*electronics*

*Article*

# A New Data-Balancing Approach Based on Generative Adversarial Network for Network Intrusion Detection System

**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, Birzeit P.O. Box 14, West Bank, Palestine; malkhanafseh@birzeit.edu
[4] Department of Computer Science, American University of Madaba, Madaba 11821, Jordan
* Correspondence: o.surakhi@aum.edu.jo

**Abstract:** An intrusion detection system (IDS) plays a critical role in maintaining network security by continuously monitoring network traffic and host systems to detect any potential security breaches or suspicious activities. With the recent surge in cyberattacks, there is a growing need for automated and intelligent IDSs. Many of these systems are designed to learn the normal patterns of network traffic, enabling them to identify any deviations from the norm, which can be indicative of anomalous or malicious behavior. Machine learning methods have proven to be effective in detecting malicious payloads in network traffic. However, the increasing volume of data generated by IDSs poses significant security risks and emphasizes the need for stronger network security measures. The performance of traditional machine learning methods heavily relies on the dataset and its balanced distribution. Unfortunately, many IDS datasets suffer from imbalanced class distributions, which hampers the effectiveness of machine learning techniques and leads to missed detection and false alarms in conventional IDSs. To address this challenge, this paper proposes a novel model-based generative adversarial network (GAN) called TDCGAN, which aims to improve the detection rate of the minority class in imbalanced datasets while maintaining efficiency. The TDCGAN model comprises a generator and three discriminators, with an election layer incorporated at the end of the architecture. This allows for the selection of the optimal outcome from the discriminators' outputs. The UGR'16 dataset is employed for evaluation and benchmarking purposes. Various machine learning algorithms are used for comparison to demonstrate the efficacy of the proposed TDCGAN model. Experimental results reveal that TDCGAN offers an effective solution for addressing imbalanced intrusion detection and outperforms other traditionally used oversampling techniques. By leveraging the power of GANs and incorporating an election layer, TDCGAN demonstrates superior performance in detecting security threats in imbalanced IDS datasets.

**Keywords:** Generative Adversarial Network; Intrusion Detection System; imbalanced dataset; machine learning; unsupervised learning

check for updates

**Citation:** 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**, *1*, 0. https://doi.org/

Academic Editors: Dariusz Rzońca and Tomasz Rak

Received: 29 May 2023
Revised: 16 June 2023
Accepted: 17 June 2023
Published: 20 June 2023

## 1. Introduction

The process of data science comprises multiple stages, starting with the collection of a dataset, followed by its preparation and exploration, and eventually modeling the data to yield solutions. However, since different problem domains have varying datasets, the data-gathering process may uncover various issues within the dataset that must be addressed and rectified before proceeding with data modeling. Successfully handling these problems can significantly impact the model's accuracy.

One application where machine learning methods are widely used is intrusion detection systems (IDSs) [1,2]. IDS is employed to monitor the network traffic and identify any unauthorized efforts to access that network through the analysis of incoming and outgoing actions, with the aim of detecting indications of potentially harmful actions [3].

Machine learning (ML) methods, such as supervised network intrusion detection, have demonstrated satisfactory effectiveness in identifying malicious payloads within network traffic datasets that are annotated with accurate labeling. Nevertheless, the substantial growth in network scale and the proliferation of applications processed by network nodes have led to an overwhelming volume of data being shared and transmitted across the network. Consequently, this has given rise to significant security threats and underscored the urgency to enhance network security. As a result, numerous researchers have focused their efforts on enhancing intrusion detection systems (IDSs) by improving the detection rate for both novel and known attacks, while concurrently reducing the occurrence of false alarms (false alarm rate or FAR) [1]. Unsupervised intrusion detection techniques have emerged as a solution that eliminates the need for labeled data [4]. These methods can effectively train using samples from a single class, typically normal samples, aiming to identify patterns that deviate from the training observations. However, the accuracy of these unsupervised learning approaches tends to decline when faced with imbalanced datasets, where the number of samples in one class significantly exceeds or falls short of the number of samples in other classes.

To tackle the issue of imbalanced datasets, oversampling techniques are frequently employed. Traditional approaches utilize interpolation to generate samples among the nearest neighbors, such as the synthetic minority oversampling technique (SMOTE) [5] and the adaptive synthetic sampling technique (ADASYN) [6]. However, a novel generative model called the generative adversarial network (GAN) has emerged, providing a fresh framework for sample generation [7]. GAN allows the generator to effectively learn data features by engaging in a game-like interaction with the discriminator to simulate data distributions. GAN has demonstrated remarkable advancements in generating images, sounds, and texts [8–10]. As a result, researchers from various domains are increasingly incorporating this method into their research endeavors.

This paper proposes a new oversampling technique based on GAN applied for IDS considering the viewpoint of imbalanced data. The new model is called the triple discriminator conditional generative adversarial network (TDCGAN). This model consists of one generator and three discriminators with an added layer at the end for election. The TDCGAN employs a structure comprising a single generator and three discriminators. The generator utilizes random noise from a latent space as input and produces synthetic data that closely resemble real data, with the intention of evading detection by the discriminators. Each discriminator is a deep neural network with distinct architecture and parameter settings. Their primary task is to extract features from the generator's output and classify the data with varying levels of accuracy, differing for each discriminator. A new layer called the election layer is incorporated at the end of the TDCGAN architecture. This layer receives the outputs from the three discriminators and conducts an election procedure to determine the optimal outcome, selecting the result that achieves the highest classification accuracy. This process resembles an ensemble method, where multiple inputs are combined to produce a superior result. The generator model is designed as a deep multi-layer perceptron (MLP) comprising an input layer, an output layer, and four hidden layers. The initial hidden layer consists of 256 neurons, while an embedded layer is employed between the hidden layers to effectively map input data from a high-dimensional space to a lower-dimensional space. The second hidden layer comprises 128 neurons, followed by a third hidden layer with 64 neurons, and a final hidden layer with 32 neurons. The ReLU activation function is applied to all of these layers, and a regularization dropout of 20% is included to prevent overfitting. The output layer is activated using the Softmax activation function, with 14 neurons corresponding to the number of features in the dataset. Each discriminator within the TDCGAN architecture is implemented as an MLP model, featuring distinct configurations in terms of hidden layers, number of neurons, and dropout percentages. The first discriminator consists of three hidden layers, with each layer containing 100 neurons and a dropout regularization of 10%. The second discriminator includes five hidden layers with varying neuron counts—64, 128, 256, 512, and 1024—for each respective layer. A dropout percentage of 40% is applied in this case. The last discriminator is composed of four hidden layers with 512, 256, 128, and 64 neurons per layer, accompanied by a 20% dropout percentage. The LeakyReLU activation function with an alpha value of 0.2 is employed for the hidden layers in all discriminators. Two output layers are utilized for each discriminator, with the Softmax activation function applied to one output layer and the sigmoid activation function to the second output layer. The model is trained using two loss functions: binary cross entropy for the first output layer and categorical cross-entropy loss for the second output layer. The output from each discriminator is extracted and fed into the final layer of the model, where the election process takes place to determine the best result. The dataset [11] used in this paper to evaluate and test our model is the UGR'16 dataset. There are many datasets for IDSs, such as KDD CUP 99-1998, CICIDS2017, DARPA-1998 and more [12]; we chose UGR'16, because it is built with real traffic and up-to-date attacks.

