, mimicking data distributions. GANs have made substantial advancements in producing diverse types of data, such as images, audio, and text [8–10]. Consequently, researchers across different fields are increasingly integrating this technique into their research efforts. As a result of the success and widespread adoption of adversarial networks, numerous modifications to the original model have been proposed in the data balancing domain.

The Triple Discriminator Conditional Generative Adversarial Network (TDCGAN) is an enhanced version of GANs, an unsupervised deep learning method. It consists of a single generator and three discriminators, with an additional layer incorporated at the end for decision making or selection [11]. In TDCGAN, the generator functions by using random noise from a latent space as input to create synthetic data that closely mimics real data, with the goal of fooling the discriminators. Each discriminator is a distinct deep neural network with its own architecture and parameter configurations. Their primary role is to extract features from the generator's output and classify the data with varying degrees of accuracy, which varies across the discriminators. In the TDCGAN architecture, a new component called the election layer is added at the end. This layer aggregates the outputs from the three discriminators and applies an election process to select the optimal result, choosing the one with the highest classification accuracy. This method is similar to an ensemble approach, where combining multiple inputs leads to a more accurate outcome.. The ensemble method in TDCGAN combines different neural network architectures and tuning strategies to achieve a superior result. Specifically, TDCGAN integrates multiple network architectures with distinct roles (e.g., generator and discriminator), each contributing to different aspects of the learning process. This integration is akin to an ensemble approach where multiple models are combined to improve overall performance.

TDCGAN tackles the challenge of high-class imbalance through an analysis of the UGR'16 dataset.

To assess the effectiveness of the proposed GAN model, TDCGAN, across various datasets, this paper introduces an evaluation framework. This framework involves applying the TDCGAN model to different Intrusion Detection System (IDS) datasets for comprehensive performance evaluation. These datasets are CIC-IDS2017, CSE-CIC-IDS2018, KDD-cup 99, and BOT-IOT datasets. Following that, four machine learning techniques were employed to conduct a comparative analysis of intrusion detection classification tasks on the four datasets. The methods are: Decision Tree, Random Forest, Multi-Layer Perceptron (MLP), and Naive Bayes. This analysis was performed both before and after applying data balancing techniques.

The results showed a substantial improvement in machine learning performance after data balancing, which highlights the importance of addressing class imbalance in machine

learning models, even when the initial performance appears satisfactory. By balancing the dataset, the machine learning model's ability to generalize and accurately classify instances from both classes improved significantly. This is crucial in scenarios where correctly identifying instances from the minority class is equally important as the majority class.

The key contributions of this paper are outlined as follows:

1. Utilize the TDCGAN model to address a data imbalance in four benchmark datasets for an intrusion detection system.
2. Employ different machine learning techniques to test intrusion detection classification across four benchmark datasets before and after data balancing.
3. Perform a comparative analysis of the models utilized.

The remainder of this paper is organized as follows. Section 2 gives an overview about IDS datasets, machine learning methods, and TDCGAN. Section 3 presents the experiment design and setup. The results are given in Section 4. Finally, Section 5 gives the conclusions.

2. Materials and Methods

2.1. Datasets

In this study, we utilize four distinct datasets to assess and compare the efficacy of the TDCGAN model against other chosen models in the context of data balancing and intrusion detection tasks. The datasets are CIC-IDS2017, CSE-CIC-IDS2018, KDD-cup 99, and BOT-IOT.

CIC-IDS2017 dataset

The CIC-IDS2017 dataset is a well-known resource in cybersecurity, commonly used for assessing the performance of intrusion detection systems. It contains network traffic data gathered from a variety of scenarios, including both normal traffic and several attack types, such as denial-of-service (DoS), distributed denial-of-service (DDoS), brute-force attacks, and botnet attacks. The dataset provides a realistic representation of network traffic in both benign and malicious environments, making it valuable for training and testing intrusion detection models. Researchers often use the CIC-IDS2017 dataset to benchmark the performance of their intrusion detection algorithms due to its diversity and comprehensiveness [12].

CSE-CIC-IDS2018 dataset

The CSE-CIC-IDS2018 dataset offers labeled network traffic flows, providing detailed information such as timestamps, source and destination IP addresses, ports, protocols, and attack types. This facilitates in-depth analysis and model training. The dataset includes seven distinct attack scenarios: Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and internal network infiltration [12].

KDD-cup 99 dataset

KDD-Cup 99 was developed for the KDD Cup 1999 competition, which focused on creating effective techniques for detecting unauthorized network access. The dataset features a substantial amount of network traffic data gathered from a simulated military network environment. It encompasses a range of network attacks, including DoS attacks, probing attacks, and user-to-root attacks, alongside normal traffic [13].

BOT-IOT dataset

The BOT-IoT dataset is a comprehensive dataset designed specifically for research in the domain of Internet of Things (IoT) security. It comprises network traffic data collected from a realistic IoT environment, including various types of IoT devices such as IP cameras, smart thermostats, and wearable fitness trackers. The dataset covers both benign traffic and malicious activities, including botnet attacks targeting IoT devices [14].

2.2. Machine Learning Methods

We opted for four methods in this study based on their demonstrated effectiveness, as highlighted by several researchers [15–18]. To assess their performance in data balancing and intrusion detection tasks, we compared them with our proposed model, TDCGAN, as introduced in [11]. Here is a brief overview of each model.

Decision Tree

Decision trees are a widely used supervised learning technique for both classification and regression tasks. They work by recursively dividing the feature space into distinct regions based on input feature values. Each node in the tree represents a decision criterion based on a particular feature, while the branches illustrate the potential outcomes of these decisions [19,20].

Random Forest

Random Forest is an ensemble learning technique that constructs multiple decision trees during training and merges their predictions using voting or averaging. Each tree is trained on a random subset of the training data and a random subset of features, which helps mitigate overfitting and enhances the model's generalization performance [21,22].

Multi-Layer Perception (MLP)

MLP is a type of feedforward artificial neural network that consists of multiple layers of nodes, including an input layer, one or more hidden layers, and an output layer. Each node in the network is connected to every node in the adjacent layers, and each connection has an associated weight. MLPs are trained using backpropagation and gradient descent to learn the optimal weights that minimize a specified loss function [23,24].

