Computer Systems
Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2023.039550
Article

A Model Training Method for DDoS Detection Using CTGAN under 5GC Traffic

Yea-Sul Kim[1], Ye-Eun Kim[1] and Hwankuk Kim[2,*]

[1]Department of Electronics Information and System Engineering, Sangmyung University, Cheonan, 31066, Korea
[2]Department of Information Security Engineering, Sangmyung University, Cheonan, 31066, Korea
*Corresponding Author: Hwankuk Kim. Email: rinyfeel@smu.ac.kr
Received: 04 February 2023; Accepted: 10 April 2023; Published: 26 May 2023

Abstract: With the commercialization of 5th-generation mobile communications (5G) networks, a large-scale internet of things (IoT) environment is being built. Security is becoming increasingly crucial in 5G network environments due to the growing risk of various distributed denial of service (DDoS) attacks across vast IoT devices. Recently, research on automated intrusion detection using machine learning (ML) for 5G environments has been actively conducted. However, 5G traffic has insufficient data due to privacy protection problems and imbalance problems with significantly fewer attack data. If this data is used to train an ML model, it will likely suffer from generalization errors due to not training enough different features on the attack data. Therefore, this paper aims to study a training method to mitigate the generalization error problem of the ML model that classifies IoT DDoS attacks even under conditions of insufficient and imbalanced 5G traffic. We built a 5G testbed to construct a 5G dataset for training to solve the problem of insufficient data. To solve the imbalance problem, synthetic minority oversampling technique (SMOTE) and generative adversarial network (GAN)-based conditional tabular GAN (CTGAN) of data augmentation were used. The performance of the trained ML models was compared and meaningfully analyzed regarding the generalization error problem. The experimental results showed that CTGAN decreased the accuracy and f1-score compared to the Baseline. Still, regarding the generalization error, the difference between the validation and test results was reduced by at least 1.7 and up to 22.88 times, indicating an improvement in the problem. This result suggests that the ML model training method that utilizes CTGANs to augment attack data for training data in the 5G environment mitigates the generalization error problem.

Keywords: 5G core traffic; machine learning; SMOTE; GAN-CTGAN; IoT DDoS detection; tabular form; cyber security; B5G; mobile network security

1 Introduction

The main characteristics of 5G networks are hyper-connectivity, ultra-high speed, and ultra-low latency. This study focuses on the hyper-connectivity of 5G networks because it is the basis for building a large-scale IoT environment [1]. In addition, as of 2021, 5G networks had over 580 million 5G subscribers worldwide. The traffic on 5G networks generated by these large IoT environments and the increasing number of users is approximately 26,331 MB per person. With this amount of traffic, IoT devices have serious security vulnerabilities, and DDoS attacks targeting these devices are increasing. Therefore, research on automated intrusion detection technology, in which ML is applied to respond to IoT DDoS attacks in the 5G network environment, is being actively conducted.

The 3rd generation partnership project (3GPP), a 5G standardization organization, defined the Network Analytics Function (NWDAF) to provide network data analysis functions by utilizing artificial intelligence (AI) in the network core [2]. This approach serves to store and utilize the trained ML model and share the analysis results in the core network. This study aims to internalize ML models trained for IoT DDoS attack detection in 5G core (5GC) as a reference model for NWDAF. To this end, this study is concerned with 5GC traffic (among several other types of traffic) when considering the 5G environment. However, studying 5GC traffic is problematic in that insufficient datasets are publicly available for training ML models because of privacy concerns. Therefore, open-source software was used to directly construct a testbed, which is a virtual environment, for the collection of 5GC traffic by conducting simulation experiments using a dataset of a wired environment. In addition, feature extraction was performed based on protocol headers to detect DDoS attacks by ML models from the collected 5GC traffic. Therefore, the experimental data were processed and used in the experiment as 5GC traffic in a tabular form with both continuous and discrete data. Additionally, as the type of GAN used to augment the data, CTGAN, which can more accurately model features of discrete data such as those in tabular form, was selected and utilized [3,4].

Because of the network traffic characteristics, significantly fewer attack data are generated than normal. This characteristic means that the dataset of the wired network, which was used to train the ML model, also has a data imbalance problem. A model trained using a dataset with these characteristics would increase the possibility of misclassification when real data that differ from the training data are entered. In this case, a generalization error problem would occur [5]. However, attack data are becoming more complex and sophisticated over time. To this end, training methods with fewer generalization errors are important when training ML models. Therefore, using the available imbalanced attack data, we created synthesized attack data using CTGAN, and an experiment was conducted using these data to train the ML model. Then we compared the corresponding results with SMOTE of the oversampling technique to confirm the performance of each trained ML model.

In this study, we conducted comparative and analytical research to develop an effective method for training an ML model to reduce the generalization error by solving the problem of insufficient and imbalanced data consisting of tabular 5GC traffic. The contributions of this study are as follows. 5GC traffic was considered because the research objective is to internalize a training model that detects IoT DDoS attacks in 5GC. This required a 5G testbed, a virtual environment, to be constructed, which was used to conduct a simulation experiment to collect 5GC traffic. Features based on protocol headers were extracted using the collected traffic to detect DDoS attacks in network traffic. Therefore, we used 5GC traffic in tabular form for the experiment. Oversampling (SMOTE) and data augmentation (GAN-CTGAN) were used to solve the insufficient and imbalanced problem of 5GC traffic. The classification performance of the trained ML models was compared and analyzed in terms of the evaluation metrics and generalization error obtained for three comparable models (Baseline, SMOTE, and GAN-CTGAN). Because the attack data of 5GC traffic is imbalanced, the attack data were augmented using GAN-CTGAN. The experiment confirmed that the ML model training method using GAN-CTGAN, compared to Baseline and SMOTE, tended to alleviate the generalization error problem.

The rest of this paper is organized as follows. Section 2 introduces relevant existing studies. Section 3 provides details of the background of this study, the 5G environment, and two techniques (SMOTE, GAN-CTGAN) to solve the data imbalance problem. Section 4 describes the datasets and procedures for collecting 5GC traffic used in the study. Section 5 reviews the experimental results and differences from previous studies. Finally, Section 6 concludes the study and suggests future research.

2 Related Research

This section discusses research related to this study. First, the latest research on network security using AI is presented. Second, related work in which the data imbalance problem was solved using SMOTE is discussed. At the same time, we explore research that has utilized SMOTE in various research areas beyond network security. Finally, studies using GANs in the field of network security are introduced. A GAN is used to solve the data imbalance problem and various other problems.

2.1 AI-Related Network Security Research

Saheed et al. (2022) [6] proposed support vector machine (SVM)-based k-means and genetic algorithm (GA) models to improve the efficiency of the intrusion detection system (IDS). The dataset they used was NSL-KDD, and the experimental results showed a low false acceptance rate (FAR) and high accuracy of 99%.

Haq et al. (2022) [7] proposed two models, DNNBoT1 and DNNBoT2, which utilized the principal component analysis (PCA) process, model optimization, and a callback function in the learning process to detect IoT-based botnet attacks. Their dataset was N-BaIoT, and the experimental results confirmed excellent training and verification accuracy.

Rizvi et al. (2023) [8] proposed a model using the 1D-dilated causal neural network (1D-DCNN) to solve the problem of IDSs requiring significant computational resources and processing time. They used two datasets (CIC-IDS2017, CSE-CIC-IDS2018) to evaluate their proposed model and showed the accuracy of CIC-IDS2017 to be 99.7% and CSE-CIC-IDS2018 to be 99.98%.

The AI-related research described above aims to improve IDS efficiency. In addition, it can be seen that NSL-KDD, N-BaIoT, CIC-IDS2017, and CSE-CIC-IDS2018, which are various datasets for each study, are being used to propose and evaluate AI-related models.

2.2 SMOTE-Related Research to Address Data Imbalance

Ma et al. (2020) [9] combined the reinforcement learning of the adapted SMOTE and adversarial environment using reinforcement learning (AE-RL) to develop the AESMOTE algorithm for anomaly detection. The corresponding solution was obtained using the NSL-KDD dataset, and the performance of the proposed model surpassed that of the previous AE-RL.

Won et al. (2020) [10] compared oversampling (SMOTE and its variants) and data augmentation using the two-side-channel protected advanced encryption standard (AES). Their experimental results indicated that adapting SMOTE variants could increase the attack's efficiency in general. Additionally, the proposed model achieved key recovery in the AS-CAD dataset (desync = 100) with 180 traces. This result was a 50% improvement from the latest technology.

Karatas et al. (2020) [11] used SMOTE in an ML-based IDS and proposed a model for the synthesis data of the minor class. These increased the data in the minor class to the average amount of data, and a significant increase in the detection rate for seldom-occurring trespassing was observed.

Zhang et al. (2020) [12] proposed a new SGM model. They tested the model performance and training rate by designing a mixed Gaussian model comprising SMOTE, undersampling, and convolutional neural network (CNN). The proposed model was validated using UNSW-NB15 and CICIDS2017. They achieved detection rates of 99.74% (binary) and 96.54% (multi) for UNSW-NB15 and a detection rate of 99.85% (15-classification) for CIC-IDS2017.