This paper makes two main contributions. Firstly, it addresses the issue of high-class imbalance by analyzing the UGR'16 dataset. Secondly, it conducts evaluations on this dataset using several commonly used machine learning algorithms for data balancing.

The rest of this paper is organized as follows: Section 2 presents some of the relevant studies in this topic. Section 3 gives an overview about IDS and UGR'16 dataset. Section 4 proposes the TDCGAN model. The design, execution and results are given in Section 5. Finally, Section 6 gives the conclusion and future works.

## 2. Related Works

The impact of data resampling on machine learning model performance has been analyzed in multiple studies, since this issue can result in diminished predictive capabilities of the model.

The concept of employing GAN models to address the class imbalance problem is introduced by Lee and Park in reference [13]. In general, GAN is an unsupervised learning technique rooted in deep learning and generates synthetic data that closely resembles the existing data. The authors in this work used GAN to effectively tackle fitting issues, class overlaps, and noise through the process of resampling by explicitly defining the desired rare class. To evaluate the classifier's performance, the re-sampled data are trained using the widely adopted machine learning technique called random forest (RF). The proposed solution demonstrates superior performance compared to the methods currently utilized. Hajisalem and Babaie in the study referenced in [14] apply swarm intelligence optimization heuristics, specifically artificial fish swarm (AFS) and bee colony optimization (BCO), for the anomaly detection process. The detection approach proposed in that research focuses on reducing the subset of characteristics.

The study referenced in [15] presents a novel solution that applies an optimum allocation technique to efficiently manage large datasets by selecting the most representative samples. This approach aims to develop a new network intrusion detection system (NIDS) based on the least support vector machine (LSVM). The samples are arranged based on the desired confidence interval and the number of observations. The authors in [16] aimed to tackle the problem arising from the increasing quantity and diversity of network attacks, which leads to insufficient data during the training phase of machine learning-based intrusion detection systems (IDSs). The authors addressed this issue by examining a considerable number of network datasets from recent years. Each dataset's limitations, such as a shortage of attack instances and other issues, are identified. As a result, Kumar, et al. proposed a new dataset that aims to resolve, or at least alleviate, the encountered problem [17].

The authors introduced a new IDS system designed to address five common conventional attacks. In this solution, the author constructs a new dataset that surpasses the UNSW-NB15 dataset. A misuse-based strategy is employed to create a fresh dataset, and a gain information technique is applied to collect features from the original UNSW-NB15 dataset.

Another IDS solution based on GAN was proposed in [18]. Due to the limited number of known attack signatures for vehicle networks, the author employ the concept of generating unknown attacks during the training process to enable the IDS to effectively handle various types of attacks. In the context of vehicle IDS, accuracy is of the utmost importance to ensure driver safety, as any false positive error could have serious consequences. Traditional IDS approaches are inadequate for dealing with numerous new and undiscovered attacks that may arise. The proposed GAN-based IDS solution successfully detects four previously unknown attacks. The authors in [19] propose a novel method by combining ADASYN and RENN techniques. This approach aims to tackle the imbalances between negative and positive instances in the initial dataset, as well as addressing the issue of feature redundancy. The RF algorithm and Pearson correlation analysis are employed to select the most relevant features. In conclusion, the studies presented in this section cover various approaches and techniques for addressing challenges in machine learning-based intrusion detection systems (IDSs) and class imbalance problems. The introduction of GAN models in reference [13] offers a promising solution by generating synthetic data to tackle class imbalances, resulting in improved model performance. The utilization of swarm intelligence optimization heuristics, such as artificial fish swarm (AFS) and bee colony optimization (BCO), for anomaly detection as described in reference [14] focuses on reducing the subset of characteristics to enhance detection accuracy. Another study referenced in [15] introduced an innovative approach that efficiently manages large datasets for network intrusion detection systems (NIDS). Addressing the problem of insufficient data during the training phase of IDS, the study mentioned in [16] examined multiple network datasets and proposed a new dataset to alleviate the limitations caused by increasing network attacks. Overall, these studies contribute valuable insights and propose effective solutions to enhance the performance and capabilities of intrusion detection systems in the face of various challenges, such as class imbalance, limited data, and emerging attack types.