Naive Bayes

Naive Bayes is a probabilistic classifier that relies on Bayes' theorem and assumes that features are independent of each other, which is the "naive" part of the method. Despite this simplification, Naive Bayes often performs effectively, particularly in text classification tasks. It computes the probability of each class based on the input features and predicts the class with the highest probability [25,26].

TDCGAN

TDCGAN, i.e., triple discriminator conditional generative adversarial network, was initially introduced by [11], and subsequently applied for the data balancing and intrusion detection tasks. This model effectively balanced the UGR'16 dataset [27] and detected intrusions with an accuracy of 95%. The TDCGAN framework incorporates one generator and three discriminators, along with an additional selection layer at the end. The generator processes random noise from a latent space to create synthetic data that closely resembles real data, with the goal of evading detection by the discriminators. Each discriminator is a deep neural network with distinct architecture and parameter settings, tasked with extracting features from the generator's output and classifying the data with varying levels of accuracy. At the end of the TDCGAN architecture, a novel component known as the election layer is introduced. This layer aggregates the outputs from the three discriminators and uses an election process to determine the optimal result, selecting the outcome with the highest classification accuracy.

As proposed in [11], the generator utilizes points from the latent space to generate data that matches the distribution of real data in the dataset. This process involves employing fully connected layers comprising of an input layer, four hidden layers, and an output layer. The generator produces the data, $\hat{y}$, by applying the function $f(x, w)$ to the dataset features, where w represents the optimized weights of the neural network. The output z_j^L of the jth neuron in the Lth layer is calculated as:

$$z_j^L = \sigma\left(\sum_i w_{ji}^L x_i + b_j^L\right) \tag{1}$$

where the notation w_{ji}^L represents the weight of connection between the computing neuron and its ith input in the preceding layer and b_j^L is a bias parameter. The symbol $\sigma(.)$ is the activation function in the hidden layer.

On the other hand, the discriminators aim to classify the data into their respective classes, utilizing a fully connected MLP network. The generator model is trained to generate new data resembling the minor class in each dataset, while the discriminators work to differentiate between real data from the dataset and the synthesized data created

by the generator. The loss is subsequently calculated by comparing the real data with the generated data, and this loss is utilized exclusively to update the generator's weights, ensuring that only the generator's parameters are adjusted. The discrepancy between the real and generated data are measured using the cross-entropy loss function, as expressed in the following equation:

$$[LOSS]_C E = -1/N \sum_{n=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log[(1 - p(y_i)] \quad (2)$$

where y_i is the real data, $p(y_i)$ is the predicted data calculated by the sigmoid activation function, and N is the number of observations in the batch.

To ensure transparency and enable a fair comparison, this paper utilizes the same architecture that was proposed previously in [11]. The model is trained for 1000 epochs with a batch size of 128. The Adam optimizer is used, with a learning rate of 0.0001. The TDCGAN model allows the generator to train until it produces synthetic data samples that closely match the distribution of the original dataset.

These models were selected based on their established performance in similar tasks and are being compared with our proposed model, TDCGAN, which introduces a novel architecture specifically designed for data balancing and intrusion detection tasks.

3. Experiments Setup

A simplified workflow for the work is outlined in Figure 1. The flowchart of our study illustrates the iterative process applied to four different datasets to evaluate the impact of data augmentation using the TDCGAN model on classification performance.

The process is described below.

Data Iteration: The model undergoes four iterations, each time processing a different dataset. In each iteration, a sample from the dataset is selected and subjected to a series of preprocessing steps to ensure data consistency.

Data Preprocessing: This step includes a series of preprocessing operations such as:

Error Elimination and Gap Filling: The selected dataset undergoes preprocessing to eliminate errors, fill gaps, remove outliers, and discard irrelevant data types. Simple linear interpolation is employed to fill in any missing values.

Feature Encoding: To prepare the dataset for machine learning algorithms, categorical features are encoded using one-hot encoding.

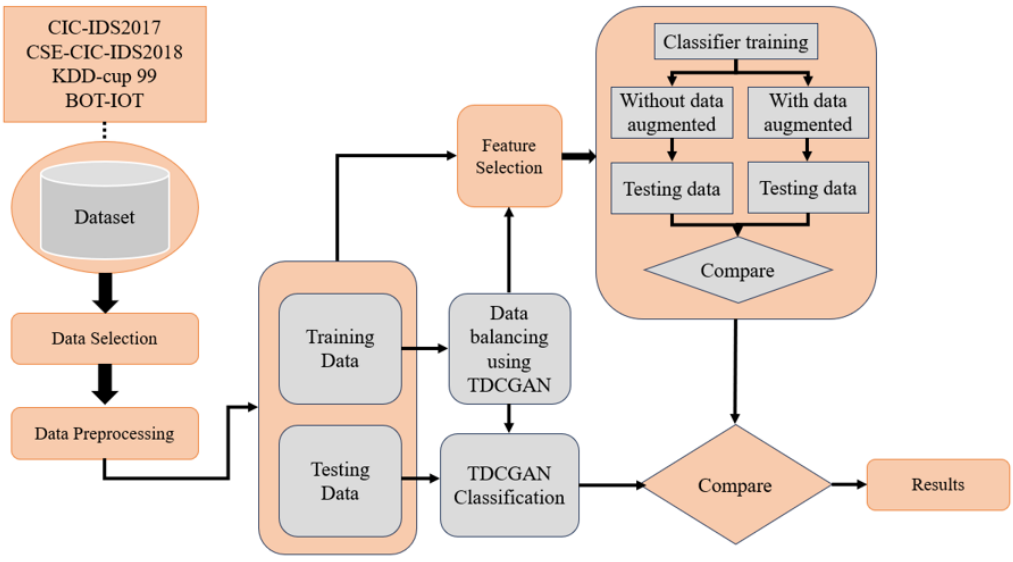


Figure 1. Flowchart of the study.

Scaling: The dataset is scaled using the MinMaxScaler from Scikit-learn, normalizing the values to the range [0,1].

Feature Selection (PCA): Principal Component Analysis (PCA) is applied to the dataset. PCA reduces the dimensionality of the data while retaining the most significant features, thereby enhancing computational efficiency and potentially improving model performance.