Haq et al. (2022) [13] proposed SMOTEDNN to classify and predict air pollution. The method was developed to solve the problem of data imbalance and non-optimization in classification models. The proposed model was developed using a dataset of monitored air pollutant levels in India released by NAMP, and the accuracy was 99.9%.

Dablain et al. (2022) [14] proposed DeepSMOTE, a SMOTE-based encoder/decoder framework for oversampling and artificial instance generation for images with imbalanced and high-dimensional data. They utilized five datasets (MNIST, Fashion-MNIST, CIFAR-10, SVHNs, and CelebA), and the performance results showed that their proposed method could generate artificial images of superior quality, outperforming GAN-based oversampling.

Joloudari et al. (2022) [15] conducted a deep learning (DL)-based sampling comparison experiment to solve the imbalanced data problem. They used three datasets (KEEL, breast cancer, Z-Alizadeh Sani). The SMOTE-based CNN model delivered the best performance, which achieved 99.08% accuracy on the imbalanced dataset.

As such, seven studies [9–15] related to SMOTE addressed the issue of data imbalance. The scope of this research is not limited to network security [9–12], which is associated with this work but also covers a wide range of studies, such as air pollution and imagery [13–15].

2.3 GAN-Related Network Security Research

Hu et al. (2023) [16] proposed a GAN-based algorithm called MalGAN, consisting of a substitute detector and generative network, to generate adversarial malware cases that bypass the black-box ML-based detection model. Furthermore, the experiment trained the generative network to minimize the probability of generating adversarial malware predicted by the substitute detector. Consequently, they confirmed that the performance of MalGAN reduced the detection rate to near zero and could render retraining-based defense methods against adversarial examples challenging to operate.

Ahmed et al. (2020) [17] proposed a model for generating uniform resource locator (URL) based phishing cases using GANs. The GAN structure used in this approach consisted of a generator network, discriminator network, and black-box phishing detector. The performance results demonstrated that the GAN network could effectively create adversarial phishing examples to thwart simple and sophisticated ML phishing detection models. Therefore, their study confirmed the need to create countermeasures in response to adversarial examples.

Abdelaty et al. (2021) [18] proposed an adversarial training approach called GADoT, which utilizes GANs to generate adversarial DDoS samples for robust model training. The dataset was detected using DDoS samples (SYN and HTTP Flood), and the classifier used was LUCID, an ML-based model. The model trained with GADoT detected all DDoS attacks with high accuracy and achieved an f1-score of at least 98% and a false negative rate (FNR) of up to 1.8% as evaluation indicators.

Guo et al. (2021) [19] proposed a GAN-based traffic augmentation (TA-GAN) framework for unbalanced traffic classifications. Their approach involved pre-training and tripartite learning processes and used a 1D-convolutional neural networks (1D-CNN) model as a classifier. The binary classification performance of their framework yielded an f1-score of up to 14.64% higher for the minority class owing to the influence of traffic imbalance.

Nan et al. (2022) [20] proposed an effective anomaly detection framework using GAN and long short-term memory (LSTM) neural networks in 5G environments. This framework analyzes the correlation between user actions to scale datasets and improves the performance without altering the data distribution. The experimental results confirmed an accuracy of 97.16% and a false positive rate (FPR) of 2.30% by utilizing the correlation between user actions and dataset expansion.

Kim et al. (2021) [21] proposed a network anomaly detection method that solves the problem of acquiring labeled network traffic data in 5G networks. Their approach was based on a GAN with scaling properties that adopt unsupervised learning. UNSW-NB15 was used as the dataset to evaluate the framework performance and utilized a CNN as a classifier. The performance evaluations confirmed a maximum accuracy of 99.30% and an f1-score of 0.99.

Park et al. (2022) [22] used boundary equilibrium GAN (BEGAN) and the autoencoder-driven DL model to develop an AI-based network intrusion detection system (NIDS) with performance reinforced by creating new synthesis data. They used four datasets (NSL-KDD, UNSW-NB15, IoT, and Real Data) to evaluate the performance of the proposed model. The classification of experimental results was superior to that of previous methods.

The above studies are ML and DL-based security-related studies using GAN in 5G and wired network environments. Currently, researchers are focusing on GANs rather than SMOTE, in which GANs are used for various purposes as well as data imbalance processing. In addition, previous research has shown that different types of GANs are being utilized depending on the type of data being augmented. This study used GAN to solve the imbalanced nature of malicious data. In addition, since the data types used are tabular, CTGAN, which generates attack network data that are effectively synthesized, was utilized [4].

3 Background

3.1 5G Network Environment

5G wireless network technology succeeded 4th generation mobile communication (4G), and its commercialization proceeded from 2018. The visions and objectives of 5G were presented by the international telecommunication union (ITU), and the development of a technological standard is underway at the international standardization organization 3GPP [23]. 3GPP defines the 5G network as a service-based architecture (SBA), which provides service by dispersing and virtualizing at the network function (NF) unit and by connecting at the service-based interface (SBI) [24]. The following sections introduce the composition of 5G in three parts and the User Plane Function (UPF) and NWDAF as two NFs of 5G that are significantly relevant to the work presented in this paper.

3.1.1 3 Parts of 5G: RAN, DN, and Core

Fig. 1 shows a diagram of the 5G network, with the core inside and the base station and data network (DN) outside. 5G network has three major components: NG-RAN (RAN), DN, and core. The RAN further comprises the base station gNodeB (gNB) and user equipment (UE), i.e., the smartphone. The user is provided network service by being connected to the core using the N3 interface of gNB. DN is connected to UE using the N6 interface of the core and delivers the data to the user.

The core can be divided into the user plane (UP), responsible for data transfer, and the control plane (CP), which controls the user end. The core provides services to all the NFs connected to the SBI.

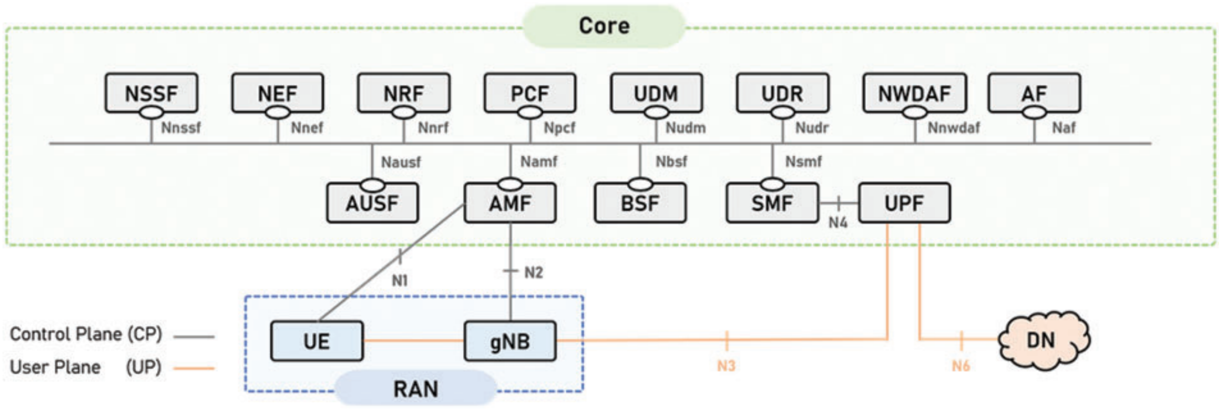

Figure 1: Three major components of 5G: RAN, DN, and core

3.1.2 UPF and NWDAF

Among the NFs used in this research, the UPF and NWDAF are the most relevant. The UPF was the NF that mainly comprised the 5G dataset in our research. In addition, NWDAF is an NF for internalizing security analysis functions using ML in 5GC, which we are studying and analyzing.

The 5G dataset was established using the N3 interface connected to UPF. The UPF is an NF that handles UP, which is responsible for packet routing and forwarding, packet inspection, quality of service (QoS) handling, and external protocol data unit (PDU) sessions and also interconnects DN [25]. UPF has three reference points, as shown in Fig. 1: N6, N4, and N3 [24]. N4 is the interface between Session Management Function (SMF) and UPF, N6 is DN and UPF, and N3 is gNB and UPF. The N3 interface undergoes encapsulation that expands the GPRS tunneling protocol (GTP) header in the previous internet protocol (IP) header through the GTP. GTP is a protocol for encapsulating and tunneling the IP packet data provided and received by the user in DN. The experiments to construct the 5G dataset utilized the N3 interface. As a result, Fig. 2 shows a structure where the UPF and RAN are connected via the N3 interface. In addition, as an experiment, IP-based packets were dumped to UEs in the RAN and changed to GTP-based packets in the 5G Network (5GC) as they passed through the N3 interface. We utilized packets from this modified 5G core in our experiments.