## 3. UGR'16 Dataset

In this paper, the UGR'16 dataset [11] is used to test the performance of the proposed model and achieve data balancing. The data are sourced from multiple netflow v9 collectors that are strategically positioned within the network of a Spanish ISP. An ISP is an Internet Service Provider. It provides access to Internet for many different hosts (most of them inside private networks, like homes or companies). The main aspects of the ISP network infrastructure are as follows:

- Netflow probes are set up on the outgoing network interfaces of two redundant border routers, BR1 and BR2, which enable access to the Internet. This configuration allows for the collection of all incoming and outgoing connections.
- The ISP has two different subnetworks. One is termed the core network, where the services that are not protected by a firewall are located. The second is the inner network, where firewall services are provided to the clients.
- At the highest level, there is a network of attacker machines consisting of five units, designated as A1–A5.
- Within the core network, five victim machines specifically for dataset collection purposes are set up. These machines, named V11–V15, are located alongside genuine clients in an existing network referred to as victim network V1.
- In relation to the inner network, a collective of 15 additional victim machines is positioned across three separate existing networks, with each network consisting of 5 machines. These networks are designated as victim network V2 (machines V21–V25), victim network V3 (machines V31–V35), and victim network V4 (machines V41–V45).

The entire dataset comprises two distinct sets: a calibration and a testing set. The calibration set is used in constructing and adjusting the machine learning models. This set contains attacks, but they are not controlled, nor labeled, and data that were recorded between March and June 2016. it includes inbound and outward ISP network traffic. The testing set, acquired in July and August of 2016, is used to evaluate the models in the detection process. For both the calibration and test sets, the collected files are consolidated into a single file per week for each of the two capture periods. These files are typically compressed tar files with an average size of approximately 14 GB. The calibration set consists of 17 files, while the test set comprises 6 files. To anonymize the IP addresses of the machines in the dataset, the CryptoPan prefix-preserving anonymization technique [20] was applied. This anonymization process is carried out using the nfanon tool [21]. Table 1 contains the list of different attacks with their corresponding labels in the UGR'16 dataset.

**Table 1.** List of attacks in UGR'16 dataset.

| Attack | Label Description | |
|---|---|---|
| DoS11 | DoS | One-to-one DoS (denial of service) attack, where the attacker A1 attacks the victim V21 |
| DoS53s | DoS | The five attackers A1–A5 attack three of the victims, each one at a different network |
| DoS53a | DoS | The attacks are executed as in DoS53s, but now every victim is sequentially selected |
| Scan11 | Scan11 | One-to-one scan attack, where the attacker A1 scans the victim V41 |
| Scan44 | Scan14 | Four-to-four scan attack, where the attackers A1, A2, A3 and A4 initiate a scan at the same time to the victims V21, V11, V31 and V41 |
| Botnet | Nerisbotnet | Mixing botnet captures recorded elsewhere in a controlled environment with our background traffic |
| IP in blacklist | Blacklist | It is an attack of class signature |
| UDP Scan | Anomaly-udpscan | Depending on the source port of the connection, each victim host is scanned through a specific range of 60 ports |
| SSH Scan | Anomaly-sshscan | An anomaly attack |
| SPAM | Anomaly-spam | An anomaly attack |

The artificial attack traffic was generated in 2h batches, during which all attack variants were executed. There are two possible scheduling patterns for the execution of the attack variants within each batch:

1. Planned scheduling: every attack within the batch is executed at a predetermined and known time, which is determined by an offset from the initial batch time, denoted as t0.
2. Random scheduling: the initial time for the execution of each of the attacks is randomly selected between t0 + 00h00m and t0 + 01h50m, thus restricting the total duration of the batch to a maximum of 2h.

The UGR'16 is created based on packet and flow data. It contains 16,900,000,000 anonymous network traffic flows. The network flow features are derived from actual network traffic, and these features are detailed in Table 2.

The UGR'16 dataset is divided into 23 compressed files, each of which is assigned to a particular week. Based on this, 16 of the files are assigned to the calibration class of datasets, and the remaining 6 to the test class. The size of each file is around 14 GB in the compressed format, and they can be downloaded in the csv format.

**Table 2.** UGR'16 dataset network flow features.

| Number | Feature Name | Type |
|---|---|---|
| 1 | Timestamp | date-time |
| 2 | Flow duration | continuous numeric |
| 3 | Source IP address | categorical |
| 4 | Source IP address | categorical |
| 5 | Source port number | discrete numeric |
| 6 | Destination port number | discrete numeric |
| 7 | Protocol | categorical |
| 8 | Flag | categorical |
| 9 | Forwarding status | numeric |
| 10 | Source type of service | discrete numeric |
| 11 | Total number of packets | Continuous numeric |
| 12 | Total number of bytes | Continuous numeric |
| 13 | Class (Label) | categorical |

## 4. Proposed Model

### *4.1. Data Preparation*

The UGR-16 dataset used in this paper contains 16.9 billion records. While the deep learning algorithms require high hardware resources, such as CPU, memory and GPU for data processing and training, a subset of data points that cover all types of normal and anomalous traffic from UGR'16 dataset was selected. The subset selection, which included all types of attacks, was conducted using specific measures to prevent imbalanced distributions and bias. The following measures were put in place:

- Stratified sampling: The subset selection process employed stratified sampling techniques to ensure proportional representation of each type of attack. This approach helped maintain a balanced distribution of attacks in the subset.
- Class balancing: Additional steps were taken to balance the representation of different attack types in the subset. This might include oversampling the minority classes or undersampling the majority classes to mitigate the imbalanced distribution.
- Randomization: To minimize any potential bias, randomization techniques were applied during the subset selection process. This ensured that the selection was not influenced by any specific order or predetermined biases.

By implementing these measures, the subset selection aimed to create a representative subset of attacks that avoided imbalanced distributions and potential biases, enabling a more reliable analysis of the dataset.

This subset was then pre-processed, including cleaning it from the missing values and removing the duplicate instances. The details of the selected subset are shown in Table 3.

**Table 3.** UGR'16 subset details.

| From | To | Class Label | Counts | Percentage |
|---|---|---|---|---|
| 27 July 2016 | 31 July 2016 | background | 197,185 | 98.5% |
| 27 July 2016 | 31 July 2016 | dos | 1169 | 0.6% |
| 27 July 2016 | 31 July 2016 | scan44 | 578 | 0.3% |
| 27 July 2016 | 31 July 2016 | blacklist | 545 | 0.3% |
| 27 July 2016 | 31 July 2016 | nerisbotnet | 227 | 0.1% |
| 27 July 2016 | 31 July 2016 | anomaly-spam | 170 | 0.1% |
| 27 July 2016 | 31 July 2016 | scan11 | 126 | 0.1% |

Within the context of network security, normal traffic tends to occur more often than malicious traffic, leading to imbalanced class proportions and an imbalanced dataset [22]. This poses a challenge for machine learning, as learning from imbalanced data is a common issue. In order to address this problem, one potential solution is to either undersample the majority class or oversample the minority classes.

