IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 December 2023, accepted 21 December 2023, date of publication 1 February 2024, date of current version 7 February 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3360879*

RESEARCH ARTICLE

# GPT and Interpolation-Based Data Augmentation for Multiclass Intrusion Detection in IIoT

**FRANCISCO S. MELÍCIAS[1], TIAGO F. R. RIBEIRO[1], CARLOS RABADÃO[1], LEONEL SANTOS[1], AND ROGÉRIO LUÍS DE C. COSTA[2]**
[1]CIIC, ESTG, Polytechnic of Leiria, 2411-901 Leiria, Portugal
[2]CIIC, Polytechnic of Leiria, 2411-901 Leiria, Portugal

Corresponding author: Rogério Luís de C. Costa (rogerio.l.costa@ipleiria.pt)

This work was supported in part by the Fundação para a Ciência e a Tecnologia (FCT), I.P., under Project UIDB/04524/2020; in part by the Scientific Employment Stimulus-Institutional Call under Grant CEECINST/00051/2018; and in part by the Agência Nacional de Inovação (ANI), S.A., under Project POCI-01-0247-FEDER-046083.

**ABSTRACT** The absence of essential security protocols in Industrial Internet of Things (IIoT) networks introduces cybersecurity vulnerabilities and turns them into potential targets for various attack types. Although machine learning has been used for intrusion detection in the IIoT, datasets with representative data of common attacks of IIoT network traffic are limited and often imbalanced. Data augmentation techniques address these problems by creating artificial data in classes with fewer samples. In this work, we evaluate the use of data augmentation when training intrusion detection models based on IIoT traffic data. We compare Generative Pre-trained Transformers (GPT) and variations on the Synthetic Minority Over-sampling TEchnique (SMOTE) and evaluate their capability to enhance intrusion detection performance. We examine the performance of five intrusion detection algorithms when trained with augmented datasets to models trained with the original non-augmented dataset. To ensure a fair comparison, we evaluated the algorithms' performance in the different scenarios using the same test dataset, which does not contain synthetic data. The results show the need for a systematic evaluation before employing data augmentation, as its impact on classification performance depends on the algorithm, data, and used technique. While deep neural networks benefit from data augmentation, the eXtreme Gradient Boosting (XGBoost), which achieved superior performance in intrusion detection between all evaluated classifiers (with F1-Score over 91%), didn't have any performance improvement when trained with augmented data. The evaluation of data generated by GPT-based methods shows such methods (especially GReaT) generate invalid data for both numerical and categorical features in a way that leads to performance degradation in multiclass classification.

**INDEX TERMS** IIoT, cybersecurity, data augmentation, machine learning.

## NOMENCLATURE

| | |
|---|---|
| *ADASYN* | Adaptive Synthetic Sampling Approach for Imbalanced Learning. |
| *BYOL* | Bootstrap Your Own Latent. |
| *CNN* | Convolutional Neural Network. |
| *DCS* | Distributed Control System. |
| *DDoS* | Distributed Denial-of-Service. |
| *DNN* | Deep Neural Network. |
| *DT* | Decision Tree. |
| *GAN* | Generative Adversarial Network. |
| *CTGAN* | Conditional Tabular Generative Adversarial Network. |
| *GPT* | Generative Pre-trained Transformer. |
| *ICMP* | Internet Control Message Protocol. |
| *ICS* | Industrial Control System. |
| *ICVAE-BSM* | Improved Conditional Variational Autoencoder - Borderline SMOTE. |
| *IDS* | Intrusion Detection System. |
| *IIoT* | Industrial Internet of Things. |
| *IoMT* | Internet of Medical Things. |
| *LR* | Logistic Regression. |

The associate editor coordinating the review of this manuscript and approving it for publication was Stefano Scanzio.

| | |
|---|---|
| *KNN* | k-Nearest Neighbors. |
| *ML* | Machine Learning. |
| *MITM* | Man-in-the-middle. |
| *NLP* | Natural Language Processing. |
| *PA* | Pseudo Anomaly. |
| *PLC* | Programmable Logic Controller. |
| *REalTabFormer* | Realistic Relational and Tabular Data using Transformers. |
| *ReLU* | Rectified Linear Unit. |
| *ROS* | Random Oversampling. |
| *RTU* | Remote Terminal Unit. |
| *SCADA* | Supervisory Control and Data Acquisition. |
| *SMOTE* | Synthetic Minority Over-sampling Technique. |
| *SQL* | Structured Query Language. |
| *SVM* | Support Vector Machine. |
| *UDP* | User Datagram Protocol. |
| *UMAP* | Uniform Manifold Approximation and Projection. |
| *XGBoost* | eXtreme Gradient Boosting. |
| *XSS* | Cross-Site Scripting. |
| *KNN* | k-Nearest Neighbors. |
| *MLP* | Multilayer Perceptron |
| *TP* | True Positives. |
| *FP* | False Positives. |
| *TN* | True Negatives. |
| *FN* | False Negatives. |
| *GReaT* | Generation of Realistic Tabular data. |

## I. INTRODUCTION

The Industrial Internet of Things (IIoT) is a network of multiple devices (such as sensors, instruments, computers, and machines) interconnected via the Internet to collect, monitor, and analyze data to streamline industrial production, aiding decision-making with real-time information, and reducing the need for human intervention [1]. The connectivity provided by IIoT networks enables the monitoring and optimization of industrial processes [2], better production quality control [3], as well as the automation of data collection and the reduction of data entry-related errors [4], more precise and real-time asset tracking and inventory monitoring [5], and facilitating the adoption of predictive maintenance methods aimed at minimizing downtime [6].

On the other hand, IIoT ecosystems may also introduce potential cybersecurity vulnerabilities mainly due to the lack of basic security protocols, such as exposed ports, legacy software, or weak authentication [7]. Indeed, IIoT systems may become a target of several types of cyberattacks with distinct consequences. For instance, integrity-compromising attacks pose a significant threat as they aim to compromise the system's reliability or data integrity. These attacks can manifest as sensor data tampering, alteration of control system commands, or modification of firmware [8]. Availability attacks aim to disrupt or limit system accessibility. Distributed Denial-of-Service (DDoS) attacks, wireless communication interference, and physical onslaughts on devices or infrastructure are examples of availability threats [9], [10]. Confidentiality attacks represent another class of threats primarily designed to compromise sensitive information. These attacks usually involve eavesdropping on wireless communications, stealing data, or gaining unauthorized access using breached credentials [11], [12]. Further, authentication attacks exploit weaknesses in the authentication procedures, such as cracking passwords, instigating Man-in-the-middle (MITM) attacks, or executing Sybil and spoofing attacks [13], take advantage of flaws in the authorization process and might result in privilege escalation, access control bypass, and exploitation of vulnerabilities in the authorization protocol [11], [14]. The wide range of potential vulnerabilities calls for adopting cybersecurity measures in IIoT networks.

In the last few years, some works proposed using Machine Learning (ML) for intrusion and anomaly detection and network vulnerability analysis on the Internet of Things [15], [16] and on IIoT networks [17], [18], [19]. However, the performance of ML models is highly dependent on the quality of training data, and utilizing limited and non-representative data leads to inadequate generalization performance [20]. Particularly in anomaly and intrusion detection within IIoT network traffic, training datasets are scarce and often lack coverage of many prevalent attacks. Additionally, there is usually an imbalance in the amount of data representing network traffic generated by non-malicious operations and the one generated by each type of malicious action [21], thus resulting in class imbalance and biased predictions. To overcome this issue, one common approach is to artificially increase the size of a dataset by creating new data for minority classes, in a process called Data Augmentation [22]. By increasing the amount of data of certain classes available to train ML models, one aims to improve the model's performance and mitigate overfitting during the training of an ML model [20].

Data augmentation techniques have been extensively studied in the computer vision domain, and are especially valuable for image classification and segmentation tasks. Typically, in computer vision, data augmentation involves applying various transformations to existing images or generating new ones by flipping, rotating, cropping, scaling, adding noise, and adjusting brightness and contrast [23], [24]. Likewise, in NLP (Natural Language Processing) applications, data augmentation revolves around transforming existing text or generating new text instances to enrich the training set. Techniques such as synonym replacement, random insertion, random deletion, and back-translation are commonly employed and greatly benefit tasks such as text classification, sentiment analysis, and machine translation [25].

But most data on anomaly or intrusion detection in IIoT networks is stored in tabular format [26]. In such a context, there are distinct methods for data augmentation.

Oversampling techniques aim to generate the least represented data to balance class distributions, effectively increasing the size of the minority classes. Random Oversampling (ROS) is one of these methods that is reasonably simple. To balance the class distribution, it randomly duplicates instances from the minority class. However, creating exact copies may lead to overfitting of classification methods. Synthetic Minority Over-sampling Technique (SMOTE) [27] generates synthetic samples by taking the feature space's interpolation for minority class samples, thus creating a broader and more generalized decision surface, which reduces the risk of overfitting. More recently, some works have proposed the use of generative deep learning models based on Generative Adversarial Networks (GANs) [28], Autoencoders [29], and Transformer networks [30] to learn data representations and subsequently generate data conditioned on certain less represented classes. A recent work [31] proposed the Realistic Relational and Tabular Data using Transformers (REalTabFormer), which employs an autoregressive model to generate non-relational tabular data. In addition to REalTabFormer, the Generation of Realistic Tabular data (GReaT) [32] also leverages a GPT-based model to generate realistic tabular data.

In this work, we evaluate the use of an autoregressive model (i.e., REalTabFormer and GReaT) and compare it to techniques based on interpolation for minority class samples (i.e., SMOTE and SMOTE-NC) for generating synthetic data on network attacks in the IIoT. We use the evaluated data augmentation methods to generate balanced datasets on IIoT network traffic. Then, we train several classification models - i.e., Decision Trees (DTs), Random Forest (RF), XGBoost [33], a multilayer perceptron (MLP) and Tabnet [34] - on such datasets with synthetic data and evaluate their performance while identifying malicious traffic on a test set containing only real-world data. We compare the data generated by each augmentation method and their per-class impact on each classification model. Our analysis supports identifying inadequate values generated by GPT-based methods and when to use tabular data augmentation to generate train data for IIoT traffic classification.

The main contributions of this work include an evaluation of using novel GPT-based data augmentation methods in the IIoT network data context, a comparison of the impact of traditional methods with ML-based augmentation methods, and the identification of which of the commonly used network traffic classification models are more impacted by augmentation strategies.

In the following section, we present some background and related work. Section III describes our evaluation methodology. In Section IV, we present and discuss experimental results. Section V) concludes the paper and outlines future research directions.