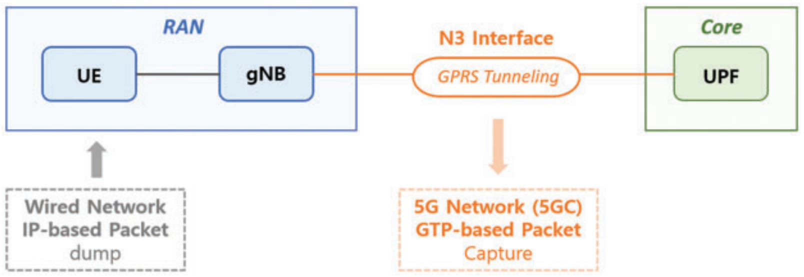

Figure 2: 5G dataset composition method for the experiment—5G network (5GC) GTP-based packet capture using N3 interface (gNB-UPF)

To provide security analysis functions to the 5GC, NWDAF, which has the ML function among the 5GC NFs, was studied and analyzed. The current 3GPP standardization defines NWDAF. The major function of NWDAF is to use ML and provide analysis results of the network data [2]. Fig. 3 shows the three steps NWDAF takes to deliver the derived network analysis results: (1) collect network data, (2) analyze network data using ML, and (3) provide the analysis results. In addition, the NWDAF structure consists of the Model Training Logical Function (MTLF) and the Analytics Logical Function (AnLF). MTLF trains the ML models and provides the trained ML models. AnLF performs inferences, derives analytics information, and exposes analytics services. As such, NWDAF of 5GC trains the ML models and uses the trained models to provide the derived network data analysis services. Therefore, we aimed to study the ML training method for detecting IoT DDoS attacks by using and collecting the 5GC traffic to provide the security analysis function using the ML model that was trained in 5GC, similar to NWDAF.

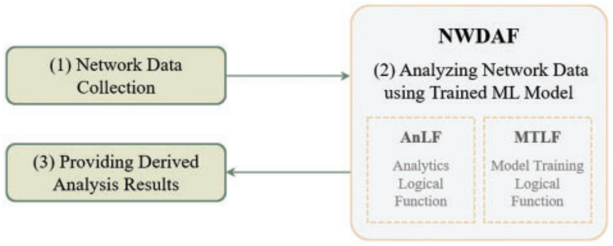
(1) Network Data Collection
(3) Providing Derived Analysis Results
NWDAF
(2) Analyzing Network Data using Trained ML Model
AnLF
Analytics Logical Function
MTLF
Model Training Logical Function

Figure 3: Procedure used by NWDAFs for providing derived network data analysis results

3.2 Techniques for Resolving Data Imbalance Problems

Several techniques exist to resolve the dataset imbalance issue, which degrades the model performance when training the ML model. In this research, oversampling and data augmentation were selected as two methods for solving the problem of imbalanced 5GC network traffic data [26,27]. Specifically, the experiment used SMOTE for oversampling and GAN-CTGAN for attack data augmentation.

3.2.1 SMOTE for Oversampling

Oversampling is the most widely used technique for overcoming network traffic imbalance due to the intuitive principle that rewards imbalance by increasing the number of traffic samples that belong to the minority class [9–15]. This study used SMOTE, which is the most usual oversampling method. SMOTE generates new data by copying virtual data in the minority class through the distance-based rule. Among previously reported oversampling methods, SMOTE improved overfitting as the disadvantage of random oversampling (ROS) [19]. However, SMOTE does not consider the majority class because new data is generated among the minority class data. Thus, the generated data only reflect the properties of data in the minority class. As a result, they may be prone to noise and unable to augment high-dimensional data efficiently. This research employed scikit-learn, an ML open-source library for SMOTE [28].

3.2.2 GAN-CTGAN for Attack Data Augmentation

Data augmentation is a technique that increases the number of data by applying various transformations to the original data to create new data [29]. This study used GAN, a popular method for data augmentation. GAN, which uses DL for augmenting data, comprises a generator that generates data and a discriminator that classifies the generated data. The two have a feedback relationship; the generator generates data that are improved by learning through feedback provided by the discriminator. GAN is being actively researched and is used for data diversification. Data can be distinguished mainly into sequential and tabular forms. This study used tabular 5GC traffic data to train the ML model, and a CTGAN validated for generating tabular network data was employed [4,30,31]. In particular, this study attempted to solve the data imbalance problem by generating attack data. To use CTGAN, we leveraged a related paper and open-source published on GitHub [32,33].

4 Experimental Design

In this work, SMOTE and GAN were used as methods of training the ML model to detect IoT DDoS attacks in the 5GC environment. Experiments were designed to evaluate the performance of the trained model. First, the experimental procedures and the methods used for collecting 5GC traffic for experimental use are introduced.

4.1 Experimental Dataset for 5GC Traffic

To collect 5GC traffic for the experiments, we need (1) a dataset from the wired network environment and (2) a 5G testbed. We used the GPRS tunneling experiment of the wired network environment dataset because the 5G testbed using this dataset could generate and gather 5GC traffic data.

4.1.1 Kitsune Dataset (Wired Network)

The kitsune dataset, which is openly offered in Kaggle, was used as the wired network environment dataset to generate and gather 5GC traffic [34,35]. Kitsune consists of nine DDoS attack datasets generated by IoT equipment in a wired network environment. The nine types of DDoS attacks are as follows: OS Scan, Fuzzing, Video Injection, ARP MitM, Active Wiretap, SSDP Flood, SYN DoS, SSL Renegotiation, and Mirai botnet. These attack types were used, excluding two (Video Injection and Mirai botnet). The exclusion was because Video Injection was the only type that included the Ethernet class's logical-link control (LLC) protocol. The Mirai botnet had 3.4 times fewer attack packets compared with the average of the eight other types.

Another reason for selecting kitsune for this study is that it was suitable for conducting the GPRS Tunneling experiment using the 5G testbed, as the provided data formats were diverse. Kitsune offers the following three data formats for the nine types of attacks: the dataset with the original packet pre-processed (.csv), a labeled dataset indicating the malignity or normality of the packet (.csv), and captured dataset of the original network (.pcap). The original network dataset (.pcap) and labeled dataset (.csv) were used for gathering 5GC traffic.

4.1.2 5G Testbed

Building the 5G testbed. The 5G testbed was constructed in a Linux 18.04 environment. To build the 5G testbed, we leveraged two open-source software programs. The open-source utilized UERANSIM for RAN and Open5GS for the core. Finally, we built a testbed with 5G RAN (UERANSIM) + 5G Core (Open5GS). UERANSIM is the simulator of 5G UE and gNB. UERANSIM is open-source software available on GitHub and can be used to test the core and study the 5G system. In addition, it is compatible with Open5GS among the 5GS open projects [36]. Open5GS implements core according to 3GPP release 16, and it was developed in the C language. Open5GS is available on GitHub, offering 10 NFs as of September 22, 2022. Active development updates have been underway for the corresponding project since September 22 [37].

Experiments on 5G testbed. Using the constructed 5G testbed, an experiment for collecting the 5GC traffic was performed. First, the packet was dumped with the UE, and the GTP packet that comprised GPRS Tunneling was captured as it passed through the gNB-UPF (N3 interface). Thus, in the experiment, we transformed the IP-based packet of the wired network into the GTP-based packet of the 5G network. Then, to capture the corresponding packet, Linux tcpdump was used.

Feature extraction. We extracted GTP-based packets, which are 5GC traffic obtained from our experiments, and converted them to CSV for use in the ML model. Table 1 summarizes information about all 57 features used in this experiment. The features are extracted based on the six network protocols used in the collected dataset. The six protocols are Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Group Management Protocol (IGMP), IP, and GTP. Wireshark's Tshark was used for feature extraction. The six protocols of the gathered packet were used as the criteria for extracting the packet according to the header. Considering the headers of these protocols, 93 pitchers were extracted, and the pitchers that did not have values were removed after checking the presence of values for all data from the extracted headers.

Table 1: Extracted feature type according to the protocol header

Protocol	Count	Feature
ICMP	4	icmp.type, icmp.code, icmp.checksum, icmp.checksum.status
TCP	17	tcp.port, tcp.seq, tcp.seq_raw, tcp.ack, tcp.ack.nonzero, tcp.ack_raw
		tcp.flag.res, tcp.flags.urg, tcp.flags.ack, tcp.flags.syn, tcp.flags.fin
		tcp.flags, tcp.flags.push, tcp.window_size, tcp.checksum
		tcp.urgent_pointer, tcp.checksum.status
UDP	5	udp.srcport, udp.dstport, udp.checksum, udp.checksum.status, udp.length
IGMP	5	igmp.type, igmp.max_resp, igmp.checksum, igmp.checksum.status
		igmp.num_grp_recs
IP	10	ip.version, ip.len, ip.id, ip.flags, ip.frag_offset, ip.ttl, ip.proto, ip.checksum
		ip.src, ip.dst
GTP	16	gtp.ip.src, gtp.ip.dst, gtp.flags, gtp.flags.version, gtp.flags.payload
		gtp.flags.reserved, gtp.flags.e, gtp.flags.s, gtp.glags.pn, gtp.message
		gtp.length, gtp.teid, gtp.ext_hdr.next, gtp.ext_hdr.length
		gtp.ext_hdr.pdu_ses_con.pdu_type, gtp.ext_her.pdu_ses_con.qos_flow_id

4.2 *Experimental Procedure*

This section describes the experimental procedure. In the experimental procedure, we first review the workflow to understand the flow and procedure of the experiment and then describe the procedure in detail based on that workflow.