In this paper, dataset records with class labels equal to the background are major. The other class labels are oversampled to obtain a balanced subset of the UGR'16 dataset. The original number of records and classes of the selected subset is given in Table 3.

Since machine learning algorithms work with numerical data, some features in the dataset need to be encoded: protocol, source IP, destination IP and class label. One-hot encoded is used to convert these features. The dataset is then scaled using MinMaxScaler from the Scikit-learn library to scale the values to the interval [0,1].

Random forest classifier is used to explore the features importance based on mean decrease in impurity (MDI). The calculation for a given feature's importance involves summing the number of splits that incorporate the feature across all trees, proportional to the number of samples that it splits. Figure 1 shows the highest numerical features of the UGR'16 dataset based on MDI value. In the proposed model, all the features are included in the process, being the most important feature is the Source_IP.

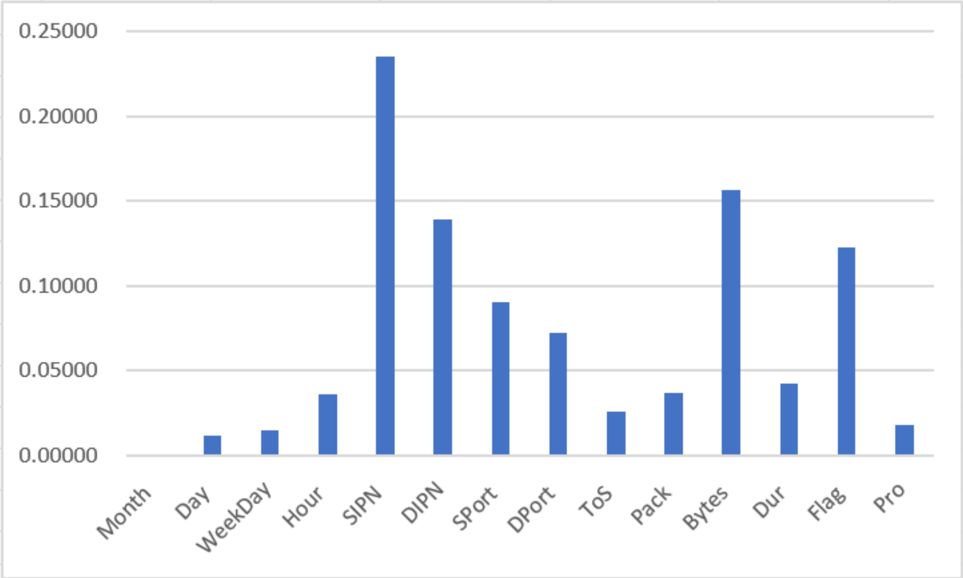

**Figure 1.** The highest numerical features of the UGR'16 dataset based on the mean decrease in impurity (MDI).

### *4.2. Setup of Proposed Model*

The generative adversarial network (GAN) is a machine learning-based deep learning method used to generate new data. It is an unsupervised learning task that involves learning from input data to produce new samples from the original dataset. GAN is used in the literature in many applications, such as computer vision [23], time-series applications [24], health [25] and more, making significant advancement and outperformance in data generation. As many improvements and versions for the GAN are proposed, in order to fit it with the application domain and increase the performance and model accuracy [26,27], this paper proposes a new version of GAN called triple discriminator conditional generative adversarial networks (TDCGANs) as an augmentation tool to generate new data for the UGR'16 dataset with the aim to restore balance in the dataset by increasing minor attack classes.

In the TDCGAN, the architecture consists of one generator and three discriminators. The generator takes random noise from a latent space as input and generates raw data that closely resemble the real data, aiming to avoid detection by discriminators. Each discriminator is a deep neural network with different architecture and different parameter settings. Each discriminator's role is to extract features from the output of the generator and classify the data with varying levels of accuracy for each them. An election layer is added to the end of TDCGAN architecture that obtains the output from the three discriminators and performs an election procedure to achieve the best result with the highest classification accuracy in a form of the ensemble method. The model aims to classify data into two groups: normal flows for the background traffic with 0 representation, and anomaly flows for the attack data with 1 representation. Additionally, in the case of anomaly flow, the model classifies it as its specific class type. Figure 2 shows the workflow of the proposed TDCGAN model. The setting details of generator and each discriminator are given below.