## II. BACKGROUND AND RELATED WORK

Industrial Control Systems (ICSs) have evolved significantly over the past two decades. Traditionally, operational technology devices such as Programmable Logic Controllers (PLCs), Remote Terminal Units (RTUs), Distributed Control Systems (DCSs), and Supervisory Control and Data Acquisition (SCADA) systems were protected by air-gapping, which physically separated them from external networks [35]. This approach effectively shielded operational technology devices from internet-based threats. However, the landscape has changed dramatically with the convergence of information and operational technology domains, which characterizes the Fourth Industry revolution [36].

While IIoT offers numerous benefits, such as reduced asset downtime and energy consumption, and provides end-to-end visibility of the production processes, it also exposes ICS networks to potential cyberattacks [37]. To better understand and mitigate these risks, analyzing IIoT network traffic, such as traffic flows or packets, has gained importance. A network flow can be defined as a sequence of packets sent from a source computer to a destination, passing an observation point in the network during a specific time interval [38]. This destination can be another host, a multicast group, or a broadcast domain. Packets within a flow share a set of characteristics, such as source and destination IP addresses, ports or MAC addresses, protocol, or type of service [39]. The flows may be observed at various points in the network and stored in a flow record. By analyzing flow records, organizations can gain insights into the communication patterns within ICS networks, detect anomalies, and identify potential security breaches.

### A. MACHINE LEARNING-BASED ATTACK DETECTION

ML-based techniques have demonstrated their efficiency in assessing and mitigating vulnerabilities and intrusion detection tasks in the IIoT context [11]. The classifiers for malicious traffic detection include tree-based algorithms (Decision Tree (DT), Random Forests, XGBoost [33]) and deep models (e.g., TabNet [34]).

#### 1) DECISION TREE

The DT algorithm employs a hierarchical model, structured like a tree, to predict outcomes, facilitating informed predictions. It is a non-parametric supervised learning algorithm that can execute both classification and regression tasks. The algorithm recursively splits the data into subsets based on the most significant feature at each node. The resulting DT structure resembles a series of if-else conditions, where the path from the root node to a leaf node represents the sequence of decisions required to arrive at a prediction. This interpretable nature of DTs allows for easy comprehension and insight into the decision-making process.

#### 2) RANDOM FOREST

The Random Forest algorithm is widely used for classification problems, being an ensemble learning method that constructs multiple DTs at training time. It takes their majority vote for classification tasks or averages their

predictions for regression tasks. Random Forest operates on the concept of bagging, where a group of models is trained on different subsets of the dataset. The final output is obtained by combining the predictions of these individual models.

#### 3) XGBOOST

XGBoost is another ensemble learning method that combines multiple DTs to make predictions. This algorithm uses a gradient-boosting framework to improve the accuracy of predictions. XGBoost iteratively adds DTs to the model, with each tree aiming to rectify the errors of its predecessor. The algorithm uses a gradient descent optimization method to minimize the loss function, which measures the discrepancy between the predicted and actual values. It also includes regularization terms in the loss function to prevent overfitting and improve generalization performance. XGBoost is particularly effective for handling large datasets with numerous features and has found applications in various domains, including intrusion detection in IIoT systems [40].

#### 4) MULTILAYER PERCEPTRON

An Multilayer Perceptron (MLP) is a type of feedforward artificial neural network that consists of an input layer, one or more hidden layers, and an output layer. Each layer comprises multiple neurons, and the neurons in adjacent layers are fully interconnected. The MLP employs various activation functions, such as Rectified Linear Unit (ReLU) or sigmoid, to introduce non-linearity into the model. The MLP operates as a feedforward algorithm, where inputs are combined with initial weights in a weighted sum and subjected to the activation function. The linear combinations are propagated to the next layer. This method consists of calculating the gradient of the error function, which quantifies the difference between the network's output and the expected output. The calculated gradients are then used to adjust the weights of the network in a way that minimizes the error function. This process is repeated for several iterations until the network's performance on the training data is satisfactory.

#### 5) TABNET

TabNet is a specialized deep-learning architecture designed for tabular data that utilizes a sequential attention mechanism for feature selection. The TabNet encoder comprises a feature transformer, an attentive transformer, and feature masking. The processed representation is then passed through a split block, which the subsequent attentive transformer employs to generate the overall output. The architecture operates through multiple sequential steps, with data being forwarded from one step to the next. At each step, the feature selection mask provides interpretable information about the model's functionality. These masks can be aggregated to determine the global importance of features. The TabNet decoder consists of a feature transformer block at each step. TabNet accepts raw tabular data without preprocessing and is trained using gradient descent-based optimization. The sequential attention mechanism enables interpretability and improved learning by selectively focusing on the most relevant features at each decision step.

**TABLE 1.** IIoT traffic datasets class distribution.

| Dataset | Year | Features | Attacks (%) | Classes |
|---|---|---|---|---|
| GP SCADA [41] | 2015 | 20 | 21.90 | 36 |
| SCADA System [42] | 2018 | 6 | 6.07 | 6 |
| Electra Modbus [43] | 2019 | 10 | 5.20 | 7 |
| Electra S7comm [43] | 2019 | 10 | 1.42 | 7 |
| TON IoT [44] | 2020 | 22 | 38.92 | 8 |
| WUSTL-IIOT [45] | 2021 | 41 | 7.28 | 5 |
| X-IIoTID [46] | 2021 | 64 | 48.66 | 19 |
| Edge-IIoTset [12] | 2022 | 61 | 46.43 | 15 |

**TABLE 2.** Edge-IIoTset dataset traffic type distribution.

| Traffic Type | Attack Type | Records | Class (%) |
|---|---|---|---|
| Normal | - | 1,615,643 | 72.80 |
| Attack | DDoS (UDP) | 121,568 | 5.48 |
| | DDoS (ICMP) | 116,436 | 5.25 |
| | SQL injection | 51,203 | 2.31 |
| | Password | 50,153 | 2.26 |
| | Vulnerability scanner | 50,110 | 2.26 |
| | DDoS (TCP) | 50,062 | 2.26 |
| | DDoS (HTTP) | 49,911 | 2.25 |
| | Uploading | 37,634 | 1.70 |
| | Backdoor attack | 24,862 | 1.12 |
| | Port scanning | 22,564 | 1.02 |
| | XSS | 15,915 | 0.72 |
| | Ransomware | 10,925 | 0.49 |
| | Man-in-the-middle | 1,214 | 0.06 |
| | Fingerprinting | 1,001 | 0.05 |
| Attacks (total) | - | 603,558 | 27.20 |
| **Total** | | 2,219,201 | 100.00 |

The aforementioned ML models offer distinct advantages and limitations for IIoT attack detection. DTs provide interpretability but may struggle with complex relationships and overfitting. Random Forest enhances accuracy through ensemble learning but can be computationally expensive and less interpretable. XGBoost excels in handling large datasets but requires careful tuning and computational resources. MLPs are capable of capturing complex patterns but demand significant training data and regularization. TabNet, as a specialized architecture for tabular data, offers interpretability and focuses on relevant features, but may require substantial computational resources.

In this work, we evaluate such classifiers, accessing the impact of data augmentation on their performance while executing multiclass classification on IIoT network data.

### B. DATA AUGMENTATION

Imbalance data is a significant challenge in training ML in many scenarios, including in network traffic classification in IIoT. In this context, the dataset composition often leans heavily towards benign network traffic, creating a scenario where the model may become biased, thus leading to suboptimal detection of cyber threats. Table 1 illustrates class

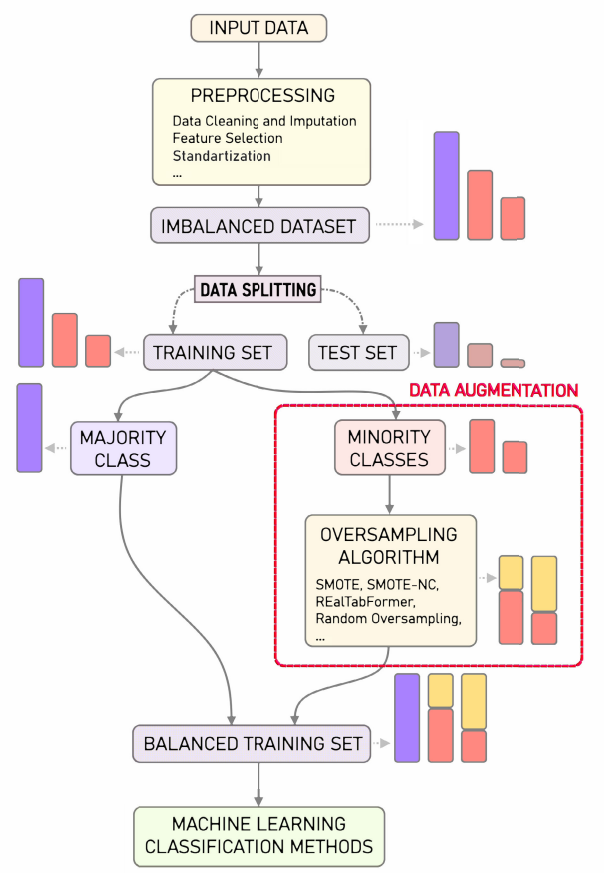



**FIGURE 1.** General data augmentation workflow for imbalanced datasets.

data imbalance in IIoT network traffic datasets, where the proportion of benign network traffic records significantly surpasses the one generated by cyberattacks. Furthermore, class imbalance may also lead to a small number of instances for some classes representing traffic resulting from malicious actions, as shown in Table 2 for the Edge-IIoTset dataset [12].

Data augmentation techniques generate synthetic data, potentially leading to more balanced datasets. They also serve as a regularization technique, mitigating the risk of overfitting and enhancing the model's generalization capabilities [47]. Bayer et al. [47] argue that enriching the dataset enables models to extrapolate more effectively from training data, thereby improving their predictive capabilities. Additionally, as acquiring labeled data can be resource-intensive, data augmentation may reduce the need for extensive manual labeling and the associated costs.

Furthermore, in scenarios with sensitive data, data augmentation proves beneficial by reducing dependence on real-world data. Models can be trained using artificially generated data, safeguarding sensitive information embedded in the original dataset. The role of data augmentation extends to addressing vulnerabilities related to adversarial examples. In adversarial training, it decreases the susceptibility of models to often imperceptible alterations in input data

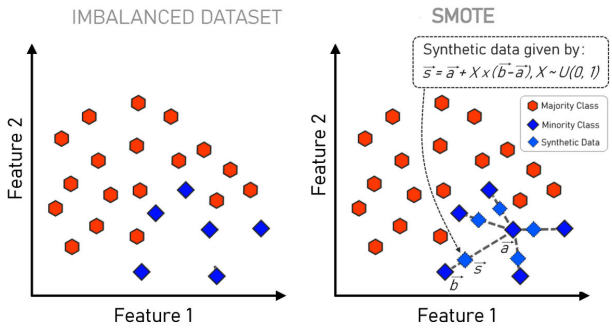



**FIGURE 2.** Simplified SMOTE illustration.

that could lead to incorrect predictions [48] On the other hand, data augmentation cannot cover all transformation possibilities nor eliminate biases in the original data [47]. Moreover, there is no consensus on the data augmentation technique to use to address the problem of unbalanced datasets, and the effectiveness of each method depends on aspects like the problem's characteristics, the dataset features, and the model to be employed.