4.2.1 Workflow

Fig. 4 is the workflow for this experiment, showing the flow and procedure. It had five procedures, and each procedure is explained as follows.

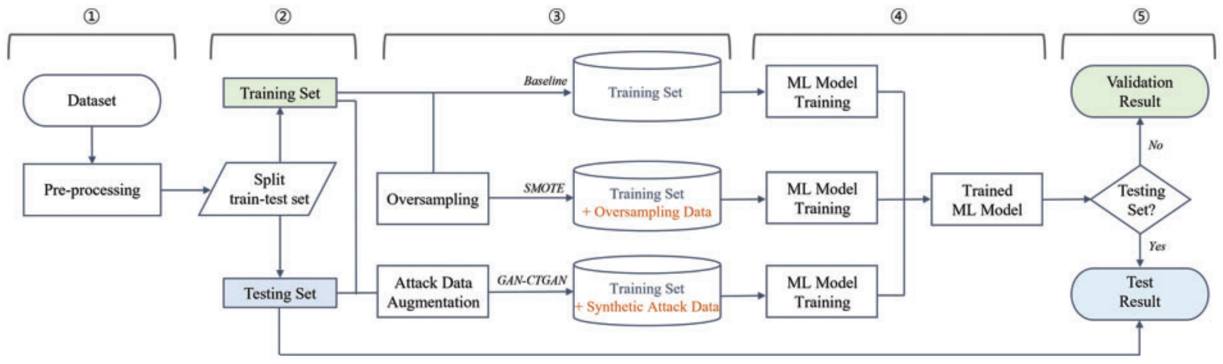

Figure 4: Workflow of experiment

(1)~(2). After pre-processing the dataset, divide it into training-testing sets.

(3). Configure the training dataset to create three comparison models (Baseline, SMOTE, and GAN-CTGAN).

(3)-a. Baseline is used as the training set.

(3)-b. SMOTE is used for training by oversampling in the training set. The reason for excluding the testing set from oversampling is that the sample data replicated by SMOTE is not included in the testing set for model performance verification.

(3)-c. GAN-CTGAN is used for training by augmenting malicious data by utilizing the training-testing set and including the synthesized malicious data in the training set. The reason for using the testing set for malicious data augmentation is that we have a small dataset to experiment with, so we want to utilize as much data as possible to generate plausible synthetic malicious data.

(4). Train the ML model using the training set for the three comparison models.

(5). For the trained ML model, validation and test results are derived depending on the presence or absence of the testing set.

4.2.2 Detailed Procedure

This time, we describe the procedures in the workflow in more detail, such as the methods used, data counts, and algorithms. Fig. 5 showed five detailed experimental procedures: (1) pre-processing, (2) splitting the training and testing sets, (3) three comparative models (Baseline, SMOTE, and GAN-CTGAN), (4) binary and multi-classification, and (5) model evaluation.

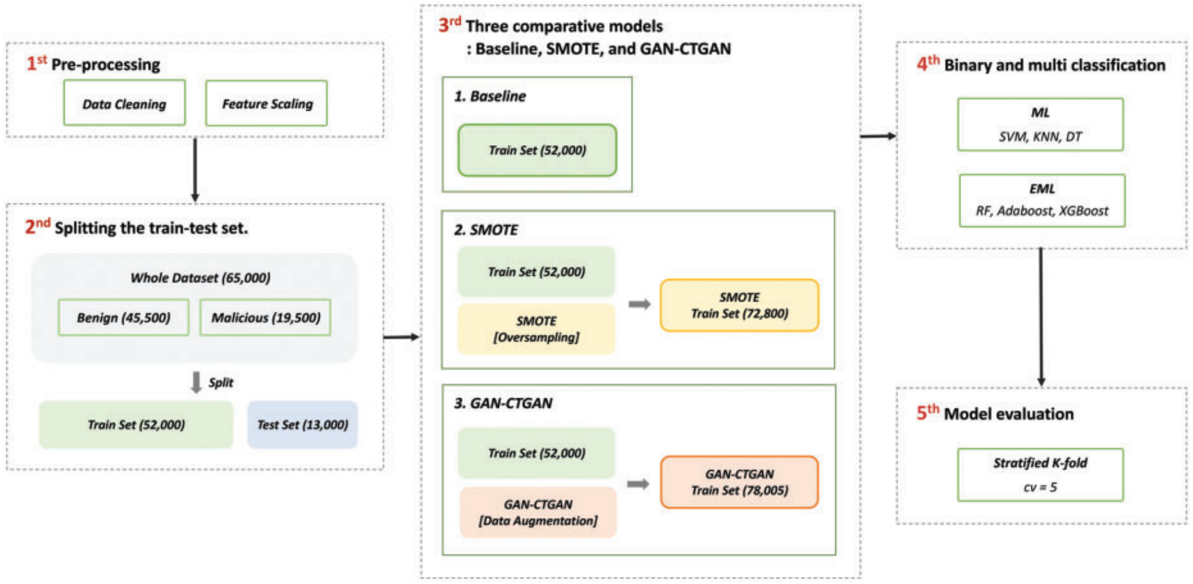

Figure 5: Experimental detailed procedure

Pre-processing. Pre-processing involves data cleaning and feature scaling. Data cleaning entailed the elimination of missing values by inserting the value 0 where no value existed. Additionally, we transformed hexadecimal numbers into decimal numbers and floats into integers. The feature scaling then went through standardization and normalization to bring the values to a consistent level.

Splitting the training and testing sets. To evaluate the performance of the ML training model in this experiment, we separate the training and testing sets. The total size of the dataset for the experiment was 65,000, and the kitsune dataset rate (7:3) for the (benign/malicious) ratio was maintained (45,500/19,500). Finally, the (benign/malicious) rate was maintained while the data were divided into the training and testing sets (52,000 and 13,000 data, respectively). To ensure that the ML model underwent effective training on the imbalanced class dataset, the training and testing set both maintained their (benign/malicious) ratios.

Three comparative models: Baseline, SMOTE, and GAN-CTGAN. This study analyzed and compared the performance of the three models. The models can be distinguished by their ML model training methods. First, the ML model was trained with the Baseline training set (52,000). Second, SMOTE was oversampled in the training set (52,000), and the ML model was trained. Lastly, the 26,005 attack data were augmented for GAN-CTGAN using the seven types of attack data, with 3,715 data per session. Thus, we added 26,005 data were added via augmentation to the original training set (52,000) and used 78,005 data to train the ML model.

Binary and multi-classification. The kitsune dataset was used with multiple attacks to experiment on binary and multi-classification. First, the data were divided into benign (label 0) and maligned (label 1) for binary classification. Subsequently, multi-classification was performed by dividing the data into benign (label 0) and seven types of attacks (labels 1–7). In addition, ML and the ensemble of machine learning (EML) were used as classifiers. First, ML classifiers were used for SVM, k-nearest neighbor (KNN), and decision tree (DT). Then, EML classifiers were used for random forest (RF), AdaBoost, and XGBoost.

Model evaluation. The stratified k-fold (cv = 5) was used to train the model and evaluate the performance because the dataset for training the ML model was insufficient and had imbalanced classes. In addition, these performed fivefold cross-validation and then generated five trained models. Finally, we utilized the validation set for the five trained models to get five validation results and averaged them to get the final validation result. We also got the same test results.

The confusion matrices used in this study were accuracy and f1-score. Accuracy intuitively refers to model prediction performance. Its definition is expressed as Eq. (1). However, it is unsuitable for use as a metric for imbalanced datasets. For this study, we also selected the f1-score, as it is highly prone to data bias due to the imbalanced classification problem. F1-score is the harmonic mean of recall and precision. Recall is a metric showing the percentage of positives correctly classified as true positives. Its definition is expressed as Eq. (2). precision is a metric showing the percentage of negatives correctly classified as true negatives. Its definition is expressed as Eq. (3). Therefore, the f1-score can cancel out differences even if the ratio of positives to negatives is imbalanced. Its definition is expressed as Eq. (4).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1} - \text{score} = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \tag{4}$$

5 Results and Discussions

This section presents the results of the experiments. First, the performance of the binary and multi-classification of the Baseline is presented for problem definition and verification. Next, the performance results for the three trained models with binary classification are compared. The results for the three models are evaluated from the perspective of the generalization error. A comparison with previous research revealed differences.

5.1 Experiment Results

5.1.1 Problem Definition and Proof with Baseline

A model that does not use either SMOTE as the Baseline oversampling technique or GAN-CTGAN as the data augmentation technique was referenced for comparison. We compared the Baseline's binary and multi-classification performance results to define and prove the problem. Table 2 shows the validation results, test results, and two value differences for binary and multi-classification. In addition, the results are shown as accuracy and f1-score of six ML classification models. For binary classification, the accuracy and f1-score decreased. The accuracy had a minimum reduction of 5.95% (DT) and a maximum value of 14.89% (AdaBoost), and the f1-score had a minimum value of 16.08% (DT) and a maximum value of 54.72% (SVM). For multi-classification, the accuracy increased, with a range of 1.74–3.51%, excluding KNN and AdaBoost. Additionally, the f1-score increased, with a minimum value of 0.06% (RF) and a maximum value of 3.17% (DT), excluding KNN.