Data Splitting: The dataset is split into two parts: 70% of the data are allocated to the training set, while the remaining 30% is used as the testing set. We divided the dataset into 70% for training and 30% for testing to ensure the reliability of our model's evaluation. This division allows us to train the model on a sufficient portion of the data while reserving a separate, unseen portion for testing. This approach helps prevent overfitting and ensures that the model's performance is assessed on data it has not encountered before, providing a more accurate measure of its generalization ability. The 70 and 30 split is a standard practice in machine learning, ensuring a robust and unbiased comparison of different models or techniques.

TDCGAN Model Application: The TDCGAN model is trained using the training data of each dataset to generate synthetic samples. These samples are used to improve data balance within the training set.

Classifier Training and Evaluation: First, the classifier is trained using only the original training data, without any TDCGAN-generated samples, and its performance is evaluated on the testing data. Then, the classifier is then trained using the original training data augmented with TDCGAN-generated samples, and its performance is again evaluated on the same testing dataset.

Performance Comparison: The performance metrics (accuracy, precision, recall, and F1-score) from both scenarios—training without and with TDCGAN-generated samples—are compared. This comparison assesses the impact of TDCGAN-generated data augmentation on the classification performance for each dataset.

Final Evaluation: The final step involves comparing the evaluation metrics across all classifiers and datasets. This comparison allows us to identify the best-performing classifier and evaluate the overall effectiveness of the TDCGAN model in enhancing classification performance.

The pseudo code for the overall proposed approach is offered in Figure 2.

```
1. datasets = [dataset1, dataset2, dataset3, dataset4]
2. for dataset in datasets:
    a. dataset = eliminate_errors_and_fill_gaps(dataset)
    b. dataset = one_hot_encode(dataset)
    c. dataset = min_max_scale(dataset)
    d. dataset = apply_PCA (dataset)
    e. training_set, testing_set = split_data (dataset, train_size=0.7)
    f. tdgan_model = train_TDCGAN (training_set)
    g. synthetic_samples = tdgan_model. generate_samples ()
    h. augmented_training_set = training_set + synthetic_samples
    i. classifier = train_classifier(training_set)
    j. performance_without_TDCGAN = evaluate_classifier (classifier, testing_set)
    k. classifier = train_classifier(augmented_training_set)
    l. performance_with_TDCGAN = evaluate_classifier (classifier, testing_set)
    m. compare performance (performance_without_TDCGAN,
       performance_with_TDCGAN)
    n. store results (performance_without_TDCGAN, performance_with_TDCGAN)
3. end for
4. final_comparison = compare_across_datasets ()
5. identify_best_classifier(final_comparison)
6. evaluate_TDCGAN_effectiveness(final_comparison)
```

Figure 2. The pseudo code of the proposed appraoch.

The selection of the data sample and the data preparation process are described in Section 3.1. The feature selection for each dataset and the details about the experiment's setup are elaborated in the following sections, respectively.

3.1. Data Selection and Preparation

In this paper, the authors conducted various experiments using four distinct datasets. Prior to initiating each experiment, the initial step involves selecting a sample from each dataset. As deep learning algorithms demand significant hardware resources, such as CPU, memory, and GPU, for data processing and training. To handle this, a subset of data points encompassing all categories of normal and anomalous traffic from each dataset was meticulously selected.

To maintain uniform distribution of instances (both attacks and non-attacks) within the dataset samples, the authors repeat this selection process ten times for each dataset. The subset selection process, inclusive of all attack types, was executed with specific measures to counter imbalanced distributions and biases. The following measures were implemented:

- Stratified sampling: The subset selection utilized stratified sampling techniques to preserve a proportional representation of each attack type, ensuring a balanced distribution of attacks within the subset.
- Class balancing: Additional measures were implemented to balance the representation of different attack types in the subset. These actions might include oversampling the minority classes or undersampling the majority classes to address issues related to imbalanced distributions.
- Randomization: Randomization techniques were used during the subset selection process to reduce potential biases. This approach ensured that the selection was not influenced by any specific order or predetermined biases.

After each selection iteration, the authors plot the data distributions for each sample and compare them to ensure equality. This step guarantees that the random selection encompasses all potential observations present in the dataset.

Following the data selection, each sample undergoes a series of preprocessing steps to ensure consistency, eliminating errors, gap filling, outliers' removal, and irrelevant data types. We utilized simple linear interpolation to fill in any gaps present in the dataset.

To facilitate processing by machine learning algorithms, certain features in the dataset require encoding. One-hot encoding is employed for this purpose. Subsequently, the dataset undergoes scaling using the MinMaxScaler from the Scikit-learn library, which scales the values to the interval [0,1].

3.2. Feature Selection

In our feature selection process, Principal Component Analysis (PCA) plays a crucial role. PCA helps identify and retain the most important features by transforming the original variables into a new set of uncorrelated variables, called principal components. By reducing the dimensionality of the data while preserving as much variance as possible, PCA enables us to streamline the computational load and enhance model performance [28]. Through PCA, we extract the key patterns and structures inherent in the data, empowering our subsequent analysis and modeling efforts with a focused and optimized feature set. Table 1 illustrates the top numerical features of each dataset based on their PCA values.

Table 1. Selected features for each dataset based on the PCA method.