There are several algorithms for oversampling tabular data. Usually, to balance a dataset, a data augmentation technique is applied to generate synthetic data for identified minority classes on training set after data pre-processing, as represented in Figure 1.

ROS is one of the simplest methods to expand the dataset of minority classes. ROS randomly selects and duplicates observations from the minority classes while allowing replacement. One downside to this approach is that it can alter the original data distribution.

SMOTE is a widely used oversampling approach that generates synthetic observations instead of duplicating existing ones. Initially, the algorithm selects a data point from the minority class and identifies the point's nearest neighbour that belongs to the same class. Then, it calculates the distance between these two points by computing the difference between the feature vectors of the points. Such a distance is multiplied by a random number ranging from 0 to 1 and added to the selected data point. As a result, the new sample is positioned along the line segment connecting the starting point and its neighbour, as represented in Figure 2. The algorithm repeats this process until it generates the desired number of synthetic samples. Using methods like ROS to add exact copies of minority cases to a dataset can lead to overfitting issues. On the other hand, SMOTE can mitigate this problem by promoting a more generalized decision region for the minority class.

SMOTE-NC is a variant of SMOTE that deals with datasets containing mixed attribute types, such as continuous and categorical features. In SMOTE-NC, continuous attributes are over-sampled by interpolating between two samples the same way in SMOTE and categorical attributes are over-sampled by selecting one of the k-nearest neighbours for each example in the feature space.

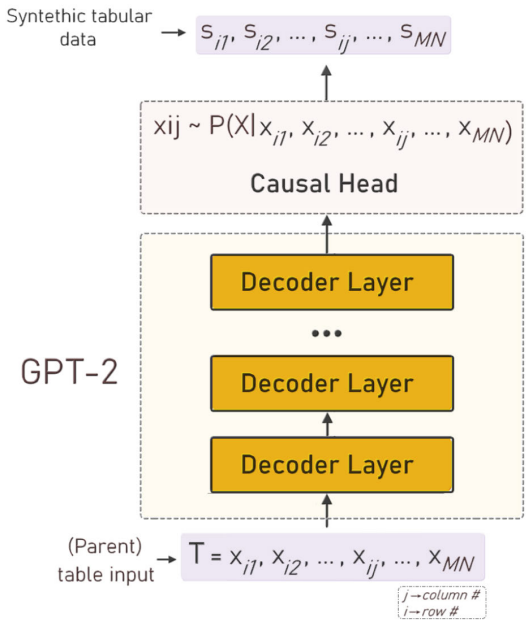



**FIGURE 3.** REaLTabFormer architecture for non-relational tabular data: the model learns the conditional distribution of the values of each of *M* columns for each *N* rows of table *T*. This method allows the model to sequentially generate the values of new synthetic observations.

More recent works use deep learning models to generate synthetic tabular data. Solatorio et al. [31] proposed REalTabFormer, a transformer-based framework for data generation. REalTabFormer uses an autoregressive model based on the GPT-2 [49] to generate non-relational tabular data, as shown in the proposed architecture in Figure 3. It also can generate synthetic relational data using a sequence-to-sequence model. Additionally, REaLTabFormer presents a statistical approach for monitoring overfitting and a target masking strategy to mitigate the risks associated with data copying to address the overfitting and data-copying problems. The method calculates the Distance to Closest Record by measuring the similarity between synthetic generated samples and the original data. Then, the distribution of these minimum distances is employed to assess overfitting. Moreover, it uses the Quantile Difference ($Q_\delta$) statistic to compare two samples and determine if they originate from different distributions. Overfitting can be detected by comparing $Q_\delta$ values against a threshold determined through bootstrapping. REaLTabFormer optimizes sample generation by building observations sequentially, one token at a time, and implementing a constrained sampling strategy using column-specific vocabularies. Invalid tokens for each column are excluded during generation.

GReaT is a pre-trained autoregressive language model that leverages a transformer architecture to generate synthetic tabular data for applications such as synthetic data generation, data augmentation, and privacy-preserving data sharing. By handling both categorical and numerical data and learning the long-range dependencies between the different columns of a tabular dataset, GReaT avoids lossy preprocessing operations such as encoding or binning categorical features. Additionally, unlike REalTabFormer, GReaT can generate data for imbalanced datasets by easily allowing itself to be conditioned on certain classes of values that may be under-represented in the dataset. Notably, GReaT is pre-trained on a large dataset of text and code, while REalTabFormer is trained on a dataset of tabular data.

Although REaLTabFormer and GReaT present promising results in some application contexts, they have not been compared to more traditional methods, like SMOTE, on their use and impact on data classification tasks while generating synthetic tabular data representing IIoT network traffic.

### C. RELATED WORK

Much of the work on data augmentation is focused on digital image data [20], [50], [51], [52], [53]. A variety of augmentation techniques involving geometric transformations, color space augmentations, feature space augmentation, or the employment of GANs are used to expand the size of the datasets and mitigate their class imbalance, and then, predominantly, employed in computational vision tasks such as image classification, segmentation, object detection, or background subtraction [54].

It's worth noting that data augmentation is also employed in NLP to address the challenges of low-resource domains, new tasks, and the need for large-scale training data. In a comprehensive survey, Feng et al. [25] discuss the growing interest in the NLP area and summarize the literature in a structured manner. They cover methodological approaches, popular NLP applications, as well as current open challenges.

Some authors have explored the concept of data augmentation in other domains, such as radio signal classification. In a recent study, Zheng et al. [55] used data augmentation techniques on radio signal data to improve the performance of Automatic Modulation Classification (AMC), which involves converting time-domain signals into the frequency domain via the Short-Time Fourier Transform (STFT). They inject noise into the signals with bidirectional noise masks before converting the signals back to the time domain using the Inverse Fourier Transform (IFFT). The augmented signals are then used to train a deep CNN (Convolutional Neural Networks). The results show that data augmentation significantly improved the automatic classification accuracy of radio signal modulations on the RadioML 2016.10a dataset [56]. In the context of data augmentation applied to tabular datasets,

Nakhwan and Duangsoithong [57] delve into three distinct augmentation methodologies: Bootstrap, GANs, and Autoencoder-based augmentation. The primary objective is to refine predictive accuracy by expanding sample sizes within datasets characterized by insufficient observations. The study utilizes these augmentation techniques across eight varied datasets designed for binary classification, spanning both health-related and earth science domains available in data repository websites.

The evaluation framework involves the generation of supplementary data through augmentation, its integration with the original dataset, and the subsequent evaluation of performance across four classifier models, namely Support Vector Machines (SVM), DT, Logistic Regression (LR), and k-Nearest Neighbors (KNN) classifiers. Notably, the experimental findings delineate that the proposed augmentation approach, employing Autoencoder and GANs, consistently outperforms the original dataset. In contrast, the Bootstrap method manifests as the least performant in terms of predictive accuracy.

However, our primary focus is on investigating data augmentation techniques applied to tabular data to more effectively address the challenge of imbalanced datasets in IDSs, whether for conventional computer networks or IoT/IIoT networks.

An example of a study carried out in this field, Gupta et al. [58] investigate the utilization of the SMOTE data augmentation algorithm to balance a dataset for Intrusion Detection in Internet of Medical Things (IoMT) networks. The authors demonstrate that, overall, using SMOTE data augmentation to address the challenge of imbalanced data yields superior performance when compared to establishing class weight ratios for the Random Forest classifier.

Jmila and Khedher [59] assess a collection of shallow ML classifiers (Adaboost, DT, Gradient boosting, LR, Random Forest, Support Vector Classifier and deep learning networks) against a variety of adversarial attacks (Gaussian noise, White-box attacks, Gray/black-box attacks), and investigate the impact of Gaussian data augmentation on enhancing the robustness of these classifiers, utilizing the NSL-KDD dataset [60] and the UNSW-NB15 dataset [61] as their testbed.

Specifically, the authors employ Gaussian data augmentation, which involves expanding the original dataset by adding copies of the original samples with Gaussian noise. This technique induces the model to produce the same prediction for a non-augmented instance and its slightly perturbed version, thereby enhancing its generalization capabilities.

In their study, Liu et al. [62] employed the Adaptive Synthetic (ADASYN) [63] oversampling technique in conjunction with the Gradient Boost DT model, namely LightGBM [64], to address the challenge of low detection rates for minority attacks stemming from imbalanced training data. The authors conducted a comprehensive evaluation of their approach across three distinct Network Intrusion datasets (NSL-KDD, UNSW-NB15 and CICIDS2017 [65]), employing different ML algorithms and data augmentation techniques. Specifically, their research not only demonstrated the positive impact of data augmentation on ML-based IDSs, particularly when dealing with imbalanced datasets but also established that ADASYN outperforms alternative techniques such as SMOTE and ROS for the datasets under consideration.

Moving into the realm of deep learning-based data augmentation models, Zhang et al. [66] propose an IDS tailored for IoT networks based on an Improved Conditional Variational Autoencoder (ICVAE-BSM) and development of the regular SMOTE known as Borderline-SMOTE [67]. Unlike conventional SMOTE, which uniformly generates synthetic samples within the minority class, Borderline-SMOTE takes a distinct approach. It specifically identifies and oversamples instances positioned near the class boundary, situated between the minority and majority classes. This approach aims. This approach aims to mitigate the problem of misclassification of minority class instances that are close to the decision boundary [67]. The authors propose a three-step workflow: first, they train the ICVAE model on the unbalanced data sets, then they use Borderline-SMOTE to generate samples in the latent space of the ICVAE decoder, thus creating synthetic data samples. The third phase involves fine-tuning the network, consisting of the combination of the ICVAE encoder and a softmax classifier.

Moreover, the authors show that, in general, the proposed model performs better when trained with the augmented data in the three data sets tested. In addition, they compare the model with analogous data augmentation and classification approaches found in the literature, such as CVAE [68], GAN [69] and SMOTE-based models, and demonstrate the superiority of ICVAE-BSM.