Table 2: Acc and f1 for the validation and test results for binary and multi-classification

Classifier		Acc/F1	Binary			Multi		
			Validation	Test	Difference	Validation	Test	Difference
ML	SVM	Acc	85.47	70.65	(−14.82)	83.54	87.05	(+3.51)
		F1	77.11	22.39	(−54.72)	74.72	76.51	(+1.79)
	KNN	Acc	86.01	79.92	(−6.09)	93.22	91.66	(−1.56)
		F1	71.88	53.83	(−18.05)	94.29	91.52	(−2.77)
	DT	Acc	90.17	84.22	(−5.95)	91.05	95.85	(+4.8)
		F1	79.18	63.10	(−16.08)	92.87	96.04	(+3.17)
EML	RF	Acc	86.32	75.96	(−10.36)	93.50	95.24	(+1.74)
		F1	72.51	49.56	(−22.95)	95.04	95.10	(+0.06)
	Adaboost	Acc	89.09	74.20	(−14.89)	70.81	68.96	(−1.91)
		F1	78.37	43.37	(−35)	18.84	18.98	(+0.14)
	XGBoost	Acc	91.86	81.02	(−10.84)	92.39	95.68	(+3.29)
		F1	83.38	57.67	(−25.71)	94.43	95.75	(+1.32)

The kitsune dataset, which consisted of data relating to seven IoT DDoS attacks, did not experience generalization errors, as the performance of the validation and test results were not significantly different for multi-classification. However, a significant performance difference between the validation and test results occurred in the case of binary classification, and a generalization error occurred because of overfitting. Therefore, we analyzed the training methods for the ML model to reduce the generalization error for the binary classification of the kitsune dataset.

5.1.2 Binary Classification: Results of Three Comparative Models

The method of presenting the binary classification results of the three comparison models confirms and compares the trend of the results from the perspective of validation and test results. Fig. 6 shows the trend of accuracy and f1-score for the six classifiers. The three comparison models' validation results (blue) and test results (orange) are shown for each classifier. Regarding validation results, SMOTE showed increased performance in both accuracy and f1-score. Regarding test results, SMOTE can confirm the improved performance in f1-score and GAN-CTGAN in accuracy. Table 3 shows the exact values of accuracy and f1-score as shown in Fig. 6. The following compares the results for each of the three comparison models regarding verification and test results for the values in Table 3.

Performance comparison for validation result. The accuracy for the Baseline validation result was between 85.47% and 79.86%, and the f1-score was between 71.88% and 92.38%. The accuracy for the SMOTE validation result was between 95.02% and 97.62%, excluding SVM (62.42%), and the f1-score was between 75.36% and 97.78%. The accuracy for the GAN-CTGAN validation result was between 80.31% and 85.65%, and the f1-score was between 71.85% and 83.70%.

Compared with the Baseline, the accuracy and f1-score tended to be higher for SMOTE (excluding SVM). The lowest accuracy was 4.92% (DT), and the highest was 9.07% (KNN). The lowest f1-score was 14.4% (XGBoost), and the highest was 23.48% (KNN). Additionally, compared with the Baseline, GAN-CTGAN tended to have lower accuracy. The lowest accuracy was 1.5% (KNN), and the highest was 8.47% (AdaBoost). Compared with the Baseline, the f1-score was reduced for SVM, AdaBoost, and XGBoost and increased for KNN, DT, and RF. From the perspective of the validation result, the oversampling technique using SMOTE exhibited higher ML training model performance than the data augmentation using GAN-CTGAN.

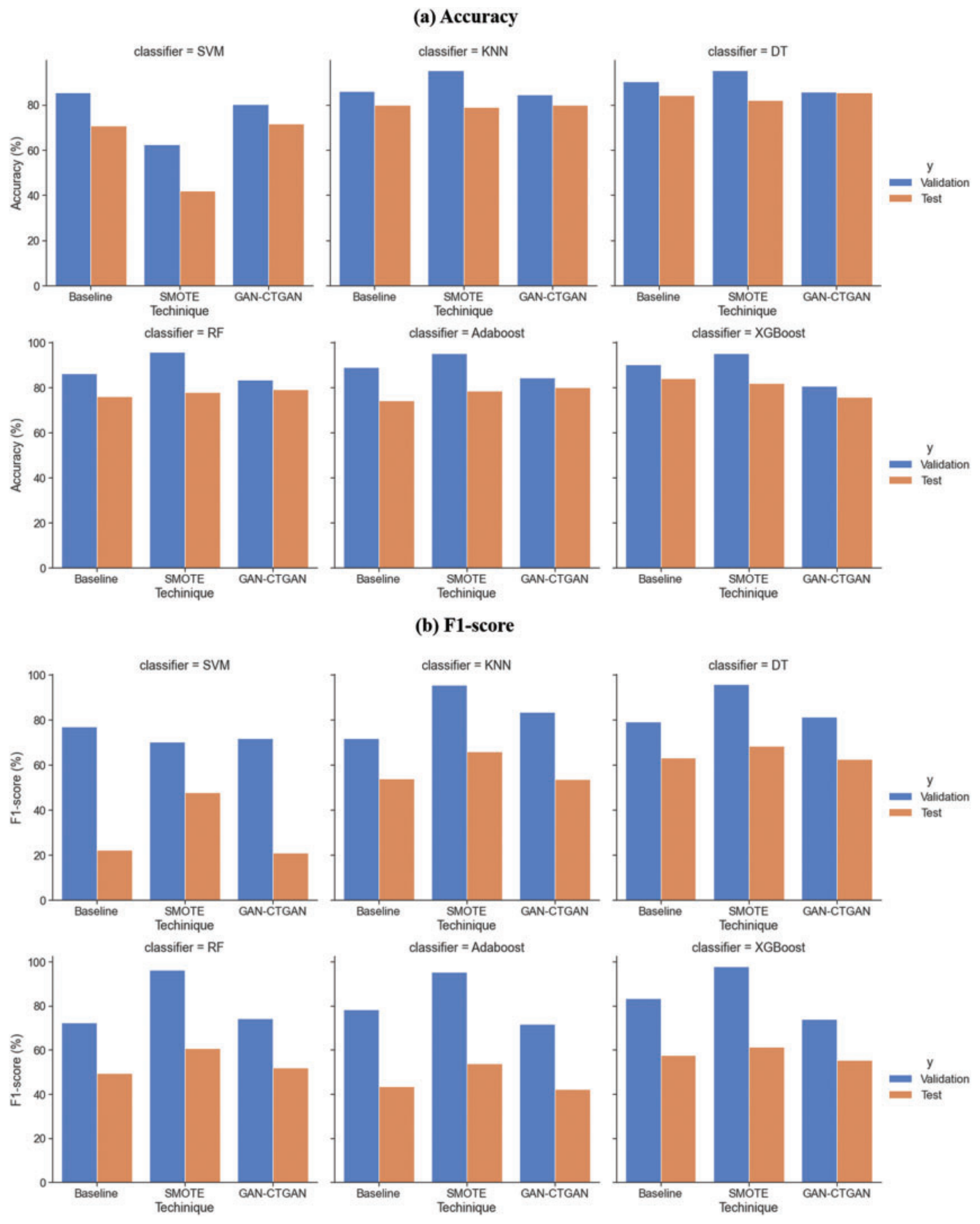

Figure 6: Graph showing the binary—acc and f1 trend for validation and test results of three comparative models. (a) Accuracy. (b) F1-score

Table 3: Binary—acc and f1 for the validation and test results of three comparative models

			Baseline		SMOTE		GAN-CTGAN	
Algorithm			Validation	Test	Validation	Test	Validation	Test
ML	SVM	Acc	85.47	70.65	62.42	41.89	80.31	71.69
		F1	77.11	22.39	70.37	47.64	71.67	20.95
	KNN	Acc	86.01	79.92	95.08	78.97	84.51	79.92
		F1	71.88	53.83	95.36	65.86	83.40	53.62
	DT	Acc	90.17	84.22	95.09	81.99	85.65	85.39
		F1	79.18	63.10	95.72	68.46	81.24	62.61
EML	RF	Acc	86.32	75.96	95.69	77.79	83.33	79.12
		F1	72.51	49.56	96.26	60.90	74.29	51.88
	Adaboost	Acc	89.09	74.20	95.02	78.62	80.62	75.75
		F1	78.37	43.37	95.40	53.77	71.85	42.37
	XGBoost	Acc	91.86	81.01	97.62	81.20	82.83	82.18
		F1	83.38	57.67	97.78	61.31	73.84	55.49

Performance comparison for the test result. The Baseline test result had accuracies between 70.65% and 84.22% and f1-scores between 43.37% and 63.10%, excluding SVM (22.39%). The SMOTE test result accuracy and f1-score were 77.79–81.99% and 53.77–68.46%, respectively, excluding SVM (47.64%). Lastly, the test result accuracy of GAN-CTGAN was between 71.69% and 85.39%, and the f1-score was between 42.37% and 62.61%, excluding SVM (20.95%).