Dataset	Original Features		Selected Features	
CIC-IDS2017	Benign	2,273,097	Benign	2,273,097
	DOS Hulk	231,073	DOS Hulk	231,073
	PortScan	158,930	PortScan	158,930
	DDoS	128,027	DDoS	128,027
	DOS GoldenEye	10,293	DOS GoldenEye	10,293
	FTP-Patator	7938	FTP-Patator	7938
	SSH-Patator	5879	SSH-Patator	5879
	DoS- slowloris	5796	DoS- slowloris	5796
	DoS slowhttptest	5499	DoS slowhttptest	5499
	Bot	1966	Bot	1966
	Web Attack Brute Force	1507	Other Attack	2227
	Web Attack XSS	652		
	Infiltration	367		
	Web Attack Sql Injection	21		
	Heartbleed	11		
CSE-CIC-IDS2018	Benign	1,222,612	Benign	1,222,612
	DDOS attack-HOIC	137,157	DDOS attack-HOIC	137,157
	DOS attack-Hulk	92,325	DOS attack-Hulk	92,325
	Bot	57,067	Bot	57,067
	FTP-Burteforce	38,881	FTP-Burteforce	38,881
	SSH-Burteforce	37,403	SSH-Burteforce	37,403
	Infilteration	32,470	Infilteration	32,470
	DOS attacks-SlowHTTPTest	27,902	DOS attacks-SlowHTTPTest	27,902
	DOS attacks-GoldenEye	8284	DOS attacks-GoldenEye	8284
	DDOS attack-LOIC-UDP	343	DOS attacks-Slowloris	2216
	Burte Force-Web	113	Other Attacks	533
	Burte Force-XSS	49		
	SQL Injection	16		
	Label	12		
KDD-cup 99	Smrf	2,807,886	Smrf	2,807,886
	Neptune	1,072,017	Neptune	1,072,017
	Normal	972,780	Normal	972,780
	Satan	15,892	Satan	15,892
	Ipsweep	12,481	Ipsweep	12,481
	Portsweep	10,413	Portsweep	10,413
	Nmap	2316	Nmap	2316
	back	2203	back	2203
	Warezclient	1020	Other attack	1422
	Teardrop	979	warezclient	1020
	Pod	264		
	Guess_passwd	53		
	Buffer_overflow	30		
	Land	21		
	Warezmaster	20		
	imap	12		
	Rootkit	10		
	Loadmodule	9		
	ftp_write	8		
	Multihop	7		
	Phf	4		
	Perl	3		
	spy	2		
BOT-IOT	UDP	3,170,060	UPD	3,170,060
	TCP	2,549,206	TCP	2,549,206
	Service_Scan	117,170	Service_Scan	117,170
	OS_Fingerprint	28,479	OS_Fingerprint	28,479
	HTTP	3902	HTTP	3902
	Normal	707	Normal	707
	Keylogging	100	Other Attack	111
	Data_Exfiltration	11		

To enhance the reliability of our comparison, we divided each sample of dataset into two parts: the first 70% served as the training set, while the remaining 30% constituted the testing set.

3.3. The Experimental Setup

The class distribution for each dataset is described in Table 1. In the experiment, initially, the TDCGAN model is applied on the training data for each dataset to generate samples and improve data balancing. The output of this training is then preserved for the data balancing evaluation issue. With visual Inspection, we examine the generated samples to ensure they resemble real data and capture important features. To assess the classification performance of the TDCGAN model, we employed it on the testing data for each dataset and conducted evaluation metrics.

After that, for the classifier evaluation and comparison, each machine learning method will be applied twice for each dataset: (1) Training without TDCGAN-generated samples: Train the classifier using only the original training data, without adding any TDCGAN-generated samples, then evaluate its performance on the testing data. (2) Training with TDCGAN-generated samples: Train the classifier using the original training data augmented with TDCGAN-generated samples, then evaluate its performance on the same testing dataset. By comparing the performance metrics (accuracy, precision, recall, and F1-score) for each classifier, we assess the impact of data augmentation with TDCGAN-generated samples on the classification performance.

Suppose the result of classifier training with TDCGAN-generated samples outperforms the results trained without TDCGAN. In that case, it suggests that the TDCGAN-generated samples have effectively improved the classifier's ability to generalize to unseen data or to capture the underlying data distribution better. Conversely, if there is no significant improvement or if performance deteriorates, it may indicate that the TDCGAN-generated samples are not beneficial for the task or that they introduce noise or bias.

The final step involves comparing the evaluation metrics of each classifier with the TDCGAN model for the classification task applied to the testing data of each dataset. This comparison allows us to evaluate the classification performance of each model and identify the best-performing one.

3.4. Evaluation Metrics

To assess the effectiveness of each model, we use performance metrics such as classification accuracy, precision, recall, and the F1 score.

Accuracy (Acc) serves as our primary metric for evaluating the correct classification of data samples, taking into account all predictions made by the model, as described by the following equation:

$$Acc = (TP + TN)/(TP + TN + FP + FN), \tag{3}$$

where TP represents true positives, indicating the number of correctly predicted anomalies, TN represents true negatives, indicating the number of correctly predicted normal instances, FP represents false positives, indicating the number of normal instances incorrectly classified as anomalies, and FN represents false negatives, indicating the number of anomalies misclassified as normal.

Precision is used to measure the accuracy of correct predictions, calculated as the ratio of correctly predicted samples to the total number of predicted samples for a particular class, as illustrated by the following equation:

$$Precision = TP/(TP + FP) \tag{4}$$

Recall, also known as the true positive rate (TPR), is used to determine the ratio of correctly predicted samples for a specific class to the total number of instances of that class, as shown by the following equation:

$$TPR(Recall) = TP/(TP + FN) \tag{5}$$

Finally, the F1 score calculates the balance between precision and recall, evaluating the trade-off between these two metrics, as given by the following equation:

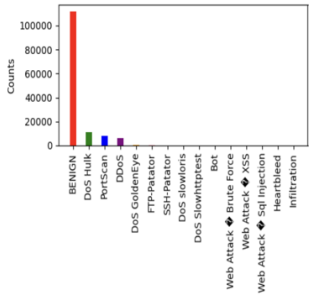
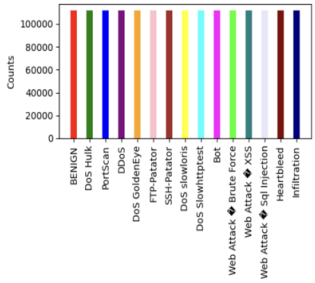
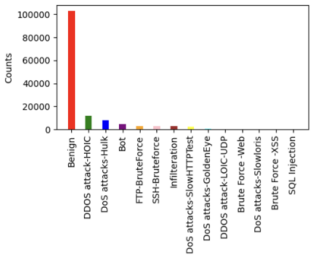
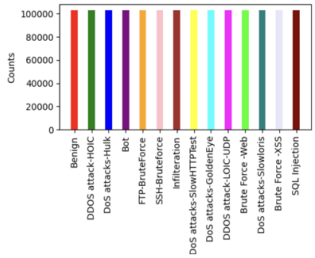
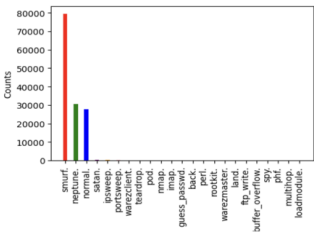
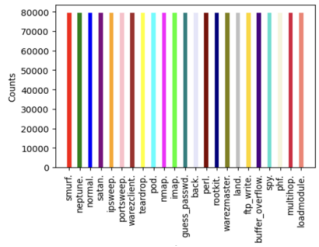
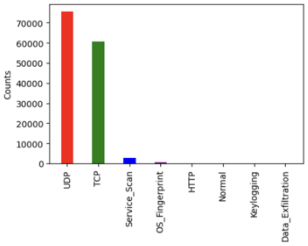
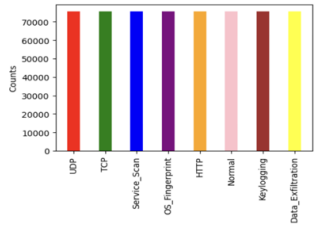
$$F1 = 2 * ((Precision * Recall)/(Precision + Recall)) \quad (6)$$