Sauber-Cole and Khoshgoftaar [70] undertake a thorough investigation into the application of GANs for addressing class imbalances in tabular datasets, with a specific focus on their utilization in intrusion detection tasks. The study emphasizes the prevalence of Conditional GAN (CGAN) architecture [71], the optimization using CNN, and the adoption of minority-class-sensitive metrics such as F1-score. The sustained popularity of SMOTE as a baseline technique and the consistent improvement of GAN performance over time are discussed. Specific challenges, such as data type heterogeneity, and the need for more generalizable data preprocessing methods, are examined. The article raises questions about the universal applicability of GANs, considering challenges in hyperparameter optimization and computational resources. It identifies research gaps, including the lack of investigations into optimal levels of class imbalance and data generation. Overall, the authors conclude that GANs have demonstrated considerable success in restoring balance to these datasets.

Li et al. [72] introduce the Denoise Autoencoder-GAN model for the detection of abnormal network traffic, while leveraging data augmentation to enhance its ability to discern anomalous network patterns effectively. The process commences with a feature extraction phase, which extracts both spatial and temporal features from raw traffic data, followed by a feature selection step to identify the most informative attributes. To augment the dataset and enable abnormal traffic detection, the authors employ multiple denoising autoencoders to form a Pseudo Anomaly generator. The

discriminator undergoes training on the anomalies generated by the PA generator, whose role is to distinguish abnormal traffic instances from their normal counterparts within the dataset. Furthermore, an adversarial learning component is introduced to adjust the discriminator's classification boundary, aligning it more closely with the distribution of normal traffic patterns, thereby enhancing both sensitivity and precision.

The effectiveness of the proposed model is evaluated on three network intrusion datasets, namely UNSW-NB15, NSL-KDD, and Kitsune [73]. The results of these evaluations demonstrate that the Denoise Autoencoder-GAN model surpasses the state-of-the-art models available at the time of publication.

Alqarni and El-Alfy [74] employed a Conditional Tabular Generative Adversarial Network (CTGAN) model to address the class imbalance by generating minority class samples within datasets. The authors proposed an IDS that combines CTGAN-based data augmentation with shallow ML algorithms: SVM, KNN, and DT. This configuration was evaluated using the imbalanced NSL-KDD dataset, and the results demonstrated that CTGAN data augmentation significantly improved the performance of imbalanced learning when compared to the non-augmented dataset, particularly in the context of SVM and DT classification models. Conversely, KNN did not exhibit noteworthy performance improvements, as it is less affected by class imbalance. Furthermore, the research highlighted CTGAN's effectiveness in accurately modeling the distribution of discrete features compared to continuous features.

Drawing inspiration from data augmentation algorithms used in digital images, Wang et al. [75] introduced the *random_shuffle* data augmentation method. This method involves transforming the feature vectors of network traffic data into grayscale images and subsequently applying operations such as horizontal and vertical flipping and random cropping to increase sample diversity. Additionally, the Fisher-Yates shuffling algorithm [76] is employed to randomly perturb the positions of these features. The authors applied the augmented data to train an enhanced self-supervised learning algorithm (Bootstrap Your Own Latent [77]), and evaluated its performance on various intrusion detection benchmark datasets (UNSW-NB15, NSL-KDD, KDD CUP99, CICIDS2017, and CIDDS-001 [78]). They compared the proposed approach with other models from the literature, and the results demonstrated competitive performance, although variations without data augmentation were not explored.

Kumar et al. [79] explore the use of Transformer-based pre-trained models for conditional data augmentation. The authors show that the addition of class labels to the text sequences gives a simple yet effective way of conditioning the pre-trained models for data augmentation. Besides that, this study looks into how pre-trained model-based data augmentation differs concerning the diversity of data, as well as to what extent these models are able to preserve the label conditioning.

Glass et al. [80] propose a self-learning strategy where tables are randomly ablated from the corpus and the retrieval-based model is trained to reconstruct the original values or headers given the partial tables as input. The authors train a dense neural retrieval model to encode table parts to vectors, and then the end-to-end model is trained to perform table augmentation tasks.

In this work, we evaluate the impact of traditional and deep learning-based tabular data augmentation methods on IIoT network traffic classification. We use augmentation methods to balance training datasets and compare the performance of several multiclass classification algorithms when trained with the non-augmented dataset and the datasets generated by each evaluated augmentation method.

## III. EVALUATION FRAMEWORK

A typical data augmentation workflow starts with the data pre-processing, and then the dataset is split into training and testing sets. Data augmentation techniques, such as oversampling, are then applied to minority classes, thus balancing the dataset. The augmented dataset is then used to train machine learning models for classification tasks, as illustrated in Figure 1. To evaluate different data augmentation solutions and their impact on various ML algorithms used to identify attacks on IIoT networks, we established a systematic assessment that expands the commonly used workflow for data augmentation.

Figure 4 presents an overview of our evaluation framework. In summary, the evaluation process starts with data pre-processing, followed by the dataset into training and test sets. Hence, our test set does not contain synthetically created data to reduce the chance of a biased evaluation of the impact of augmentation algorithms on classification performance. Then, on the training data, we apply both undersampling on the majority class and oversampling on underrepresented classes. We have five scenarios to evaluate-comprising using no-augmentation and four augmentation algorithms from the literature. Each scenario generates a distinct dataset that will be used to train classification models. Also, the use of feature engineering techniques varies according to the requirements of each scenario. Finally, we use the five distinct augmented sets to train each considered ML model. All trained models are then used to classify the same test set, which does not contain synthetically generated data.

In the following, we detail each phase of our evaluation framework.

### A. METHODOLOGY

Our strategy contains five main stages: (**I**) dataset pre-processing, (**II**) random undersampling of the majority class, (**III**) data splitting, (**IV**) data augmentation scenarios, and (**V**) evaluation of attack identification alternatives, as represented in Figure 4.

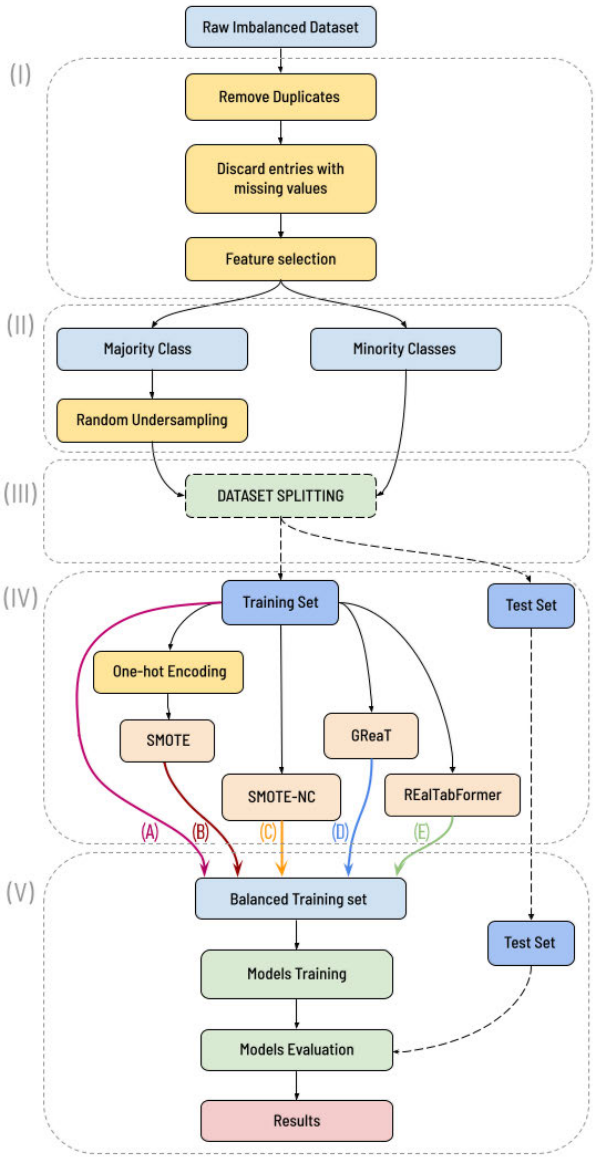



**FIGURE 4.** Workflow with alternative scenarios for IIoT traffic data augmentation and classification evaluation.

**Dataset pre-processing (I)** - Initially, the raw dataset is cleaned by eliminating lines containing missing values and removing duplicate lines. This process is necessary to ensure the quality and reliability of the data. Additionally, attributes that could potentially identify the source or type of attack are removed. These attributes were specific to the dataset collection setup and did not contribute to a more general analysis of network traffic classification. Both the input and output datasets resulting from this step are imbalanced.

**Random undersampling of the majority class (II)**. At this stage, we employ a technique of random undersampling on the majority class, *Normal* traffic type, reducing its instances to 100,000 (this number may be experimentally tuned considering available resources). Undersampling is required as increasing the number of rows in each of the other 14 classes to match the original number of rows of the majority class (over 1,6 million rows) would lead to a dataset so large (with over 24 million rows) that processing it would require extensive resources.

**Dataset Splitting (III)**. After applying random undersampling to the majority class, the dataset is split into training (80%) and testing (20%) sets. We exclusively use data from the original dataset for testing (do not use any synthetic data generated by data augmentation algorithms).

**Data Augmentation Scenarios (IV)**. In this stage, we create distinct balanced datasets by evaluating and applying data augmentation techniques. Each data augmentation algorithm has specific requirements and may demand distinct data transformations before augmentation. We consider five scenarios (Figure 4):

- **Scenario A** serves as the control condition. In this scenario, we do not apply any data augmentation algorithm, which allows us to establish a baseline performance metric. This approach enables a fair and comparative analysis of the subsequent data augmentation techniques.
- **Scenario B**, we use SMOTE, which cannot directly handle categorical features. Thus, we transform the categorical data with one-hot encoding before applying the augmentation.
- **Scenario C**, we apply SMOTE-NC directly, thus taking advantage of the algorithm's capability to handle categorical features effectively.
- **Scenario D**, we fine-tune the GReaT model in the training portion of the dataset and take advantage of this model's ability to generate class-conditioned data. The process begins by dividing the dataset into two distinct groups, each containing approximately the same number of instances. These groups are formed by separating the data into two categories of attacks, disregarding instances with no attack data. Subsequently, two distinct models are trained on each of these groups. These models are then employed to generate data for their respective types of attacks until the number reaches 100,000, including the original data. Finally, these two balanced sets are concatenated, resulting in a complete and balanced dataset. This procedure was necessary due to the inherent limitations of the graphics card memory. Under normal conditions, the training would have been carried out on the entire dataset. Once again, due to practical considerations related to computational resources availability, this model was fine-tuned for only one epoch.
- **Scenario E**, we fine-tune the REalTabFormer model following a similar procedure as the GReaT model, yet there is a crucial distinction. Unlike the GReaT model, the REalTabFormer lacks the ability to generate attack-conditioned data. After partitioning the data into two distinct attack groups with roughly the same size, we generate data until each class comprises at least 100,000 instances. Finally, the generated data

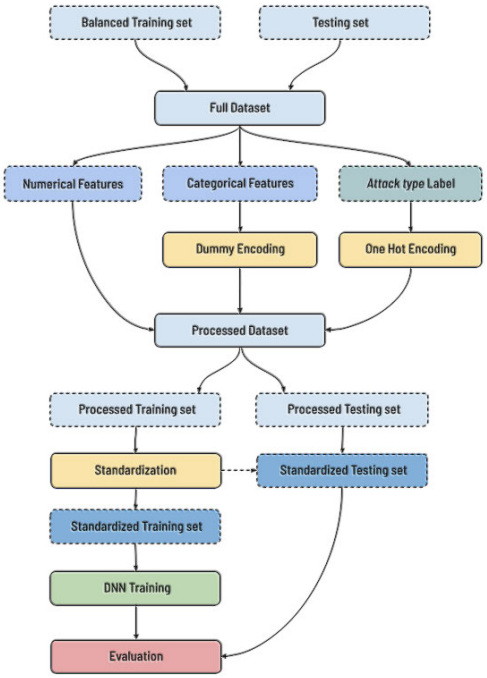



**FIGURE 5.** MLP data augmentation workflow (stage V). It includes a block for converting categorical variables into binary variables, as well as label binary encoding. In addition, the data is standardized to facilitate learning.