In the case of SMOTE, compared with the Baseline, the accuracy was reduced for SVM, KNN, and DT and increased for RF, AdaBoost, and XGBoost. The f1-score tended to increase; the lowest value was 3.64% (XGBoost), and the highest value was 25.25% (SVM). For GAN-CTGAN, compared with the Baseline, the accuracy increased, and the F1-score tended to decrease, excluding RF. The lowest accuracy was 0% (KNN), and the highest was 3.16% (RF). The lowest f1-score was 0.21% (KNN), and the highest was 2.18% (XGBoost), excluding RF (2.35%). Therefore, from the perspective of the test result, the f1-score indicated improved performance for oversampling using SMOTE compared with data augmentation using GAN-CTGAN. However, relative to oversampling with the three classifier accuracies reduced, data augmentation generally exhibited superior performance for the six classifiers.

5.2 Perspective of Generalization Error

In the case of the binary classification of the Baseline, the performance difference between the validation and test results was significant. Therefore, generalization errors tended to occur because of overfitting. Oversampling based on SMOTE and data augmentation based on GAN-CTGAN were used to study the method for ML model training method. We evaluated the performance of the trained model from the perspective of the generalization error in comparison with the Baseline for SMOTE and GAN-CTGAN. Two methods were used to evaluate the generalization error for the trained model: (1) the learning curve and (2) the difference between the validation and test results.

5.2.1 *Learning Curve*

This section examines the learning curve for the three comparative models. Fig. 7 shows the learning curves for the three comparative models for SVM and RF. The scikit-learn learning curves were used, and the cross-validation value was 10. The graph shows the accuracy for the training (red) and validation (green) sets for the size of the training data sample. The estimated variance was expressed by deriving the standard deviation of the average accuracy for each accuracy and utilizing the fill_between function. In the Baseline, both SVM and RF showed significant differences in accuracy and estimated variance between training and validation. Therefore, the Baseline is likely to be overfitted and exhibit generalization errors.

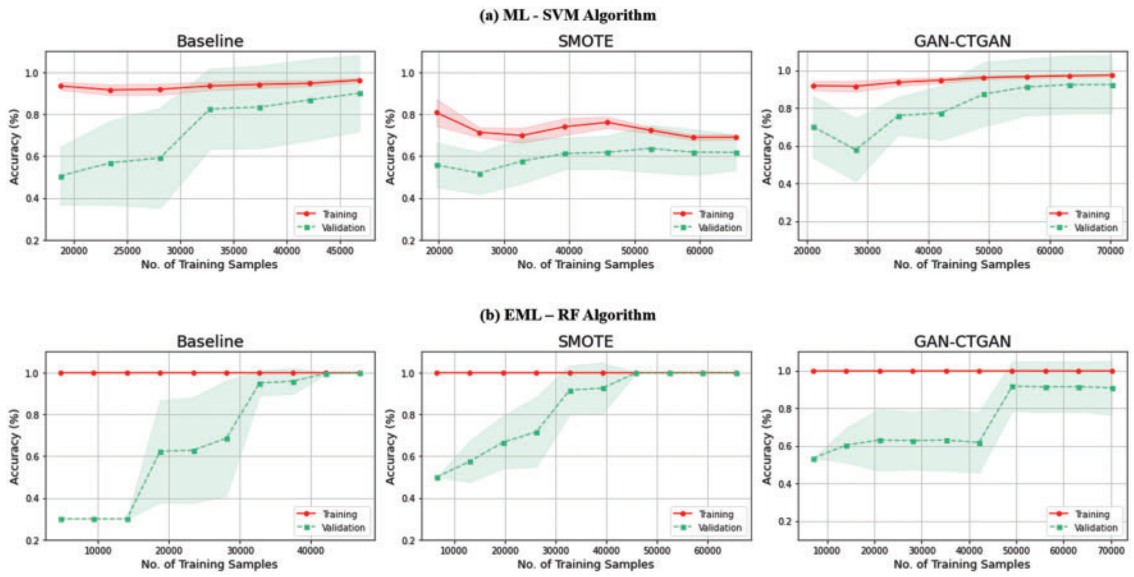

Figure 7: Learning curve graphs for SVM and RF of three comparative models. (a) ML-SVM algorithm. (b) EML-RF algorithm

For SVM, the SMOTE decreased training and validation accuracy and estimated variance compared to the Baseline. The GAN-CTGAN increased training and validation accuracy and decreased estimation variance compared to the Baseline. Compared with that Baseline, the SMOTE and GAN-CTGAN training accuracies were the same for RF, but the validation accuracy was higher, and the estimated variance was lower. Therefore, the estimated variance was reduced for both SMOTE and GAN-CTGAN, with the possibility of the generalization error having decreased by reducing the overfitting.

5.2.2 *Performance Differences Between Validation and Test Results*

The difference between the validation and test results is checked to assess the generalization error regarding the three comparison models. Fig. 8 consists of two graphs of accuracy and f1-score, which show the difference trend between the validation and test results for each of the six classifiers. We can also see in Fig. 8 that the model with the smallest value difference among the three comparison models mitigates the generalization error problem. Table 4 shows the exact value of the difference between the accuracy and f1-score value expressed in Fig. 8. The following is a comparison of the results for each of the three comparison models for the values in Table 4. SMOTE exhibited a performance difference: the accuracy compared with the Baseline increased to a minimum of 1.51% (AdaBoost) and a maximum of 10.02% (KNN). The performance difference was also evident from the f1-score, which increased to a minimum of 6.63% (AdaBoost) and a maximum of 12.44% (RF), excluding SVM (31.99% reduction). Additionally, a performance difference arose where the f1-score decreased to a minimum of 0.54% (RF) and a maximum of 7.26% (XGBoost), excluding KNN (11.73% increase) and DT (2.55% increase). Therefore, from the validation and test results perspective, the generalization error tended to increase for SMOTE and decrease for GAN-CTGAN.

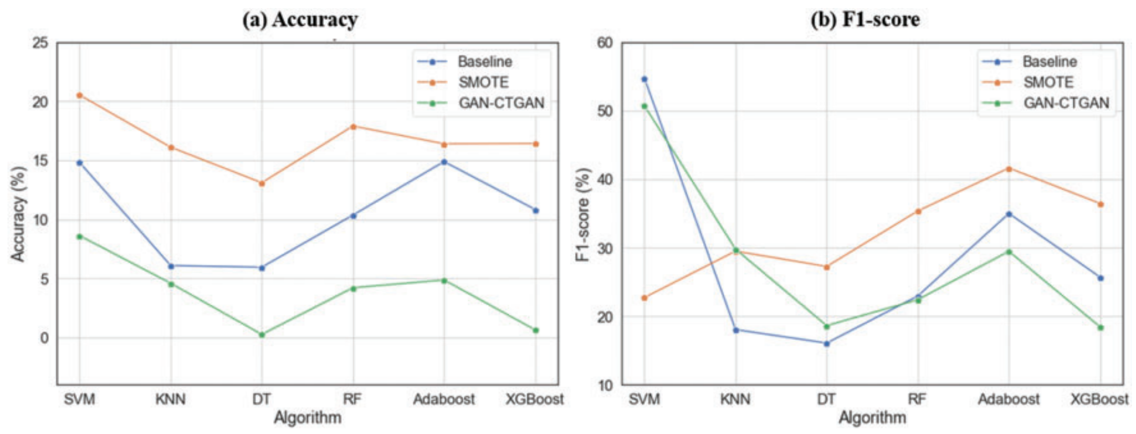

Figure 8: Graphs showing the trend of the difference between acc and f1 of the validation and test results for the three comparison models. (a) Accuracy. (b) F1-score

Table 4: Difference in acc and f1 between validation and test results of the three comparative models

		Baseline		SMOTE		GAN-CTGAN	
Algorithm		Acc diff.	F1 diff.	Acc diff.	F1 diff.	Acc diff.	F1 diff.
ML	SVM	14.82	54.72	20.53	22.73	8.62	50.72
	KNN	6.09	18.05	16.11	29.5	4.59	29.78
	DT	5.95	16.08	13.1	27.26	0.26	18.63
EML	RF	10.36	22.95	17.9	35.36	4.21	22.41
	Adaboost	14.89	35	16.4	41.63	4.87	29.48
	XGBoost	10.84	25.71	16.42	36.47	0.65	18.45

5.3 *Computational Complexity*

This section presents an assessment of the computational complexity of the experimental environment of SMOTE and GAN-CTGAN, the techniques that are used to solve the data imbalance of network traffic. The computational complexity was determined by measuring each technique's required time and RAM usage. First, the time library of Python was used to measure the required time. Next, the required time was measured by checking the current time at the start and end of the code of SMOTE and GAN-CTGAN. The following uses the psutil library to measure RAM usage. The method measures the RAM usage when the code starts and ends running and excludes the RAM usage at the start from the RAM usage at the end.