4. Results and Discussion

4.1. Experimental Results

To asses data balancing for each dataset after using TDCGAN model, we compared the class distribution before and after data balancing; the results are shown in Table 2.

Table 2. The class distribution before and after data balancing using the TDCGAN model.

Dataset	Class Distribution before Data Balancing	Class Distribution after Data Balancing
CIC-IDS2017		
CSE-CIC-IDS2018		
KDD-cup 99		
BOT-IOT		

After that, TDCGAN is utilized on the testing data for each dataset to evaluate its performance in classification tasks. This crucial step allows us to assess how effectively the model can categorize data instances following the balancing process. After applying the TDCGAN model, we perform thorough evaluations to assess its accuracy, precision, recall, and F1-score metrics. These results provide valuable insights into the effectiveness of the model in handling the testing data, offering a clear understanding of its classification capabilities in real-world scenarios. The results are presented in Table 3.

Table 3. The performance metrics for the TDCGAN model on testing data after data balancing.

Dataset	Accuracy	Precision	Recall	F1-Score
CIC-IDS2017	0.96	0.97	0.99	0.98
CSE-CIC-IDS2018	0.95	0.95	0.97	0.98
KDD-cup 99	0.95	0.96	0.95	0.96
BOT-IOT	0.96	0.95	0.95	0.96

To scrutinize the efficacy of data balancing techniques applied to TDCGAN across four distinct datasets, our experimental methodology involves conducting thorough assessments on both the training data prior to and following the data balancing process for each machine learning classifier. By doing so, we can elucidate the impact of data balancing on the classifiers' performance. Additionally, we employ the testing data to directly compare the performance of each classifier before and after the data balancing procedure. This comprehensive approach enables us to gauge the effectiveness of TDCGAN in improving classification accuracy and the robustness of various classifiers across diverse datasets. The results are presented in the Tables 4–11 below.

Table 4. The performance metrics for decision tree before data balancing.

Dataset	Accuracy	Precision	Recall	F1-Score
CIC-IDS2017	0.70	0.92	0.86	0.89
CSE-CIC-IDS2018	0.69	0.89	0.82	0.86
KDD-cup 99	0.75	0.85	0.63	0.73
BOT-IOT	0.72	0.83	0.79	0.80

Table 5. The performance metrics for decision tree after data balancing.

Dataset	Accuracy	Precision	Recall	F1-Score
CIC-IDS2017	0.85	0.92	0.90	0.90
CSE-CIC-IDS2018	0.77	0.91	0.89	0.86
KDD-cup 99	0.88	0.89	0.78	0.82
BOT-IOT	0.88	0.92	0.91	0.90

Table 6. The performance metrics for random forest before data balancing.

Dataset	Accuracy	Precision	Recall	F1-Score
CIC-IDS2017	0.68	0.83	0.77	0.72
CSE-CIC-IDS2018	0.61	0.79	0.78	0.80
KDD-cup 99	0.70	0.82	0.75	0.82
BOT-IOT	0.81	0.87	0.88	0.90

Table 7. The performance metrics for random forest after data balancing.

Dataset	Accuracy	Precision	Recall	F1-Score
CIC-IDS2017	0.78	0.80	0.83	0.79
CSE-CIC-IDS2018	0.82	0.84	0.86	0.87
KDD-cup 99	0.88	0.90	0.89	0.89
BOT-IOT	0.90	0.89	0.91	0.91

Table 8. The performance metrics for MLP before data balancing.

Dataset	Accuracy	Precision	Recall	F1-Score
CIC-IDS2017	0.82	0.84	0.85	0.87
CSE-CIC-IDS2018	0.84	0.88	0.82	0.84
KDD-cup 99	0.83	0.82	0.84	0.82
BOT-IOT	0.85	0.83	0.81	0.82

Table 9. The performance metrics for MLP after data balancing.

Dataset	Accuracy	Precision	Recall	F1-Score
CIC-IDS2017	0.92	0.90	0.93	0.91
CSE-CIC-IDS2018	0.93	0.92	0.94	0.91
KDD-cup 99	0.92	0.91	0.92	0.92
BOT-IOT	0.92	0.93	0.94	0.94

Table 10. The performance metrics for Naive Bayes before data balancing.

Dataset	Accuracy	Precision	Recall	F1-Score
CIC-IDS2017	0.60	0.71	0.72	0.69
CSE-CIC-IDS2018	0.65	0.67	0.70	0.68
KDD-cup 99	0.70	0.75	0.80	0.82
BOT-IOT	0.72	0.74	0.78	0.77

Table 11. The performance metrics for Naive Bayes after data balancing.

Dataset	Accuracy	Precision	Recall	F1-Score
CIC-IDS2017	0.80	0.85	0.82	0.81
CSE-CIC-IDS2018	0.73	0.81	0.73	0.76
KDD-cup 99	0.85	0.87	0.85	0.87
BOT-IOT	0.80	0.82	0.81	0.83

To assess the performance of the TDCGAN model relative to other machine learning models in terms of classification accuracy across four datasets, a comparative analysis of these models is conducted. The results are shown in the Figures 3–6.