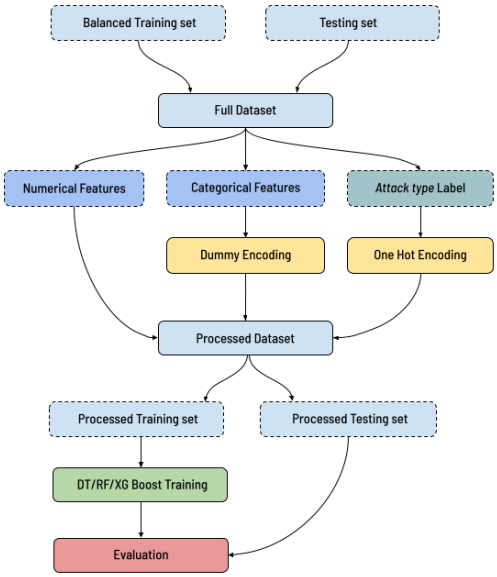



**FIGURE 6.** DT-based workflow (stage V). The implemented models of DT, Random Forests (both from scikit-learn), and XGBoost do not inherently support categorical values. These categorical features must be converted to numerical values for these models to process them effectively.

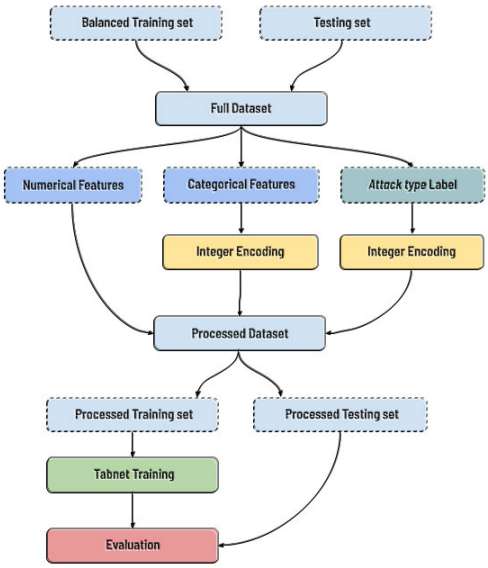



**FIGURE 7.** Tabnet workflow (stage V). The categorical attributes and attack labels were converted into integer data.

is concatenated with the original data resulting in a balanced dataset.

**Evaluation of Attack Detection Models (V).** We instantiate and evaluate the models outlined in Section II-A. Each model has unique pre-processing requirements and data-handling capabilities. Figures 5, 6, and 7 present the workflow for the MLP, the tree-based algorithms (DT, Random Forests, and XGBoost), and TabNet, respectively. The main differences in the workflows are related to categorical features and label encoding. In the workflow of DT, Random Forests, and XGBoost, categorical features are converted to numerical values for the augmentation models to process them effectively. The data augmentation workflow for the MLP includes a task for converting categorical variables into binary variables, label encoding, and data standardization to facilitate learning. The workflow for Tabnet includes converting categorical attributes and attack labels into integer data.

### B. EVALUATION METRICS

In malicious traffic identification context, *True Positives* (TP) occurs when a model correctly classifies network data representing malicious traffic. A *False Positive* (FP) means the model classifies regular network communication data as malicious. *True Negatives* (TN) occur when the model correctly identifies regular network communications traffic, and *False Negatives* (FN) happen when a model classifies network traffic data generated by a cyberattack as a regular one. FN are particularly critical in this context because the worst possible outcome is missing the identification of a cyberattack.

#### 1) ACCURACY

It is the ratio of the sum of True Positives and True Negatives to the total number of instances, as defined in Equation 1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

#### 2) PRECISION

Precision is the ratio of True Positives to the sum of True Positives and False Negatives, as defined in Equation 2.

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

#### 3) RECALL

Also known as sensitivity, recall measures the proportion of actual positive instances correctly identified as such. It is computed by dividing the True Positives by the sum of True Positives and False Negatives, as in Equation 3.

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

#### 4) F1-SCORE

F1-Score combines Precision and Recall as defined in Equation 4. It provides a single metric that ranges from 0 (worst) to 1 (best) and summarizes the performance.

$$F_1\text{-}Score = 2 \cdot \left( \frac{Precision \cdot Recall}{Precision + Recall} \right) \tag{4}$$

In multiclass classification, evaluating model performance involves considering metrics beyond binary classification.

#### 5) WEIGHTED AVERAGE

Weighted averaging takes into account class imbalance by assigning weights based on the number of instances in each class. This approach gives more significance to classes with a larger representation in the dataset.

#### 6) MACRO-AVERAGE

The macro-average, on the other hand, treats all classes equally, providing a uniform assessment across the entire classification task. Macro-averaging calculates metrics (e.g., precision, recall, or F1-score) for each class independently, and then averages them. We use this approach as it is the one that better measures the classifiers' performance for minority classes.

$$Macro - Average = \frac{1}{C} \sum_{i=1}^{C} M_i \tag{5}$$

where $C$ is the total number of classes and $M_i$ is the metric calculated for the $i$-th class.

## IV. EXPERIMENTAL EVALUATION

We implemented the framework described in the previous section to evaluate GPT and interpolation-based data augmentation strategies and their impact on ML techniques for IIoT traffic classification.

### A. EXPERIMENTAL ENVIRONMENT

Experiments were conducted using Python scripts and Jupyter Notebooks, which are made available in a public GitHub repository [81]. This repository provides the necessary documentation to enable reproducibility. Most of the experiments were carried out on an HP Victus, 32 GB of memory, Intel i5-12500Hx16 processor, with Nvidia GeForce RTX 4600 and Pop!_OS 22.04 LTS operating system. Due to the higher computational requirements, the training, and data generation of the GReaT model was carried out on a computer with the operating system Windows 10, with 32 GB of RAM, an Intel i7-10700K processor and an Nvidia GeForce RTX 3090 graphics card.

**TABLE 3. Edge-IIoT dataset split. Number of entries per class without augmentation.**

| Traffic Type | Training set | Test set |
|---|---|---|
| Normal | 1,091,222 | 272,776 |
| DDoS (UDP) | 96,966 | 24,601 |
| DDoS (ICMP) | 54,438 | 13,501 |
| SQL injection | 40,755 | 10,071 |
| Vulnerability scanner | 40,086 | 9,940 |
| DDoS (TCP) | 40,053 | 10,009 |
| Password | 39,825 | 10,108 |
| DDoS (HTTP) | 38,916 | 9,628 |
| Uploading | 29,534 | 7,273 |
| Backdoor | 19,244 | 4,782 |
| Port Scanning | 15,915 | 4,062 |
| Cross-site scripting (XSS) | 12,031 | 3,035 |
| Ransomware | 7,763 | 1,926 |
| Fingerprinting | 707 | 146 |
| MITM | 282 | 76 |

#### 1) DATASET

We used the Edge-IIoT [12] dataset. The dataset was created for cybersecurity IIoT applications based on data from an IIoT testbed containing several devices, sensors, protocols, and configurations. Data is classified into 15 classes, one representing *Normal* or *Non-attack* IIoT network traffic and 14 classes representing network traffic resulting from cyberattacks. The dataset was constructed by capturing network traffic flows using Wireshark, selecting 61 features from 1,176 found features, and saving the dataset in CSV files. For our analysis, we used the public version of the dataset, accessible on Kaggle [82]. Table 2 presents the distribution of rows per class. To evaluate the augmentation and classification algorithms, we used the methodology described in Section III. After removing duplicates and entries with empty values and applying feature selection, we split the dataset and generated the test set to evaluate the performance of classification methods. Table 3 provides specific entry counts for each class in the test dataset and in the non-augmented training set.

#### 2) MODEL HYPERPARAMETERS

We primarily adhered to the default values for hyperparameters with minimal customization. An exception was TabNet, where we explored a range of hyperparameters, including the width of the decision prediction layer, the width of attention embedding for each mask, the number of steps, gamma, and momentum. We also set specific training parameters, such as the maximum number of epochs, patience, batch size, and virtual batch size. In the Random Forest algorithm,

**TABLE 4. Hyperparameters of the ML models. Configuration hyperparameters for each ML model, along with the main framework and version used. If not explicitly specified, default parameters of the respective framework are assumed.**

| Model | Main Framework | Hyperparameters |
|---|---|---|
| Decision Trees | scikit-learn (v1.0.2) | Criterion: Gini<br>Splitter: Best<br>Minimum Samples Split: 2<br>Minimum Samples Leaf: 1 |
| Random Forest | scikit-learn (v1.0.2) | Criterion: Gini<br>Splitter: Best<br>Number of Estimators: 10<br>Minimum Samples Split: 2<br>Minimum Samples Leaf: 1<br>Maximum Features: sqrt(`n_features`) |
| TabNet | pyTorch-tabnet (v4.0) | Decision Prediction Layer Width (`n_d`): 64<br>Decision Prediction Layer Width (`n_a`): 64<br>Attention Embedding Width (`n_a`): 64<br>Number of Steps (`n_steps`): 5<br>Mask Feature Reusage Coefficient: 1.5<br>Scheduler Parameters:<br>- Gamma: 0.95<br>- Step Size: 20<br>Training Parameters:<br>- Optimizer: Adam, Learning Rate (LR) = 0.022<br>- Epochs: 100<br>- Batch Size: 16384<br>- Virtual Batch Size: 256 |
| XGBoost | XGBoost (v1.6.2) | Maximum Depth: 8<br>Learning Rate: 0.1<br>Number of Estimators: 100<br>Subsample: 0.7<br>Base Score: 0.5 |
| DNN (MLP) | Tensorflow-Keras (v1.13.1) | Architecture:<br>- Activation: ReLU<br>- First Layer Nodes: Equal to the Number of Features<br>- Hidden Layers Nodes: 256, 128, 64, 32<br>- Batch Norm & Dropout (20%) after each Hidden Layer<br>- Final Layer Nodes: Equal to the Number of Classes<br>Training Parameters:<br>- Loss Function: Sparse Categorical Cross Entropy<br>- Optimizer: Adam, Learning Rate (LR) =0.001<br>- Reduce LR On Plateau (monitor=Loss, factor=0.3)<br>- Epochs: 100<br>- Batch Size: 512 |

we only changed the number of estimators to 10. In the case of XGBoost, we adjusted variables such as maximum depth, learning rate, number of estimators, and subsample. The DNN (MLP) was configured with layers activated by ReLU, starting with the first layer matching the number of features. Subsequent layers contained 256, 128, 64, and 32 nodes, respectively, and, in the final layer, the number of nodes equivalent to the number of classes. Additional details, including the classification function, number of epochs, and batch size, are summarized in Table 4.