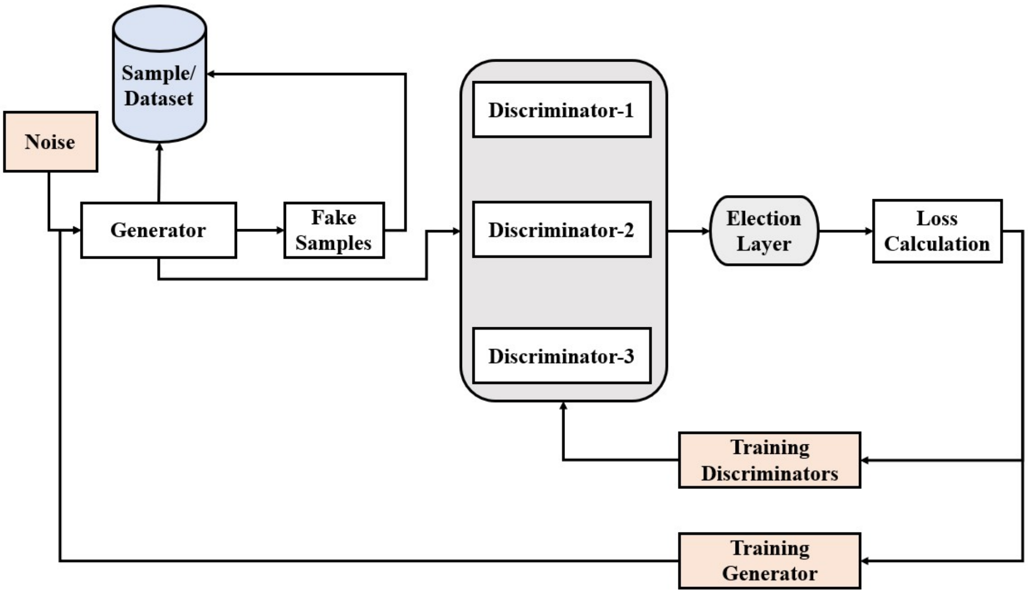

**Figure 2.** Workflow of TDCGAN model.

The model of the generator is a deep multi-layer perceptron (MLP) composed of an input layer, output layer and four hidden layers. Initially, the generator takes a point from the latent space to generate new data. The latent space is a multi-dimensional hypersphere normal distributed points, where each variable is drawn from the distribution of the data in the dataset. An embedded layer in the generator creates a vector representation for the generated point. Through training, the generator learns to map points from the latent space into specific output data, which are different each time the model is trained. Taken a step further, new data are then generated using random points in the latent space. So, these points are used to generate specific data. The discriminator distinguishes the new data generated by the generator from the true data distribution.

GAN is an unsupervised learning method. Both the generator and discriminator models are trained simultaneously [28]. The generator produces a batch of samples, which, along with real examples from the domain, are fed to the discriminator. The discriminator then classifies them as either real or fake. Subsequently, the discriminator undergoes updates to improve its ability to distinguish between real and fake samples in the subsequent round. Additionally, the generator receives updates based on its success or failure in deceiving the discriminator with its generated samples.

In this manner, the two models engage in a competitive relationship, exhibiting adversarial behavior in the context of game theory. In this scenario, the concept of zero-sum implies that when the discriminator effectively distinguishes between real and fake samples, it receives a reward, or no adjustments are made to its model parameters. Simultaneously, the generator is penalized with significant updates to its model parameters.

Alternatively, when the generator successfully deceives the discriminator, it receives a reward, or no modifications are made to its model parameters. Whereas, the discriminator is penalized. This is the generic GAN approach.

In the proposed TDCGAN model, the generator takes as input points from the latent space and produces data for the data distribution of the real data in the dataset. This is done through fully connected layers with four hidden layers, one input layer and one output layer. The discriminators try to classify the data into their corresponding class, which is done through a fully connected MLP network.

MLP has gained widespread popularity as a preferred choice among neural networks [29,30]. This is primarily attributed to its fast computational speed, straightforward implementation, and ability to achieve satisfactory performance with relatively smaller training datasets.

In this paper, the generator model learns how to generate new data similar to the minor class in the URG'16 dataset, while discriminators try to distinguish between real data from the dataset and the new one generated by generator. During the training process, both the generator and discriminator models are conditioned on the class label. This conditioning enables the generator model, when utilized independently, to generate minor class data within the domain that corresponds to a specific class label. The TGCGAN model can be formulated by integrating both the generator and three discriminators' models into a single, larger model.

The discriminators undergo separate training, where each of the model weights are designated as non-trainable within the TDCGAN model. This ensures that solely the weights of the generator model are updated during the training process. This trainability modification specifically applies when training the TDCGAN model, not when training the discriminator independently. So, the TDCGAN model is employed to train the generator's model weights by utilizing the output and error computed by the discriminator models.

Thus, a point in the latent space is provided as input to the TDCGAN model. The generator model creates the data based on this input, which is subsequently fed into the discriminator model. The discriminator then performs a classification, determining whether the data are real or fake, and in the case of fake data, the model classifies them to their corresponding type of attck.

The generator takes a batch of vectors (z), which are randomly drawn from the Gaussian distribution, and maps them to G(z), which has the same dimension as the dataset. The discriminators take the output from the generator and try to classify it. The loss is then evaluated between the observed data and the predicted data and is used to update the weights of the generator only to ensure that only generator weights are updated. The difference between the observed data and the predicted data is estimated using the cross-entropy loss function, which is expressed in the following equation:

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

where $y_i$ is the true label (1 for malicious traffic and 0 for normal traffic) and $p(y_i)$ is the predicted probability of the observation $(i)$ calculated by the sigmoid activation function. $N$ is the number of observations in the batch.

The generator model has four hidden layers. The first one is composed of 256 neurons with a rectified linear unit (ReLU) activation function. An embedded layer is used between hidden layers to efficiently map input data from a high-dimension to lower-dimension space. This allows the neural network to learn the data relationship and process it efficiently. The second hidden layer consists of 128 neurons, the third has 64 neurons and the last one has 32 neurons, with the ReLU activation function used with them all, and a regularization dropout of 20% is added to avoid overfitting. The output layer is activated using the Softmax activation function with 14 neurons as the number of features in the dataset.

After defining the generator, we define the architecture of each discriminator in the proposed model. Each discriminator is a MLP model with a different number of hidden layers, different number of neurons and different dropout percentage. The first discriminator is composed of 3 hidden layers with 100 neurons for each and 10% dropout regularization. The second has five hidden layers with 64, 128, 256, 512, and 1024 neurons for each layer, respectively. The dropout percentage is 40%. The last discriminator has 4 hidden layers with 512, 256, 128, and 64 neurons for each layer and 20% dropout percentage.

The LeakyReLU(alpha = 0.2) is used as an activation function for the hidden layers in the discriminators. Two output layers are used for each discriminator with the Softmax function as an activation function for one output layer and the Sigmoid activation function for the second output layer. The model is trained with two loss functions, binary cross entropy for the first output layer, and categorical cross-entropy loss for the second output layer. The output is extracted from each discriminator and is then fed to the last layer in the model, where the election is performed, to obtain the best result.

The TDCGAN model can be defined by combining both the generator model and the three discriminator models into one larger model. This large model is used to train the weights in the generator model, using the output and error calculated of the discriminators. The discriminators are trained separately by taking real input from the dataset.