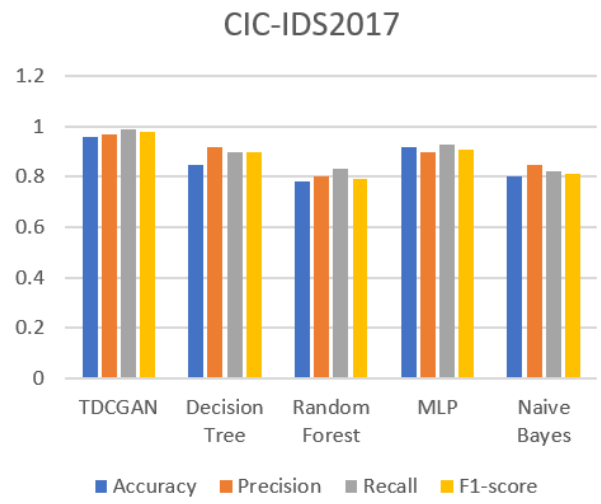


Figure 3. CIC-IDS2017.

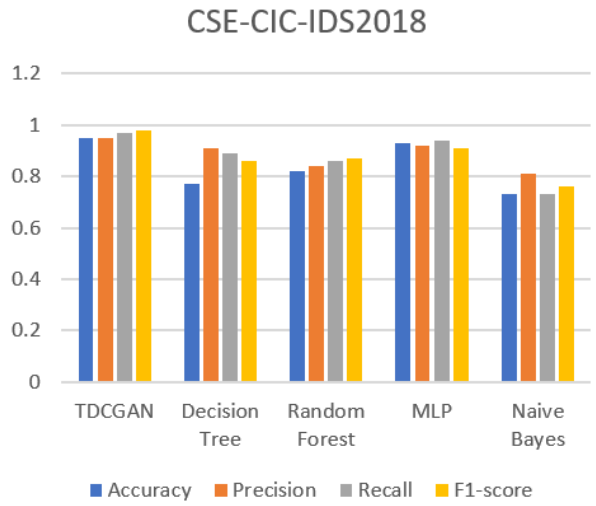


Figure 4. CSE-CIC-IDS2018.

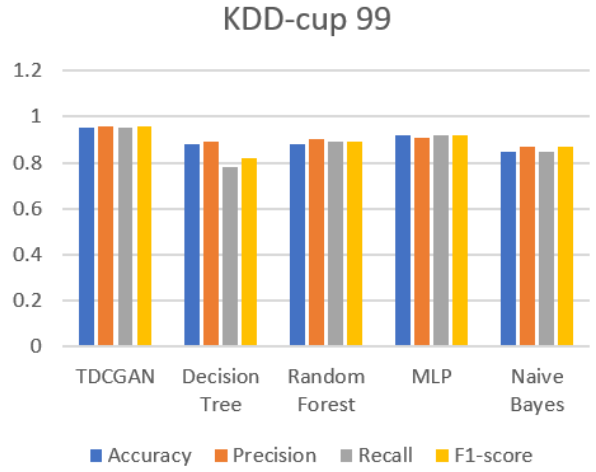


Figure 5. KDD-cup 99.

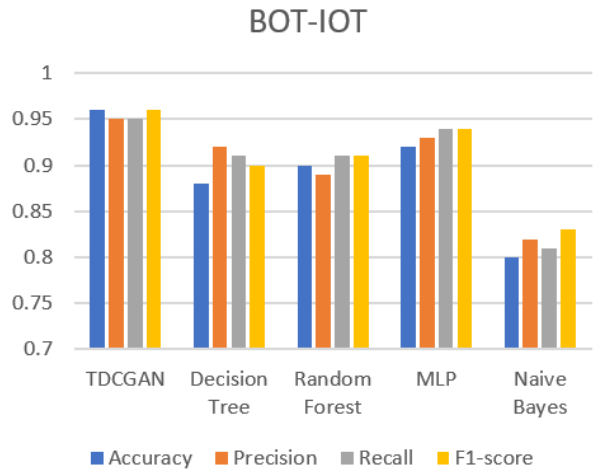


Figure 6. BOT-IOT.

4.2. Discussion

Before data balancing, the decision tree model yielded respectable performance across the datasets, with accuracies ranging from 0.69 to 0.75. However, there were noticeable disparities in performance metrics such as recall and F1-score, particularly for the KDD-cup 99 dataset, where the recall was relatively low at 0.63, indicating that the model struggled to correctly identify instances of the positive class.

After data balancing, there is a clear improvement in the performance of the decision tree model across all datasets. The accuracy has generally increased, with significant enhancements in metrics like precision, recall, and F1-score. For instance, in the CIC-IDS2017 dataset, the recall increased from 0.86 to 0.90, indicating that the model became more adept at correctly identifying instances of the positive class. Similar improvements are observed in other datasets as well.

The random forest model exhibited varying degrees of performance across the datasets. While it achieved relatively high accuracies, there were notable differences in precision, recall, and F1-score among the datasets. For instance, in the CSE-CIC-IDS2018 dataset, the precision was 0.79, indicating that the model correctly identified only 79% of the positive class instances, which might not be desirable depending on the application.

After data balancing, there is a substantial improvement in the performance of the random forest model across all datasets. The accuracy has notably increased, and there are significant enhancements in precision, recall, and F1-score as well. For example, in the CIC-IDS2017 dataset, the recall increased from 0.78 to 0.83, indicating a significant improvement in correctly identifying instances of the positive class.

The multilayer perceptron (MLP) model demonstrated strong performance across all datasets, with accuracies ranging from 0.82 to 0.85. Additionally, the model exhibited a good precision, recall, and F1-score values, indicating its ability to accurately classify instances from both classes. For instance, in the CIC-IDS2017 dataset, the model achieved a precision of 0.84, suggesting that 84% of the instances classified as positive were indeed true positives. Likewise, the recall and F1-score values were also high, demonstrating a strong balance between accurately identifying positive instances and reducing false negatives.