#### 3) DATA AUGMENTATION HYPERPARAMETERS

We used the implementation of SMOTE and SMOTE-NC available at the `imblearn` library with the default hyperparameters. Concerning REalTabFormer, we established the training parameters and set the batch size to 64 and 10 epochs. However, the training was interrupted before this number of epochs due to the Early Stopping with $\mathbf{Q}'_{\delta}$ condition, as described in the [31]. In the case of GReaT, we chose to train the model for a single epoch driven by computational constraints.

**TABLE 5.** Hyperparameters of the ML models. If not specified, assume the default parameter of the respective framework.

| Augmentation Algorithm | Main Frameworks | Hyperparameters |
|---|---|---|
| SMOTE | imblearn (v0.11.0) | Nearest Neighbors: 5<br>Sampling Strategy: `not majority` (resample all classes but the majority class) |
| SMOTE-NC | imblearn (v0.11.0) | Nearest Neighbors: 5<br>Sampling Strategy: `not majority` (resample all classes but the majority class) |
| REalTabFormer | PyTorch, Transformers, c (v0.1.2) | Model Type: Tabular<br>Numeric Maximum Length: 11<br>Training Parameters:<br>- Batch Size: 64<br>- Epochs: 10<br>- Gradient Accumulation Steps: 4 |
| GReaT | PyTorch, Transformers | Training Parameters:<br>- Pretrained GPT version: distilgpt2<br>- Epochs: 1<br>- Batch Size: 16 |

### B. IIOT TRAFFIC CLASSIFICATION RESULTS

Data augmentation techniques generate more synthetic data until all classes have the same number of rows as the majority class, which may significantly increase the size of the dataset and lead to high computational resource consumption during classification. To deal with such an issue, we undersampled the majority class on the training set, selecting 100,000 rows, as described in Section III-A. Then, we executed the remaining steps of our evaluation methodology and assessed the performance of each evaluated model and configuration on classifying network data on the test set.

First, we present precision and recall results for per-class classification for each method and training dataset. In green highlight are the best results for each class and metric, while in red are the worst.

Table 6 presents the results for the DT classifier. Training the model with augmented data improved the precision for seven and the recall for five classes. However, no augmentation method improved the model's performance for all classes compared to the one achieved by the model when trained with the original dataset. Using SMOTE slightly increased the precision for four classes, while the other methods slightly increased the precision for five of the dataset's classes. On the other hand, using data augmentation significantly reduced the precision in identifying the Fingerprinting attack. When using data generated by RealTabFormer, the precision dropped to half. Precision dropped to approximately one-third when using training data from SMOTE or SMOTE-NC, and it was reduced to approximately 5% of the original one when using data generated by GReaT. Using synthetically generated data for training reduced the recall values of the DT classifier for almost all classes. The worst performance was achieved using data generated by GReaT, which led to the worst recall in over 65% of the classes.

Table 7 presents precision and recall when using the RF algorithm for classification. The performance achieved by the model trained with unaugmented data was significantly lower than that obtained for DT, especially for recall. However, the model trained on data generated by SMOTE achieved even worse results than the model trained on unaugmented data in almost all classes (with precision and recall values lower than 10% for several of them). The training files with data generated by SMOTE-NC led to the best precision results in most classes, while GReaT was the one that led to the best recall values in most classes. However, using GReaT data led to significant drops in recall when identifying Fingerprinting and MITM attacks.

The precision and recall obtained for each class using MLP are presented in Table 8. In this case, we highlight the classifier was not able to balance the precision and recall results (with one of the metrics very high and the other very low) in several configurations, as in the case of the Port Scanning attack without using data augmentation, in case of identifying the Fingerprinting attack when trained with data from SMOTE, and the XSS attack when trained with data from SMOTE-NC. As in the previous classifiers, the data generated by GReaT led to a low performance in MITM identification (in this case, a precision below 20%).

The precision and recall obtained using Tabnet are in Table 9). The overall performance achieved with this classifier is significantly lower than other methods. Data augmentation proved valuable in identifying some attacks, such as UDP DDoS and Port Scanning, but data generated by SMOTE-NC led to the worst result among all methods for seven classes in terms of precision and for six classes in terms of recall.

Table 10 presents the performance metrics for the XGBoost classifier. There was significant variation in some attacks, such as Uploading and Fingerprinting. In the case of MITM,

**TABLE 6.** Performance of the Decision Tree: Precision and Recall (%) Across Different Data Augmentation Techniques.

| | Precision | | | | | Recall | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *No Augmentation* | SMOTE | SMOTE-NC | REalTabFormer | GReaT | *No Augmentation* | SMOTE | SMOTE-NC | REalTabFormer | GReaT |
| Normal | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| DDoS (UDP) | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 92.50 |
| DDoS (ICMP) | 100.00 | 100.00 | 100.00 | 100.00 | 99.87 | 99.99 | 99.77 | 99.76 | 99.99 | 98.61 |
| SQL injection | 82.82 | 83.58 | 83.53 | 83.05 | 85.45 | 83.82 | 83.32 | 83.37 | 80.67 | 75.32 |
| Vulnerability sc. | 96.27 | 95.99 | 96.05 | 95.66 | 95.66 | 96.58 | 96.43 | 96.31 | 96.14 | 95.76 |
| DDoS (TCP) | 100.00 | 100.00 | 100.00 | 99.98 | 99.87 | 100.00 | 100.00 | 100.00 | 100.00 | 91.16 |
| Password | 82.97 | 83.11 | 83.00 | 85.44 | 85.13 | 82.34 | 82.67 | 82.57 | 80.79 | 73.31 |
| DDoS (HTTP) | 92.78 | 93.18 | 93.09 | 79.30 | 87.99 | 93.03 | 87.34 | 87.48 | 85.28 | 86.84 |
| Uploading | 80.08 | 79.82 | 79.81 | 85.90 | 83.59 | 79.02 | 80.42 | 80.27 | 74.95 | 72.18 |
| Backdoor | 95.82 | 96.81 | 96.77 | 97.74 | 98.25 | 96.34 | 94.58 | 94.56 | 95.17 | 90.46 |
| Port scanning | 96.73 | 96.70 | 96.75 | 95.98 | 94.00 | 96.01 | 89.34 | 89.32 | 95.79 | 93.72 |
| XSS | 83.62 | 70.92 | 71.03 | 68.36 | 70.75 | 83.43 | 84.71 | 84.65 | 90.54 | 87.58 |
| Ransomware | 92.70 | 88.92 | 88.96 | 93.70 | 94.83 | 92.32 | 92.47 | 92.47 | 91.95 | 90.55 |
| Fingerprinting | 62.89 | 21.58 | 21.41 | 34.41 | 3.07 | 68.49 | 76.71 | 76.71 | 73.29 | 57.53 |
| MITM | 100.00 | 100.00 | 100.00 | 88.37 | 1.98 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |

**TABLE 7.** Performance of the Random Forest: Precision and Recall (%) Across Different Data Augmentation Techniques.

| | Precision | | | | | Recall | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | No Augmentation | SMOTE | SMOTE-NC | REalTabFormer | GReaT | *No Augmentation* | SMOTE | SMOTE-NC | REalTabFormer | GReaT |
| Normal | 90.72 | 70.62 | 100.00 | 100.00 | 100.00 | 98.61 | 99.95 | 100.00 | 100.00 | 100.00 |
| DDoS UDP | 100.00 | 99.94 | 100.00 | 100.00 | 92.42 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| DDoS ICMP | 99.96 | 99.92 | 99.73 | 99.99 | 99.96 | 99.99 | 99.84 | 100.00 | 100.00 | 99.88 |
| SQL injection | 73.66 | 22.32 | 83.82 | 81.56 | 80.33 | 43.89 | 82.86 | 83.08 | 80.55 | 85.54 |
| Vulnerability sc. | 94.88 | 91.94 | 96.02 | 95.83 | 96.11 | 93.51 | 11.41 | 97.66 | 96.99 | 97.80 |
| DDoS TCP | 100.00 | 40.55 | 100.00 | 100.00 | 98.83 | 100.00 | 100.00 | 100.00 | 100.00 | 99.99 |
| Password | 77.61 | 5.56 | 83.88 | 80.92 | 79.58 | 46.27 | 4.64 | 82.25 | 85.26 | 85.07 |
| DDoS HTTP | 72.31 | 0.44 | 88.37 | 86.88 | 87.47 | 53.01 | 0.28 | 93.02 | 80.86 | 90.43 |
| Uploading | 65.79 | 25.67 | 79.53 | 74.51 | 76.46 | 46.87 | 22.68 | 83.02 | 89.42 | 85.94 |
| Backdoor | 92.81 | 0.00 | 94.40 | 94.98 | 93.98 | 91.77 | 0.00 | 98.67 | 98.50 | 99.10 |
| Port Scanning | 95.86 | 1.01 | 95.27 | 99.83 | 99.56 | 85.79 | 100.00 | 96.58 | 95.75 | 93.85 |
| XSS | 74.96 | 93.28 | 86.29 | 92.32 | 91.30 | 56.90 | 9.91 | 70.80 | 69.41 | 69.61 |
| Ransomware | 88.53 | 0.00 | 92.63 | 91.90 | 90.60 | 86.02 | 0.00 | 93.85 | 94.75 | 97.27 |
| Fingerprinting | 66.44 | 50.68 | 76.71 | 71.92 | 58.22 | 61.39 | 91.36 | 23.88 | 51.47 | 6.67 |
| MITM | 98.68 | 98.68 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 98.70 | 2.86 |

only the data generated by GReaT led to a large (negative) difference in performance. SMOTE was the data augmentation algorithm that led to a more balanced performance.