The model is then trained for 1000 epochs with a batch size of 128. The optimizer is Adam with a learning rate equal to 0.0001. The proposed model allows the generator to train until it produces a new set of data samples that resembles the real distribution of the original dataset.

Nevertheless, this training strategy frequently fails to function effectively in various application scenarios. This is due to the necessity of preserving the relationships within the feature sets of the generated dataset by the generator, while the dataset used by the discriminator may differ from it. This disparity often leads to instability during the training of the generator.

In numerous instances, the discriminator quickly converges during the initial stages of training, thereby preventing the generator from reaching its optimal state. To tackle this challenge in network intrusion detection tasks, we adopt a modified training strategy, where three discriminators with different architectures are used. This approach helps preventing the early emergence of an optimal discriminator, ensuring a more balanced training process between the generator and discriminators.

### *4.3. Training Phase*

The primary objective of the training methodology employed in the GAN framework is for the generator to generate fake data that closely resemble real data, and for the discriminator to acquire sufficient knowledge to differentiate between real and fake samples. Both the generator and discriminator are trained until the discriminator can no longer distinguish real data from fake data. This mean that the generated network can estimate the data sample's distribution and achieve Nash equilibrium.

In order to assess the performance of our model with precision, it is customary to divide the data into training and test sets to produce accurate predictions on unseen data. The training set is utilized for model fitting, while the test set is employed to measure the predictive precision of the trained model. The dataset is split into 70% for training and validation and 30% for testing. The training set is divided into minor class data and other class data. The TDCGAN model uses the minor class to generate data. The generator is trained to model the distribution of the anomaly data (minor class), while fixing the discriminator. The output from the generator is fed as input to the discriminator to predict it. The error is estimated, and the generator's weight is then updated. The training continues until the discriminator cannot distinguish if the input data come from the generator's output or from the real anomaly dataset. In the training process, we make sure that all architectures undergo an equal number of epochs and that the weights from the final epoch are selected to generate artificial attack samples.

We begin by adhering to this iterative training procedure and ultimately utilize the generator to produce attack samples. Eventually, we incorporate the generated attack samples into the training set.

By this, we oversample minor classes in the dataset during the training phase. The test dataset is then used to test the model performance.

## 5. Experimental Results

Within this section, we methodically plan and execute a sequence of experiments, and subsequently analyze the obtained results.

### *5.1. Experimental Setup*

Our experiments were carried out on the Python Colab Jupyter notebook that runs in the browser with the integrated free GPUs and freely installed Python libraries. The system setup is shown in Table 4.

**Table 4.** System environment specifications.

| Unit | Description |
|---|---|
| Processor | Intel® Xeon® |
| CPU | 2.30 GHz with No.CPUs 2 |
| RAM | 12 GB |
| OS | |
| Packages | TensorFlow 2.6.0 |

### *5.2. Performance Metrics*

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

We utilize the metric of accuracy (Acc) to quantify the correct classification of data samples, considering all predictions made by the model as measured by the following equation:

$$Acc = (TP + TN)/(TP + TN + FP + FN) \tag{2}$$

where $TP$ is the true positive, which represents the number of truly predicted anomalies; $TN$ is the true negative, which indicates the number of truly predicted normal instances; $FP$ is the false positive indicator that denotes the number of normal instances that are incorrectly classified as anomalies; and $FN$ is the false negative indicator that indicates the number of the number of anomalies that are misclassified as normal.

Precision is employed to assess the accuracy of the correct predictions, calculated as the ratio of accurately predicted samples to the total number of predicted samples for a specific class as given in the following equation:

$$Precision = TP/(TP + FP) \tag{3}$$

Recall, which is known as the true positive rate ($TPR$), is used to determine the ratio of correctly predicted samples of a particular class to the total number of instances within the same class as given by the following equation:

$$TPR(Recall) = TP/(TP + FN) \tag{4}$$

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

$$F1 = 2 * ((Precision * Recall)/(Precision + Recall)) \tag{5}$$

### *5.3. Experimental Results and Analysis*

The performance of the TDCGAN model is evaluated on the testing dataset. The previous metrics are used to evaluate and compare the results. The results after training the TDCGAN model for URG’16 dataset balancing are given in Table 5.

**Table 5.** Performance evaluation metrics score for TDCGAN model.

| Accuracy | Precision | F1 Score | Recall |
|---|---|---|---|
| 0.95 | 0.94 | 0.94 | 0.96 |

Figure 3 shows the loss function while training the model for different numbers of epochs: 200, 400, 600, 800 and 1000.

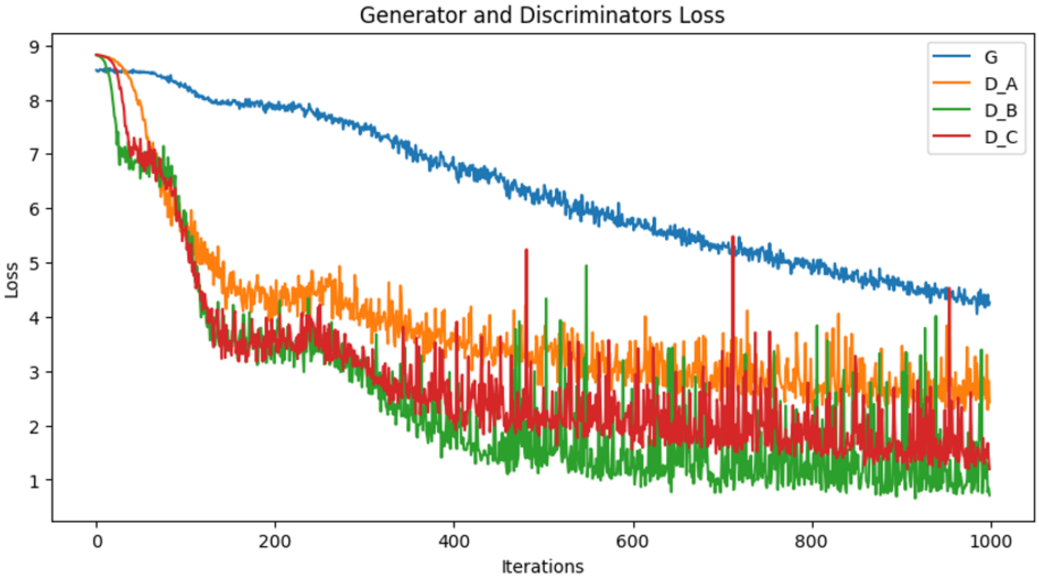

**Figure 3.** The loss function of G: generator, D_A: first discriminator, D_B: second discriminator and D_C: third discriminator in the TDCGAN model.