Fig. 9 and Table 5 show the time required and RAM usage when processing SMOTE (Oversampling) and GAN-CTGAN (Data Augmentation). Fig. 9 visualizes the trend of time and RAM usage when performing SMOTE and GAN-CTGAN. In addition, Table 5 shows the exact values for the trends depicted in Fig. 9. The following is a comparison of the results of SMOTE and GAN-CTGAN for Table 5. Compared to GAN-CTGAN, the time required by SMOTE was 1158 times less and used 12 times less RAM. SMOTE uses less time and less memory than GAN-CTGAN. However, previous experimental results show that SMOTE generates simple data using less time and space resources. In addition, compared to SMOTE, GAN-CTGAN is more time-consuming and uses a large amount of memory. However, previous experimental results confirmed that sophisticated data were created using considerable time and space resources.

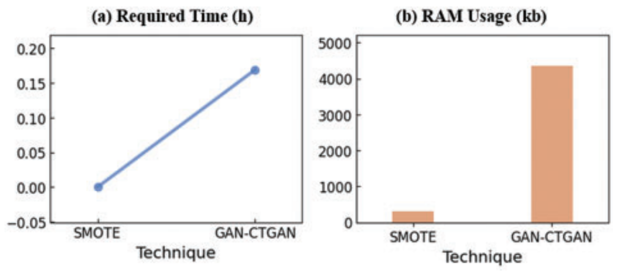

Figure 9: Graph of computational complexity. (a) Required time (h). (b) RAM usage (kb)

Table 5: Measurement of computational complexity

Technique	Required time (h)	RAM usage (kb)
SMOTE	0.00014586	321.175
GAN-CTGAN	0.16894671	3913.957

5.4 Comparison of Previous Research

This section compares our results to the related work reviewed in Section 2, and the differences are identified. In this study, we conducted a comparative experiment using SMOTE and GAN-CTGAN to address the problem of generalization error due to data imbalance. For each method, a comparison with related studies is performed to identify our results' meaning, differences, or limitations. The approach involves comparing the SMOTE experiments with SMOTE-related studies, after which the GAN-CTGAN experiments are compared with GAN-related studies.

5.4.1 SMOTE-Related Research

The SMOTE studies discussed in Section 2 aim to address the data imbalance problem. In addition, SMOTE is used not only in the field of network security related to this study but also in various research fields. Therefore, three studies introduced in Section 2 were selected and compared with the SMOTE experiment. The parts to be compared can be divided into four categories in Table 6. (1) Dataset, (2) ML/DL models, (3) SMOTE description (research field/purpose/other tech.), and (4) Results. Of these categories, the main category is (3) SMOTE description. Because the research fields using SMOTE and the techniques used together differ for each corresponding category, this was analyzed and compared with a focus on these techniques. The purpose of this comparison is not to compare the performance, as the datasets are different, but to analyze the differences between the previous study and this study and the information related to SMOTE to identify the meaning, differences, or limitations of the experimental results.

Table 6: Comparison of SMOTE-related research

Authors (year)	Dataset	ML/DL models	SMOTE description (research field/ purpose/ other tech.)	Results
Zhang et al. (2020) [12]	o UNSW-NB15 o CICIDS2017	o ML -RF o DL -MLP, CNN	o (research field) Network security - (purpose) Data imbalance - (other tech.) GMM	Acc (96.5%~99.7%)
Haq et al. (2022) [13]	o NAMP	o ML -SVM, KNN, XGBoost, RF o DL -DNN,	o (research field) Air pollution - (purpose) Data imbalance - (other tech.) Model optimization	Acc (99.9%)
Dablain et al. (2022) [14]	o MNIST o Fashion-MNIST o CIFAR-10 o SVHNs o CelebA	o DL -CNN	o (research field) Image - (purpose) Data imbalance - (other tech.) Encoder/Decoder	Acc (96.4%~98.1%)
In this research	o Kitsune	o ML -SVM, KNN, DT o EML -RF, Adaboost, XGBoost	o (research field) Network security - (purpose) Data imbalance - (other tech.) None	Acc (41.8%~81.99%) F1 (47.64%~68.46%)

As a result of the comparison, recent SMOTE studies use both ML and DL, but it is confirmed that SMOTE performs well in DL. In addition, other techniques (GMM, Encoder, etc.) are used together rather than using SMOTE alone. As such, this SMOTE experiment utilized SMOTE only on ML, not DL, and did not combine it with other techniques. Compared to GAN-CTGAN, it was confirmed that the SMOTE experimental results were not mitigated regarding the generalization error. Therefore, the comparison with previous studies indicated that it is necessary to confirm the experimental results by using DL and other techniques (GMM, Encoder, etc.) rather than the SMOTE application method of this study.

5.4.2 GAN-Related Research

The GAN research discussed in Section 2 is from the field of network security, which is related to this research. This research used GANs to solve various problems, not just the data imbalance issue. Among the studies introduced in Section 2, three were selected for comparison with ours. The purpose of this comparison is not to compare the performance of the datasets but to analyze the differences between the previous study and this study and the GAN-related information to identify the experimental results' meaning, differences, or limitations. The parts to be compared can be divided into the four categories in Table 7: (1) Dataset, (2) ML/DL models, (3) GAN description (type/purpose/method), and (4) Results. In particular, compared to previous studies, the main category is (3) GAN description (type/purpose/method). The type, purpose, and method of GAN used for each study in this category differ. Focusing on these features, we analyzed and compared previous research.

Table 7: Comparison of GAN-related research

Authors (year)	Dataset	ML/DL models	GAN description (type/purpose/method)	Results
Ahmed et al. (2020) [17]	o Own dataset	o ML -RF, DT, LR, NN, SVM	o (type) GAN - (purpose) Bypassing URL classification algorithms for ML models - (method) Creating adversarial attacks	FPR (0.12% or less)
Park et al. (2022) [22]	o NSL-KDD o UNSW-NB15 o IoT o Real data	o DL -DNN, CNN, LSTM	o (type) BEGAN - (purpose) Resolving data imbalance - (method) Creating synthetic data for minor attack	F1 (93.8%)
Abdelaty et al. (2021) [18]	o CICIDS2017 o Adversarial SYN flood attacks	o ML -LUCID	o (type) GADoT - (purpose) Adversarial training for robust model - (method) Training set augmented adversarial DDoS attacks	F1 (98% or more) FNR (1.8% or less)
In this research	o Kitsune	o ML -SVM, KNN, DT o EML -RF, Adaboost, XGBoost	o (type) CTGAN - (purpose 1) Resolving insufficient training data of 5GC Traffic - (method 1) Creating synthetic data for 5GC traffic in tabular form - (purpose 2) Resolving data imbalance - (method 2) Training set augmented by IoT DDoS attacks	Acc (71.6% or more) F1 (62.6% or less)

The comparison revealed that recent GAN research mainly focused on the field of network security, and in particular, GAN is used in various ways as well as data augmentation for data imbalance in this study. Our work was impeded by the availability of insufficient 5GC traffic in tabular form to train ML models. Therefore, CTGAN, which has been proven to generate learning data from network data in tabular form effectively, was used in the experiment [4,30,31]. In addition, owing to network traffic, the IoT DDoS attack was augmented using CTGAN and then included in the training set to train the ML model. Consequently, two perspectives, the evaluation index values and generalization errors were examined as the results of the experiment. The experimental results confirmed that CTGAN did not change significantly in the evaluation index values compared to Baseline, but the generalization error tended to be alleviated by reducing overfitting. Our investigation confirmed that various methods and applications based on GAN had been developed, as well as data augmentation methods in the field of network security.

6 Conclusion

This study conducted a comparative experiment for an ML model training method to mitigate the generalization error problem under insufficient and imbalanced data. This experiment shows that the method of augmenting attack data using GAN-CTGAN is effective as an ML model training method for classifying IoT DDoS attacks in 5G environments. First, to solve the problem of insufficient data, we built our own 5G testbed and constructed a dataset of a 5G environment. Next, we processed three training datasets (Baseline, SMOTE, and GAN-CTGAN) using SMOTE and GAN-CTGAN to solve the imbalanced data problem. Accordingly, it was confirmed that the generalization error problem occurred in the ML model trained with the Baseline. The ML model trained with GAN-CTGAN showed the tendency to alleviate the generalization error problem the most.

The meaning of the experimental results was analyzed from various perspectives by comparing the computational complexity and previous studies regarding SMOTE and GAN-CTGAN as the techniques used. First, we measured the computational complexity of generating data with SMOTE and GAN-CTGAN. Although GAN-CTGAN has a higher computational complexity than SMOTE, it was analyzed that the generalization error was mitigated, as it generated sophisticated data. Next, we compared SMOTE and GAN to previous studies to see the latest trends in each technique. As a recent trend, the SMOTE showed effective performance when using DL and other techniques together rather than ML alone, and it was confirmed that the GAN has various methods and applications in the field of network security.