Table 11 presents the macro-average performance results for the various data augmentation and classification algorithms evaluated. The RF, Tabnet and MLP classifiers significantly benefitted from data augmentation. For instance, the F1-score achieved by RF when using the non-augmented data from training was 81%, while the F1-score when using data generated by REalTabFormer was over 90%. On the other hand, the XGBoost and DT classifiers didn't significantly benefit from data augmentation but still got the best performance among all evaluated classifiers (with F1-score over 91%). The performance of Tabnet was significantly lower than the one of other classifiers, even when trained with augmented data.

**TABLE 8.** Performance of the MLP: Precision and Recall (%) Across Different Data Augmentation Techniques.

| | Precision | | | | | Recall | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | No Augmentation | SMOTE | SMOTE-NC | REalTabFormer | GReaT | *No Augmentation* | SMOTE | SMOTE-NC | REalTabFormer | GReaT |
| Normal | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 99.99 | 100.00 |
| DDoS (UDP) | 99.96 | 100.00 | 99.99 | 99.99 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 99.79 |
| DDoS (ICMP) | 99.81 | 100.00 | 100.00 | 100.00 | 100.00 | 99.72 | 99.77 | 99.46 | 99.64 | 99.36 |
| SQL injection | 45.54 | 83.58 | 51.91 | 48.09 | 64.90 | 91.90 | 83.32 | 47.19 | 57.59 | 28.85 |
| Vulnerability sc. | 99.85 | 95.99 | 97.64 | 99.82 | 99.90 | 84.43 | 96.43 | 83.98 | 83.64 | 83.81 |
| DDoS (TCP) | 82.12 | 100.00 | 100.00 | 94.82 | 100.00 | 100.00 | 100.00 | 58.11 | 64.95 | 57.96 |
| Password | 92.81 | 83.11 | 48.78 | 51.32 | 46.25 | 17.49 | 82.67 | 63.15 | 48.45 | 80.86 |
| DDoS (HTTP) | 73.94 | 93.18 | 94.39 | 60.31 | 94.45 | 95.25 | 87.34 | 60.15 | 61.15 | 53.95 |
| Uploading | 67.27 | 79.82 | 67.01 | 90.41 | 66.27 | 48.33 | 80.42 | 48.04 | 30.32 | 48.11 |
| Backdoor | 95.98 | 96.81 | 99.75 | 96.68 | 99.93 | 98.29 | 94.58 | 93.12 | 97.34 | 93.12 |
| Port scanning | 100.00 | 96.70 | 48.28 | 50.60 | 47.49 | 49.98 | 89.34 | 99.85 | 91.78 | 99.85 |
| XSS | 60.40 | 70.92 | 32.73 | 34.26 | 32.33 | 35.88 | 84.71 | 85.40 | 88.90 | 93.71 |
| Ransomware | 100.00 | 88.92 | 95.89 | 97.54 | 100.00 | 88.84 | 92.47 | 92.16 | 90.76 | 88.84 |
| Fingerprinting | 67.27 | 21.58 | 25.89 | 67.12 | 41.35 | 46.58 | 76.71 | 84.25 | 67.12 | 67.12 |
| MITM | 100.00 | 100.00 | 100.00 | 100.00 | 19.24 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |

**TABLE 9.** Performance of the TabNet: Precision and Recall (%) Across Different Data Augmentation Techniques.

| | Precision | | | | | Recall | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | No Augmentation | SMOTE | SMOTE-NC | REalTabFormer | GReaT | *No Augmentation* | SMOTE | SMOTE-NC | REalTabFormer | GReaT |
| Normal | 100.00 | 99.99 | 100.00 | 99.99 | 99.92 | 100.00 | 100.00 | 99.99 | 100.00 | 100.00 |
| DDoS UDP | 9.29 | 99.75 | 100.00 | 97.76 | 95.26 | 49.50 | 99.99 | 99.99 | 100.00 | 100.00 |
| DDoS ICMP | 100.00 | 99.41 | 98.61 | 99.85 | 99.26 | 37.59 | 99.61 | 100.00 | 99.79 | 100.00 |
| SQL injection | 80.46 | 5.98 | 0.00 | 22.95 | 60.94 | 39.19 | 100.00 | 0.00 | 53.84 | 41.65 |
| Vulnerability sc. | 39.79 | 99.28 | 99.96 | 84.39 | 37.75 | 86.66 | 48.28 | 44.14 | 55.96 | 97.35 |
| DDoS TCP | 100.00 | 58.11 | 58.11 | 92.96 | 58.11 | 82.13 | 71.39 | 71.39 | 56.08 | 100.00 |
| Password | 20.77 | 99.48 | 19.62 | 51.12 | 30.59 | 54.53 | 43.00 | 36.45 | 49.94 | 40.12 |
| DDoS HTTP | 91.37 | 34.21 | 0.08 | 10.03 | 26.03 | 59.89 | 97.92 | 80.00 | 12.39 | 98.39 |
| Uploading | 28.89 | 23.28 | 75.32 | 27.40 | 41.33 | 90.37 | 88.08 | 24.89 | 89.25 | 59.71 |
| Backdoor | 94.73 | 93.64 | 48.03 | 93.31 | 92.56 | 98.59 | 98.76 | 99.96 | 99.82 | 83.12 |
| Port Scanning | 0.00 | 49.90 | 49.90 | 8.00 | 98.20 | 0.00 | 31.72 | 31.82 | 30.86 | 47.07 |
| XSS | 37.89 | 4.35 | 2.73 | 74.86 | 99.84 | 29.09 | 38.60 | 100.00 | 31.22 | 18.68 |
| Ransomware | 88.89 | 88.89 | 92.16 | 91.85 | 42.63 | 99.94 | 97.55 | 44.31 | 96.25 | 77.45 |
| Fingerprinting | 0.00 | 67.12 | 67.81 | 47.95 | 67.81 | 0.00 | 52.13 | 34.26 | 10.92 | 45.62 |
| MITM | 100.00 | 100.00 | 100.00 | 100.00 | 36.84 | 98.70 | 98.70 | 100.00 | 100.00 | 1.79 |

### C. DISCUSSION

The results in Table 11 demonstrate that data augmentation positively impacted the macro-average performance metrics for some classification algorithms. Table 12 demonstrates how data augmentation influenced each classifier's performance, providing a per-class perspective. It presents the F1-Score delta variation for each class, comparing the performance of models trained with the non-augmented dataset to the ones training using augmented datasets. Negative values specify the decrease in the F1-score if compared to the one achieved using the original data for training, and positive values identify the increase in the F1-score if using synthetically generated data for training.

Table 12 provides evidence that using data augmentation led to a drop in the F1-Score in most classes and practically no increase in performance for the XGBoost and DT classifiers. Therefore, the use of data augmentation proved unnecessary for these classifiers. Also, XGBoost achieved

**TABLE 10.** Performance of the XGBoost: Precision and Recall (%) Across Different Data Augmentation Techniques.

| | Precision | | | | | Recall | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *No Augmentation* | SMOTE | SMOTE-NC | REalTabFormer | GReaT | *No Augmentation* | SMOTE | SMOTE-NC | REalTabFormer | GReaT |
| Normal | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| DDoS UDP | 100.00 | 100.00 | 100.00 | 100.00 | 99.96 | 99.98 | 99.99 | 99.99 | 99.99 | 100.00 |
| DDoS ICMP | 99.60 | 99.68 | 99.27 | 99.47 | 99.69 | 99.76 | 99.98 | 99.18 | 100.00 | 99.91 |
| SQL injection | 83.41 | 76.91 | 76.81 | 81.61 | 79.45 | 86.22 | 96.15 | 96.54 | 81.97 | 88.86 |
| Vulnerability sc. | 95.88 | 95.59 | 95.64 | 95.41 | 95.96 | 99.64 | 99.41 | 99.49 | 99.30 | 99.15 |
| DDoS TCP | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| Password | 81.77 | 89.14 | 89.29 | 78.25 | 81.28 | 87.17 | 80.53 | 80.35 | 91.64 | 88.08 |
| DDoS HTTP | 96.92 | 95.51 | 95.40 | 94.71 | 91.30 | 92.74 | 94.44 | 94.59 | 70.89 | 93.51 |
| Uploading | 89.44 | 89.36 | 89.34 | 72.60 | 85.44 | 79.07 | 79.18 | 79.18 | 95.74 | 81.77 |
| Backdoor | 96.84 | 93.87 | 93.94 | 95.69 | 93.45 | 97.09 | 99.51 | 99.49 | 98.20 | 99.84 |
| Port Scanning | 100.00 | 97.17 | 97.10 | 99.75 | 100.00 | 96.42 | 96.57 | 96.52 | 95.71 | 93.47 |
| XSS | 85.60 | 90.25 | 90.68 | 91.63 | 92.36 | 87.21 | 82.80 | 82.47 | 80.21 | 74.21 |
| Ransomware | 91.33 | 92.47 | 92.47 | 91.33 | 89.36 | 95.91 | 95.24 | 95.34 | 96.65 | 99.42 |
| Fingerprinting | 45.21 | 78.08 | 72.60 | 73.97 | 59.59 | 56.90 | 23.46 | 19.92 | 41.06 | 10.51 |
| MITM | 98.68 | 98.68 | 98.68 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 7.76 |

**TABLE 11.** Comparative Analysis of Multiclass Classification Performance Using Macro Average (%).

| | | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Decision tree | *No augmentation* | 91.11 | 91.42 | 91.26 |
| | SMOTE | 87.37 | 91.18 | 88.19 |
| | SMOTE-NC | 87.36 | 91.16 | 88.16 |
| | REalTabFormer | 87.19 | 90.97 | 88.45 |
| | GReaT | 80.03 | 87.03 | 78.58 |
| Random forest | *No augmentation* | 86.15 | 77.60 | 81.03 |
| | SMOTE | 46.71 | 54.86 | 40.65 |
| | SMOTE-NC | 91.78 | 88.19 | 88.99 |
| | REalTabFormer | 91.38 | 89.44 | 90.13 |
| | GReaT | 89.65 | 80.93 | 80.71 |
| Tabnet | *No augmentation* | 59.47 | 61.75 | 55.79 |
| | SMOTE | 68.23 | 77.72 | 65.40 |
| | SMOTE-NC | 60.82 | 64.48 | 53.47 |
| | REalTabFormer | 66.83 | 65.75 | 62.44 |
| | GReaT | 65.80 | 67.40 | 59.66 |
| XGBoost | *No augmentation* | 90.98 | 91.87 | 91.34 |
| | SMOTE | 93.11 | 89.82 | 90.36 |
| | SMOTE-NC | 92.75 | 89.54 | 90.01 |
| | REalTabFormer | 91.63 | 90.09 | 90.25 |
| | GReaT | 91.19 | 82.43 | 82.90 |
| MLP | *No augmentation* | 85.66 | 77.11 | 77.95 |
| | SMOTE | 87.37 | 91.18 | 88.19 |
| | SMOTE-NC | 77.48 | 80.99 | 75.98 |
| | REalTabFormer | 79.40 | 78.78 | 76.61 |
| | GReaT | 74.14 | 79.69 | 71.58 |

the best performance, which outreached previous work, as represented in Table 13. Still, augmentation led to some performance improvements in other classification algorithms

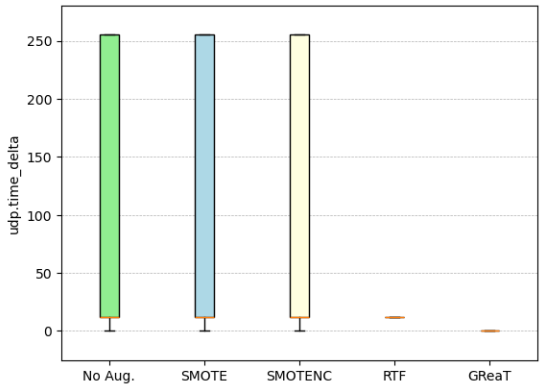



**FIGURE 8.** Boxplot illustrating the distribution of the *udp.time.delta* numerical feature across augmented datasets for the MITM class. Outliers have been omitted to enhance visualization clarity.

rather than XGBoost and DT. Hence, we analyzed the values generated by the data augmentation algorithms to identify the causes that made them lead to distinct impacts on performance.