Even with its already strong performance, data balancing further improved the MLP model's effectiveness. Following data balancing, there was a noticeable enhancement in accuracy, precision, recall, and F1-score across all datasets. For instance, in the CIC-IDS2017 dataset, accuracy rose from 0.82 to 0.92, reflecting a significant boost in the model's capability to correctly classify instances from both classes. Likewise, precision, recall, and F1-score also saw improvements, indicating a more balanced and accurate classification, particularly for the minority class.

Lastly, before data balancing, the Naive Bayes model exhibited moderate performance across all datasets, with accuracies ranging from 0.60 to 0.72. While the model achieved

relatively high recall values, indicating its ability to correctly identify instances of the positive class, precision and F1-score values were comparatively lower, suggesting a higher rate of false positives and a suboptimal balance between precision and recall.

After data balancing, there is a noticeable improvement in the performance of the Naive Bayes model across all datasets. The accuracy has generally increased, and there are significant enhancements in precision, recall, and F1-score as well. For example, in the CIC-IDS2017 dataset, the accuracy increased from 0.60 to 0.80, indicating a substantial improvement in the model's overall classification performance. Moreover, precision, recall, and F1-score also improved, suggesting a more balanced and accurate classification across both classes.

The significant improvements in the performance of all machine learning models—decision tree, random forest, multilayer perceptron (MLP), and Naive Bayes—after data balancing strongly indicate that the use of the TDCGAN model for data balancing has substantially contributed to enhancing classification accuracy, as well as metrics like precision, recall, and F1-score. The TDCGAN model effectively tackles class imbalance by generating synthetic instances of the minority class while maintaining the underlying data distribution. This augmentation of the dataset ensures a more balanced representation of both classes, thereby enabling the machine learning models to learn more effectively from the data and make fairer and more accurate predictions. The consistent improvements across all models and datasets further highlight the efficacy of the TDCGAN approach in mitigating class imbalance issues and enhancing the overall performance of the machine learning classifiers.

When comparing the classification performance of each model across the four datasets, the results indicate overall improvement in performance for all models, with slightly superior results observed for the TDCGAN model. Conversely, the Naive Bayes model exhibited comparatively lower performance, particularly evident in the BOT-IOT dataset. The Naive Bayes model assumes conditional independence among features. However, the BOT-IOT dataset likely contains complex correlations between features that violate this assumption, leading to suboptimal performance of the Naive Bayes model. Specifically, in network intrusion detection tasks, certain features such as packet size, protocol type, and connection duration are often interdependent, and these correlations are critical for accurately distinguishing between normal and malicious traffic. The limitations of the Naive Bayes model in capturing these correlations may explain its lower performance on the BOT-IOT dataset. Furthermore, while TDCGAN is designed to handle high-class imbalance and generate synthetic samples, its effectiveness in preserving complex feature correlations during generation could also influence the overall performance when used in conjunction with models like Naive Bayes.

5. Conclusions

To demonstrate the impact of data balancing on enhancing classification accuracy and improving machine learning performance, this research conducted a comparative analysis of four machine learning models: Decision Tree, Random Forest, Multi-Layer Perceptron (MLP), and Naive Bayes, across various datasets. The primary objective of the study was to assess the efficacy of integrating the TDCGAN model for data balancing task. Utilizing diverse benchmark IDS datasets, the study implemented the TDCGAN model to address class imbalance. Subsequently, the machine learning models were utilized to classify the data both before and after applying data balancing techniques, aiming to assess the resulting effects on classification performance.

Before data balancing, the performance of various machine learning models, including decision tree, random forest, multilayer perceptron (MLP), and Naive Bayes, exhibited inconsistencies across datasets. Notably, metrics such as recall and F1-score displayed discrepancies, indicating challenges in accurately identifying positive class instances. However, after employing the TDCGAN model for data balancing, substantial improvements were observed across all models and datasets. The augmentation of the dataset with

synthetic instances generated by TDCGAN led to notable enhancements in accuracy, precision, recall, and F1-score. For instance, in the CIC-IDS2017 dataset, recall surged from 0.86 to 0.90 for the decision tree model, showcasing a significant improvement in positive class identification. Similar enhancements were observed across other models, highlighting the effectiveness of TDCGAN in mitigating class imbalance and enhancing overall classification performance.

The consistent improvements seen in the performance of machine learning models post-data balancing strongly indicate the efficacy of the TDCGAN approach. By generating synthetic instances of the minority class while preserving the underlying data distribution, TDCGAN effectively addressed class imbalance issues present in the datasets. This facilitated a more balanced representation of both classes, enabling the models to learn more effectively and make fairer and more accurate predictions. The substantial enhancements across metrics such as precision, recall, and F1-score underscore the significant role played by TDCGAN in refining the classification capabilities of the machine learning models. Thus, the study highlights the importance of employing advanced data balancing techniques like TDCGAN to improve the performance of machine learning models in scenarios with imbalanced datasets.

In summary, while the integration of TDCGAN has demonstrated significant improvements in addressing class imbalance and enhancing classification performance, it also introduces certain challenges. The addition of TDCGAN increases the complexity of the model development process, requiring careful tuning of additional hyperparameters, which demands extensive experimentation and expertise. Furthermore, the implementation of TDCGAN adds computational overhead, increasing the time and resources needed for generating synthetic instances and integrating them into the training data. These factors highlight the need for a balanced approach, where the benefits of using TDCGAN are carefully weighed against the added complexity and resource requirements

Author Contributions: Conceptualization, O.S., M.A. and M.J.; methodology, O.S., M.A. and M.J.; software, M.J.; validation, O.S., M.A., M.J. and A.M.M.; formal analysis, O.S., M.A., M.J. and A.M.M.; investigation, O.S.; resources, M.J.; data curation, M.J.; writing—original draft preparation, O.S. and M.A.; writing—review and editing, O.S. and M.A.; visualization, O.S., M.A. and M.J.; supervision, A.M.M.; project administration, O.S., M.A. and M.J.; funding acquisition, M.J. and A.M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research at AlQuds University.

Data Availability Statement: Data are contained within the article.