In this experiment, the performance of the trained ML model was analyzed using one kitsune dataset. As a future research direction, we plan to evaluate the performance of the trained ML model in terms of higher feasibility by using multiple datasets rather than one dataset. In addition, there are not only CTGAN but also TableGAN and CopularGAN as GAN types for generating tabular data. Since the CTGAN is effective in the 5GC environment, we plan to carry out research to conduct comparative experiments on other types of GANs. In addition, this study created attack data using training and test sets due to a lack of data when augmenting attack data with CTGAN. In future research, this should be improved so that only the training set can be utilized to generate data.

Funding Statement: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00796, Research on Foundational Technologies for 6G Autonomous Security-by-Design to Guarantee Constant Quality of Security).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. M. French and J. P. Shim, "The digital revolution: Internet of things, 5G, and beyond," *Communications of the Association for Information Systems*, vol. 38, no. 1, pp. 840–850, 2016. https://doi.org/10.17705/1CAIS.03840

[2] 3GPP—Technical Specification Group Services and System Aspects, "Architecture enhancement for 5G System (5GS) to support network data analytics services (release 18)," 2022. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.288/23288-i00.zip

[3] L. Xu, M. Skoularidou, A. Cuesta-Infante and K. Veeramachaneni, "Modeling tabular data using conditional GAN," *Advances in Neural Information Processing Systems*, vol. 32, pp. 1–15, 2019.

[4] S. Bourou, A. El Saer, T. H. Velivassaki, A. Voulkidis and T. Zahariadis, "A review of tabular data synthesis using GANs on an IDS dataset," *Information*, vol. 12, no. 9, pp. 375–388, 2021. https://doi.org/10.3390/info12090375

[5] C. Nadeau and Y. Bengio, "Inference for the generalization error," in *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, London, England, vol. 12, pp. 307–313, 1999.

[6] Y. K. Saheed, M. O. Arowolo and A. U. Tosho, “An efficient hybridization of k-means and genetic algorithm based on support vector machine for cyber intrusion detection system,” *International Journal on Electrical Engineering and Informatics*, vol. 14, no. 2, pp. 426–442, 2022. https://doi.org/10.15676/ijeei.2022.14.2.11

[7] M. A. Haq and M. A. R. Khan, “Dnnbot: Deep neural network-based botnet detection and classification,” *Computers, Materials & Continua*, vol. 71, no. 1, pp. 1729–1750, 2022. https://doi.org/10.32604/cmc.2022.020938

[8] S. Rizvi, M. Scanlon, J. Macgibney and J. Sheppard, “Deep learning based network intrusion detection system for resource-constrained environments,” in *Proc. 13th Int. Conf. on Digital Forensics and Cybercrime (ICDF2C)*, Boston, MA, USA, pp. 1–7, 2023.

[9] X. Ma and W. Shi, “Aesmote: Adversarial reinforcement learning with SMOTE for anomaly detection,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 943–956, 2020. https://doi.org/10.1109/TNSE.2020.3004312

[10] Y. S. Won, D. Jap and S. Bhasin, “Push for more: On comparison of data augmentation and SMOTE with optimised deep learning architecture for side-channel,” in *Proc. Information Security Applications: 21st Int. Conf. (WISA 2020)*, Jeju Island, Korea, vol. 21, pp. 227–241, 2020.

[11] G. Karatas, O. Demir and O. K. Sahingoz, “Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-data dataset,” *IEEE Access*, vol. 8, pp. 32150–32162, 2020. https://doi.org/10.1109/ACCESS.2020.2973219

[12] H. Zhang, L. Huang, C. Q. Wu and Z. Li, “An effective convolutional neural network based on SMOTE and gaussian mixture model for intrusion detection in imbalanced dataset,” *Computer Networks*, vol. 177, no. 18, pp. 107315–107324, 2020. https://doi.org/10.1016/j.comnet.2020.107315

[13] M. A. Haq, “Smotednn: A novel model for air pollution forecasting and aqi classification,” *Computers, Materials & Continua*, vol. 71, no. 1, pp. 1403–1425, 2022. https://doi.org/10.32604/cmc.2022.021968

[14] D. Dablain, B. Krawczyk and V. Chawla, “DeepSMOTE: Fusing deep learning and SMOTE for imbalanced data,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022. https://doi.org/10.1109/TNNLS.2021.3136503

[15] J. H. Joloudari, A. Marefat, M. A. Nematollahi, S. S. Oyelere and S. Hussain, “Effective class-imbalance learning based on SMOTE and convolution neural networks,” arXiv preprint arXiv:2209.00653, pp. 1–43, 2022.

[16] W. Hu and Y. Tan, “Generating adversarial malware examples for black-box attacks based on GAN,” in *Proc. Data Mining and Big Data: 7th Int. Conf. (DMBD)*, Beijing, China, pp. 409–423, 2023.

[17] A. Ahmed and G. Karabatis, “Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks,” in *Proc. of the 6th Int. Workshop on Security and Privacy Analytics (IWSPA)*, LA, New Orleans, USA, pp. 53–60, 2020.

[18] M. Abdelaty, S. Scott-Hayward, R. Doriguzzi-Corin and D. Siracusa, “GADoT: GAN-based adversarial training for robust DDoS attack detection,” in *Proc. 2021 IEEE Conf. on Communications and Network Security (CNS)*, pp. 119–127, 2021.

[19] Y. Guo, G. Xiaong, Z. Li, J. Shi, M. Cui *et al.*, “TA-GAN: GAN based traffic augmentation for imbalanced network traffic classification,” in *Proc. 2021 Int. Joint Conf. on Neural Networks (IJCNN)*, Shenzhen, China, pp. 1–8, 2021.

[20] H. Nan, X. Zhu and J. Ma, “An efficient correlation-aware anomaly detection framework in cellular network,” *China Communications*, vol. 19, no. 8, pp. 168–180, 2022. https://doi.org/10.23919/JCC.2022.08.013

[21] H. J. Kim, J. Lee, C. Park and J. G. Park, “Network anomaly detection based on GAN with scaling properties,” in *Proc. of the 2021 Int. Conf. on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Korea, pp. 1244–1248, 2021.

[22] C. Park, J. Lee, Y. Kim, J. G. Park, H. Kim *et al.,* "An enhanced AI-based network intrusion detection system using generative adversarial networks," *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2330–2345, 2022. https://doi.org/10.1109/JIOT.2022.3211346

[23] ITU, "IMT vision—Framework and overall objectives of the future development of IMT for 2020 and beyond," 2015. [Online]. Available: https://www.itu.int/rec/R-REC-M.2083-0-201509-P/en

[24] 3GPP—Technical Specification Group Services and System Aspects, "System architecture for the 5G system (5GS); stage2 (release 18)," 2022. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/23501-i00.zip

[25] Cisco Systems. Inc, "Ultra cloud core 5G user plane function, release 2021.01—configuration and administration guide," 2021. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/wireless/ucc/upf/2021-01/b_ucc-5g-upf-config-and-admin-guide_2021-01/m_upf-overview.html

[26] R. Mohammed, J. Rawashdeh and M. Adbullah, "Machine learning with oversampling and undersampling techniques: Overview study and experimental results," in *Proc. 2020 11th Int. Conf. on Information and Communication Systems (ICICS)*, Irbid, Jordan, pp. 243–248, 2020.

[27] A. Mikolajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *Proc. 2018 Int. Interdisciplinary PhD Workshop (IIPhDW)*, Swinoujscie, Poland, pp. 117–122, 2018.

[28] The imbalanced-learn developers, [Online]. Available: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

[29] C. Shorten and T. M. Khoshogoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019. https://doi.org/10.1186/s40537-019-0197-0

[30] J. Moon, S. Jung, S. Park and A. E. Hwang, "Conditional tabular GAN-based two-stage data generation scheme for short-term load forecasting," *IEEE Access*, vol. 8, pp. 205327–205339, 2020. https://doi.org/10.1109/ACCESS.2020.3037063

[31] O. Habibi, M. Chemmakha and M. Lazaar, "Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT botnet attacks detection," *Engineering Applications of Artificial Intelligence*, vol. 118, no. 9, pp. 105669–105691, 2023. https://doi.org/10.1016/j.engappai.2022.105669

[32] Github, "CTGAN-conditional GAN for generating synthetic tabular data," [Online]. Available: https://github.com/sdv-dev/CTGAN

[33] L. Xu, M. Skoularidou, A. Cuesta-Infante and K. Veeramachaneni, "Modeling tabular data using conditional GAN," *Advances in Neural Information Processing Systems*, vol. 32, pp. 1–11, 2019.

[34] Kaggle, "Kitsune network attack dataset," [Online]. Available: https://www.kaggle.com/datasets/ymirsky/network-attack-dataset-kitsune

[35] Y. Mirsky, T. Doitshman, Y. Elovici and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," arXiv preprint arXiv:1802.09089, pp. 1–15, 2018.

[36] Github, "UERANSIM: Open source 5G UE and RAN (gNodeB) implementation," [Online]. Available: https://github.com/aligungr/UERANSIM

[37] Open5GS, "Open5GS: Quickstart," [Online]. Available: https://open5gs.org/open5gs/docs/guide/01-quickstart/