We compare the performance of the TDCGAN model for data balancing on the testing dataset with some resampling methods. The methods are as follows: (1) The synthetic minority oversampling technique (SMOTE) is a method for oversampling that produces artificial instances from the minor class. Its purpose is to create a training set that is either synthetically balanced or close to balance in terms of class distribution, which is subsequently utilized for classifier training. We use the implementations provided in the imbalanced-learn Python library, which provides a range of resample techniques that can be combined for evaluation comparison. (2) Random oversampling is used, which randomly duplicates the instances from the minor class. (3) Then, we combine SMOTE with edited nearest neighbor (ENN) SMOTEENN. (4) With Borderline-SMOTE (oversample technique using Borderline-SMOTE), the minority instances which are near the borderline are oversampled. (5) SVMSMOTE combines the support vector machine (SVM) with SMOTE. (6) We oversample using SMOTE-Tomek Links. Tomek Links denotes a technique used to detect pairs of closest neighbors within a dataset that exhibit dissimilar classes. Eliminating either one or both instances from these pairs, particularly those from the majority class, results in a reduction in noise or ambiguity within the decision boundary of the training dataset. (7) SMOTE_NC (synthetic minority over-sampling technique for nominal and continuous) is used to oversample data with categorical features. (8) CGAN (conditional generative adversarial network) is a conditional GAN that generates data under a conditional generation. Lastly, (9) CTGAN (conditional tabular generative adversarial networks) models tabular data using CGAN. The results are listed in Table 6 and shown in Figure 4.

**Table 6.** Performance evaluation metrics score for TDCGAN model and other resampling methods.

| Model | Accuracy | Precision | F1 Score | Recall |
|---|---|---|---|---|
| SMOTE | 0.88 | 0.86 | 0.87 | 0.91 |
| Random Oversampling | 0.85 | 0.89 | 0.90 | 0.88 |
| SMOTEENN | 0.86 | 0.89 | 0.90 | 0.89 |
| The Borderline SMOTE | 0.84 | 0.87 | 0.87 | 0.88 |
| SVMSMOTE | 0.89 | 0.90 | 0.91 | 0.89 |
| SMOTE-Tomek Links | 0.90 | 0.87 | 0.89 | 0.87 |
| SMOTE_NC | 0.85 | 0.88 | 0.86 | 0.85 |
| CGAN | 0.83 | 0.83 | 0.83 | 0.83 |
| CTGAN | 0.76 | 0.76 | 0.76 | 0.76 |
| TDCGAN | 0.95 | 0.94 | 0.94 | 0.96 |

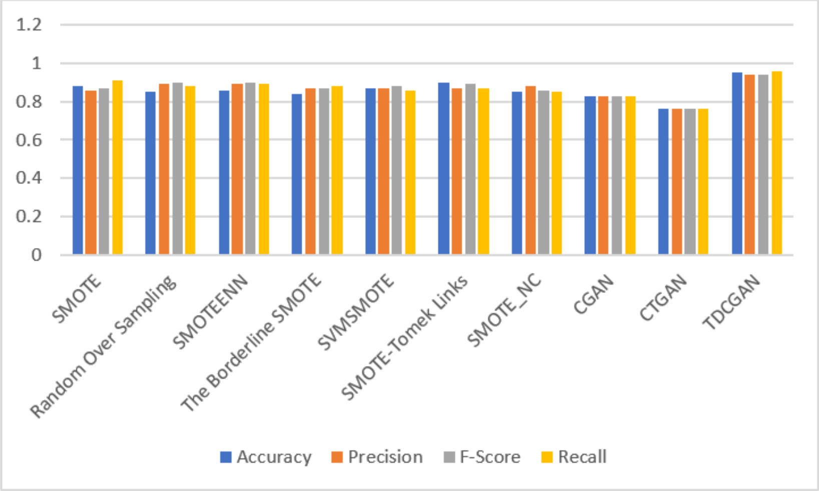

**Figure 4.** Performance evaluation metrics score for TDCGAN model and other resampling methods.

After conducting extensive experiments on UGR'16 dataset, our proposed model showcases its remarkable effectiveness in generating synthetic network traffic datasets, which in turn aids in the identification of anomalous network traffic. Through benchmarking, our model surpassed other similar generative models, achieving an impressive accuracy of over 0.95%.

## 6. Conclusions and Future Works

The imbalanced distribution of attacks in historical network traffic presents a significant challenge for intrusion detection systems (IDSs) based on traditional machine learning methods. These methods often struggle to effectively address the issue of imbalanced learning. In response, this paper introduces a novel technique called TDCGAN, a technology based on generative adversarial networks (GANs), specifically designed to tackle the problem of imbalanced datasets in IDS. The proposed TDCGAN model consists of a generator and three discriminators, all implemented using multi-layer perceptron (MLP) networks. This architecture allows the generator to generate synthetic data closely resembling real network traffic, while the discriminators aim to differentiate between genuine and attack traffic. To further enhance the TDCGAN framework, an additional layer is added at the end of the network to select the optimal outcome from the outputs produced by the three discriminators, enhancing the overall performance of the model. To evaluate the effectiveness of the proposed approach, the UGR'16 dataset, widely used in IDS research, is utilized for testing and evaluation purposes. A subset of the dataset is extracted and divided into training and testing sets. The experimental results showcase the outstanding performance of the proposed TDCGAN model across various evaluation metrics, including accuracy, precision, F1 score, and recall. Additionally, a comparison is made with other oversampling machine learning techniques, highlighting the superiority of the proposed method. While balancing datasets can be beneficial, it is important to note that it might not always be necessary or feasible, especially in cases where the class imbalance reflects the real-world distribution. In many real-world applications, the distribution of classes is often imbalanced. For instance, fraud detection, disease diagnosis, or rare event prediction typically involve imbalanced datasets. By balancing the dataset during training, the model learns to handle these imbalances and becomes more effective in addressing real-world scenarios. Additionally, balancing the dataset should be performed carefully to avoid introducing artificial patterns or losing valuable information from the original data.