Acknowledgments: This research was funded by the Deanship of Scientific Research at AlQuds University. In addition, the work has been partially funded by the Spanish Ministry of Science, Innovation and Universities MICIU/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR, under projects TED2021-131699B-I00 and TED2021-129938B-I00. It has also been funded by projects PID2020-113462RB-I00, PID2020-115570GB-C22, and PID2023-147409NB-C21 of the Spanish Ministry of Economy and Competitiveness; as well as project C-ING-179-UGR23 financed by the "Consejería de Universidades, Investigación e Innovación" (Andalusian Government, FEDER Program 2021–2027).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep learning approach for intelligent intrusion detection system. *IEEE Access* **2019**, *7*, 41525–41550. [CrossRef]
2. Thabtah, F.; Hammoud, S.; Kamalov, F.; Gonsalves, A. Data imbalance in classification: Experimental evaluation. *Inf. Sci.* **2020**, *513*, 429–441. [CrossRef]
3. Liu, Y.; Li, X.; Chen, X.; Wang, X.; Li, H. High-Performance Machine Learning for Large-Scale Data Classification considering Class Imbalance. *Sci. Program.* **2020**, *2020*, 1953461. [CrossRef]

4. Tyagi, S.; Mittal, S. Sampling approaches for imbalanced data classification problem in machine learning. In Proceedings of the ICRIC 2019: Recent Innovations in Computing, Jammu, India, 20–21 March 2019; Springer: Berlin/Heidelberg, Germany, 2020; pp. 209–221.
5. Khushi, M.; Shaukat, K.; Alam, T.M.; Hameed, I.A.; Uddin, S.; Luo, S.; Yang, X.; Reyes, M.C. A comparative performance analysis of data resampling methods on imbalance medical data. *IEEE Access* **2021**, *9*, 109960–109975. [CrossRef]
6. Tran, N.; Chen, H.; Jiang, J.; Bhuyan, J.; Ding, J. Effect of class imbalance on the performance of machine learning-based network intrusion detection. *Int. J. Perform. Eng.* **2021**, *17*, 741.
7. Dablain, D.; Krawczyk, B.; Chawla, N.V. DeepSMOTE: Fusing deep learning and SMOTE for imbalanced data. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 6390–6404. [CrossRef]
8. Ayoub, S.; Gulzar, Y.; Rustamov, J.; Jabbari, A.; Reegu, F.A.; Turaev, S. Adversarial approaches to tackle imbalanced data in machine learning. *Sustainability* **2023**, *15*, 7097. [CrossRef]
9. Huang, L.; Lin, K.C.J.; Tseng, Y.C. Resolving intra-class imbalance for gan-based image augmentation. In Proceedings of the 2019 IEEE International Conference on Multimedia and Expo (ICME), Shanghai, China, 8–12 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 970–975.
10. Abayomi-Alli, O.O.; Damaševičius, R.; Qazi, A.; Adedoyin-Olowe, M.; Misra, S. Data augmentation and deep learning methods in sound classification: A systematic review. *Electronics* **2022**, *11*, 3795. [CrossRef]
11. Jamoos, M.; Mora, A.M.; AlKhanafseh, M.; Surakhi, O. A New Data-Balancing Approach Based on Generative Adversarial Network for Network Intrusion Detection System. *Electronics* **2023**, *12*, 2851. [CrossRef]
12. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.
13. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 1–6.
14. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and other botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
15. Mienye, I.D.; Sun, Y.; Wang, Z. Prediction performance of improved decision tree-based algorithms: A review. *Procedia Manuf.* **2019**, *35*, 698–703. [CrossRef]
16. Primartha, R.; Tama, B.A. Anomaly detection using random forest: A performance revisited. In Proceedings of the 2017 International Conference on Data and Software Engineering (ICoDSE), Palembang, Indonesia, 1–2 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
17. Mohammed, A.J.; Arif, M.H.; Ali, A.A. A multilayer perceptron artificial neural network approach for improving the accuracy of intrusion detection systems. *IAES Int. J. Artif. Intell.* **2020**, *9*, 609.
18. Gu, J.; Lu, S. An effective intrusion detection approach using SVM with naïve Bayes feature embedding. *Comput. Secur.* **2021**, *103*, 102158. [CrossRef]
19. Gulati, P.; Sharma, A.; Gupta, M. Theoretical study of decision tree algorithms to identify pivotal factors for performance improvement: A review. *Int. J. Comput. Appl* **2016**, *141*, 19–25. [CrossRef]
20. Pandey, M.; Sharma, V.K. A decision tree algorithm pertaining to the student performance analysis and prediction. *Int. J. Comput. Appl.* **2013**, *61*, 1–5. [CrossRef]
21. Winham, S.J.; Freimuth, R.R.; Biernacka, J.M. A weighted random forests approach to improve predictive performance. *Stat. Anal. Data Mining ASA Data Sci. J.* **2013**, *6*, 496–505. [CrossRef]
22. Schoppa, L.; Disse, M.; Bachmair, S. Evaluating the performance of random forest for large-scale flood discharge simulation. *J. Hydrol.* **2020**, *590*, 125531. [CrossRef]
23. Surakhi, O.M.; Zaidan, M.A.; Serhan, S.; Salah, I.; Hussein, T. An optimal stacked ensemble deep learning model for predicting time-series data using a genetic algorithm—An application for aerosol particle number concentrations. *Computers* **2020**, *9*, 89. [CrossRef]
24. Zaidan, M.A.; Surakhi, O.; Fung, P.L.; Hussein, T. Sensitivity Analysis for Predicting Sub-Micron Aerosol Concentrations Based on Meteorological Parameters. *Sensors* **2020**, *20*, 2876. [CrossRef]
25. Rish, I. An empirical study of the naive Bayes classifier. In Proceedings of the IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, Seattle, WA, USA, 4 August 2001; Volume 3, pp. 41–46.
26. Lowd, D.; Domingos, P. Naive Bayes models for probability estimation. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 529–536.
27. Maciá-Fernández, G.; Camacho, J.; Magán-Carrión, R.; García-Teodoro, P.; Therón, R. UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Comput. Secur.* **2018**, *73*, 411–424. [CrossRef]
28. Malhi, A.; Gao, R.X. PCA-based feature selection scheme for machine defect classification. *IEEE Trans. Instrum. Meas.* **2004**, *53*, 1517–1525. [CrossRef]