As for future work, the proposed TDCGAN model shows promise for application in IDS within a vehicle ad hoc network (VANET) environment to detect unknown attacks. This opens up avenues for further research and development in leveraging the capabilities of TDCGAN for enhanced intrusion detection in dynamic vehicular networks.

**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 work was partially funded by projects PID2020-113462RB-I00, PID2020-115570GB-C22 and PID2020-115570GB-C21 granted by Ministerio Español de Economía y Competitividad; as well as project TED2021-129938B-I0, granted by Ministerio Español de Ciencia e Innovación.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Surakhi, O.M.; García, A.M.; Jamoos, M.; Alkhanafseh, M.Y. A Comprehensive Survey for Machine Learning and Deep Learning Applications for Detecting Intrusion Detection. In Proceedings of the 2021 22nd International Arab Conference on Information Technology (ACIT), Muscat, Oman, 21–23 December 2021; pp. 1–13.
2. AlKhanafseh, M.Y.; Surakhi, O.M. VANET Intrusion Investigation Based Forensics Technology: A New Framework. In Proceedings of the 2022 International Conference on Emerging Trends in Computing and Engineering Applications (ETCEA), Karak, Jordan, 23–24 November 2022; pp. 1–7.
3. Susilo, B.; Sari, R.F. Intrusion detection in IoT networks using deep learning algorithm. *Information* **2020**, *11*, 279. [CrossRef]
4. Schlegl, T.; Seeböck, P.; Waldstein, S.M.; Schmidt-Erfurth, U.; Langs, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In Proceedings of the Information Processing in Medical Imaging: 25th International Conference, IPMI 2017, Boone, NC, USA, 25–30 June 2017; Springer: Cham, Switzerland, 2017; pp. 146–157.
5. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
6. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328.
7. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets Advances in neural information processing systems. *arXiv* **2014**, arXiv:1406.2661.
8. Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4681–4690.
9. Su, H.; Shen, X.; Hu, P.; Li, W.; Chen, Y. Dialogue generation with gan. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
10. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2223–2232.
11. 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]
12. Abdulrahman, A.A.; Ibrahem, M.K. Toward constructing a balanced intrusion detection dataset based on CICIDS2017. *Samarra J. Pure Appl. Sci.* **2020**, *2*, 132–142.
13. Lee, J.; Park, K. GAN-based imbalanced data intrusion detection system. *Pers. Ubiquitous Comput.* **2021**, *25*, 121–128. [CrossRef]

14. Hajisalem, V.; Babaie, S. A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Comput. Netw.* **2018**, *136*, 37–50. [CrossRef]
15. Kabir, E.; Hu, J.; Wang, H.; Zhuo, G. A novel statistical technique for intrusion detection systems. *Future Gener. Comput. Syst.* **2018**, *79*, 303–318. [CrossRef]
16. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.
17. Kumar, V.; Sinha, D.; Das, A.K.; Pandey, S.C.; Goswami, R.T. An integrated rule based intrusion detection system: Analysis on UNSW-NB15 data set and the real time online dataset. *Clust. Comput.* **2020**, *23*, 1397–1418. [CrossRef]
18. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN based intrusion detection system for in-vehicle network. In Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, Ireland, 28–30 August 2018; pp. 1–6.
19. Cao, B.; Li, C.; Song, Y.; Qin, Y.; Chen, C. Network Intrusion Detection Model Based on CNN and GRU. *Appl. Sci.* **2022**, *12*, 4184. [CrossRef]
20. Fan, J.; Xu, J.; Ammar, M.H.; Moon, S.B. Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *Comput. Netw.* **2004**, *46*, 253–272. [CrossRef]
21. Haag, P. NFDUMP-NetFlow Processing Tools. 2011. Available online: http://nfdump.sourceforge.net (accessed on 16 June 2023).
22. Ndichu, S.; Ban, T.; Takahashi, T.; Inoue, D. AI-Assisted Security Alert Data Analysis with Imbalanced Learning Methods. *Appl. Sci.* **2023**, *13*, 1977. [CrossRef]
23. Wang, Z.; She, Q.; Ward, T.E. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–38. [CrossRef]
24. Jiang, W.; Hong, Y.; Zhou, B.; He, X.; Cheng, C. A GAN-based anomaly detection approach for imbalanced industrial time series. *IEEE Access* **2019**, *7*, 143608–143619. [CrossRef]
25. Yang, Y.; Nan, F.; Yang, P.; Meng, Q.; Xie, Y.; Zhang, D.; Muhammad, K. GAN-based semi-supervised learning approach for clinical decision support in health-IoT platform. *IEEE Access* **2019**, *7*, 8048–8057. [CrossRef]
26. Wang, X.; Guo, H.; Hu, S.; Chang, M.C.; Lyu, S. Gan-generated faces detection: A survey and new perspectives. *arXiv* **2022**, arXiv:2202.07145.
27. Xia, X.; Pan, X.; Li, N.; He, X.; Ma, L.; Zhang, X.; Ding, N. GAN-based anomaly detection: a review. *Neurocomputing* **2022**, *493*, 497–535. [CrossRef]
28. Durgadevi, M. Generative Adversarial Network (GAN): A general review on different variants of GAN and applications. In Proceedings of the 2021 6th International Conference on Communication and Electronics Systems (ICCES), Coimbatre, India, 8–10 July 2021; pp. 1–8.
29. 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] [PubMed]
30. Surakhi, O.; Serhan, S.; Salah, I. On the ensemble of recurrent neural network for air pollution forecasting: Issues and challenges. *Adv. Sci. Technol. Eng. Syst. J.* **2020**, *5*, 512–526. [CrossRef]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.