Figures 8, 9 and 10 present the ranges of values generated by the different algorithms for two numerical features in the MITM attack class, which was one of the classes where the classifiers had the worst results. Concerning the features depicted in Figures 8 and 9, the SMOTE and SMOTE-NC algorithms generated data distributions consistent with the original dataset (as expected due to its statistical nature), and GPT-based algorithms (i.e., RealTabFormer and GReaT) generated nearly all data instances with a single value. For the feature represented in Figure 10, RealTabFormer generated a fair amount of data instances whose values are outside the feature's range in the original dataset.

**TABLE 12.** Delta in F1 Scores (%) for different classes and data augmentation techniques.

| | Decision tree | | | | Random Forest | | | | Tabnet | | | | XGBoost | | | | MLP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SMOTE | SMOTE-NC | REalTabFormer | GReaT | SMOTE | SMOTE-NC | REalTabFormer | GReaT | SMOTE | SMOTE-NC | REalTabFormer | GReaT | SMOTE | SMOTE-NC | REalTabFormer | GReaT | SMOTE | SMOTE-NC | REalTabFormer | GReaT |
| Normal | 0 | 0 | 0 | 0 | -12 | 5 | 5 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DDoS (UDP) | 0 | 0 | 0 | -4 | 0 | 0 | 0 | -4 | 84 | 84 | 83 | 82 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DDoS (ICMP) | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 45 | 45 | 45 | 45 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SQL injection | 0 | 0 | -1 | -3 | -20 | 28 | 26 | 28 | -41 | -52 | -21 | -3 | 1 | 1 | -3 | -1 | 23 | -11 | -8 | -21 |
| Vulnerability scanner | 0 | 0 | -1 | -1 | -74 | 3 | 2 | 3 | 10 | 7 | 13 | 0 | 0 | 0 | 0 | 0 | 5 | -1 | 0 | 0 |
| DDoS (TCP) | 0 | 0 | 0 | -5 | -42 | 0 | 0 | -1 | -26 | -26 | -20 | -17 | 0 | 0 | 0 | 0 | 10 | -17 | -13 | -17 |
| Password | 0 | 0 | 0 | -4 | -53 | 25 | 25 | 24 | 30 | -5 | 20 | 5 | 0 | 0 | 0 | 0 | 53 | 26 | 20 | 29 |
| DDoS (HTTP) | -3 | -3 | -11 | -5 | -61 | 29 | 23 | 28 | -22 | -72 | -61 | -31 | 0 | 0 | -14 | -2 | 7 | -10 | -22 | -15 |
| Uploading | 1 | 0 | 1 | -2 | -31 | 26 | 27 | 26 | -7 | -6 | -2 | 5 | 0 | 0 | -1 | 0 | 24 | 0 | -11 | -1 |
| Backdoor | 0 | 0 | 0 | -2 | -92 | 4 | 4 | 4 | 0 | -32 | 0 | -9 | 0 | 0 | 0 | 0 | -1 | -1 | 0 | -1 |
| Port scanning | -3 | -3 | 0 | -3 | -89 | 5 | 7 | 6 | 39 | 39 | 13 | 64 | -1 | -1 | 0 | -2 | 26 | -2 | -1 | -2 |
| XSS | -6 | -6 | -6 | -5 | -47 | 13 | 15 | 14 | -25 | -27 | 11 | -1 | 0 | 0 | -1 | -4 | 32 | 2 | 4 | 3 |
| Ransomware | -2 | -2 | 0 | 0 | -87 | 6 | 6 | 7 | -1 | -34 | 0 | -39 | 0 | 0 | 0 | 1 | -3 | 0 | 0 | 0 |
| Fingerprinting | -32 | -32 | -19 | -60 | 1 | -27 | -4 | -52 | 59 | 46 | 18 | 55 | -14 | -19 | 2 | -33 | -21 | -15 | 12 | -4 |
| MITM | 0 | 0 | -6 | -96 | 0 | 1 | 0 | -94 | 0 | 1 | 1 | -96 | 0 | 0 | 1 | -85 | 0 | 0 | 0 | -68 |

**TABLE 13.** Comparison of F1-scores (%) with previous work.

| Class | XGBoost | Results from [12] |
|---|---|---|
| Normal | 100 | 100 |
| DDoS (UDP) | 100 | 100 |
| DDoS (ICMP) | 100 | 100 |
| SQL injection | 85 | 57 |
| Vulnerability scanner | 98 | 90 |
| DDoS (TCP) | 100 | 90 |
| Password | 85 | 45 |
| DDoS (HTTP) | 95 | 83 |
| Uploading | 85 | 90 |
| Backdoor | 97 | 90 |
| Port scanning | 98 | 66 |
| XSS | 86 | 43 |
| Ransomware | 94 | 79 |
| Fingerprinting | 50 | 61 |
| MITM | 99 | 100 |

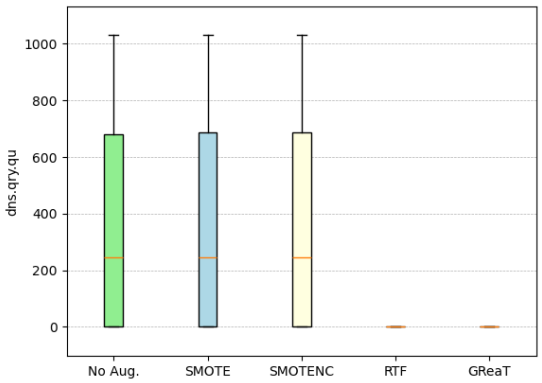



**FIGURE 9.** Boxplot depicting the distribution of the *dns.qry.qu* numerical feature within the MITM class. Outliers have been excluded for improved visualization.

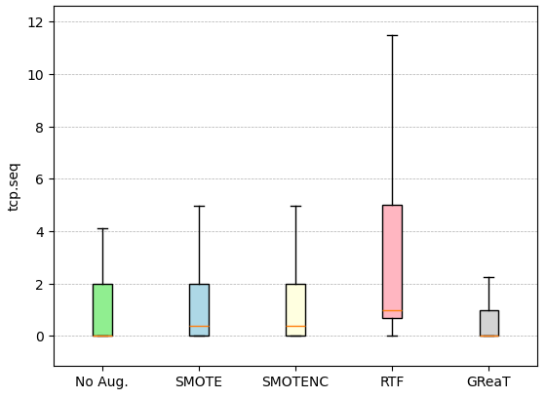



**FIGURE 10.** Boxplot illustrating the distribution of the *tcp.seq* numerical feature for the MITM class. Outliers have been excluded to enhance clarity in visualization.

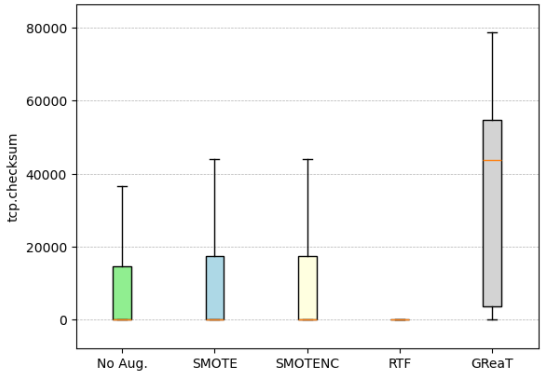



**FIGURE 11.** Boxplot illustrating the distribution of the *tcp.checksum* numerical feature for the FingerPrinting class. Outliers have been omitted to optimize visual clarity.

The GPT-based algorithms demonstrated distinct behaviour for some classes and features. For instance, Figure 11 presents the ranges of values generated for a numerical feature in the FingerPrinting attack class. For such a feature and class, RealTabFormer generated almost one value for all

**TABLE 14.** Fingerprinting attack distribution (percentage) of categorical features.

| | http.request.method | | | http.request.version | | http.request.version | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Categories | 0.0 | 0 | GET | 0.0 | 0 | 0.0 | 0 | 1.0 | HTTP/1.1 | HTTP/1.0 |
| Non Augmentation | 100.00 | 0.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| SMOTE | 100.00 | 0.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| SMOTE-NC | 100.00 | 0.00 | 0.00 | 100.00 | 0.00 | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| REatTabFormer | 99.97 | 0.03 | 0.00 | 99.94 | 0.06 | 99.98 | 0.00 | 0.03 | 0.00 | 0.00 |
| GReaT | 35.89 | 61.31 | 2.72 | 96.77 | 3.23 | 35.88 | 61.31 | 0.00 | 2.68 | 0.12 |

**TABLE 15.** MITM attack distribution (percentage) of categorical features.

| | http.request.method | | | http.request.version | | | | dns.qry.name.len | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Categories | 0.0 | 0 | GET | 0.0 | 0 | 1.0 | HTTP/1.1 | 0.0 | 0 | 1.0 |
| Non Augmentation | 100.00 | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 0.00 | 34.75 | 0.00 | 65.25 |
| SMOTE | 100.00 | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 0.00 | 34.89 | 0.49 | 64.61 |
| SMOTE-NC | 100.00 | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 0.00 | 35.53 | 0.00 | 64.47 |
| REatTabFormer | 99.68 | 0.31 | 0.01 | 99.77 | 0.15 | 0.00 | 0.08 | 91.88 | 0.00 | 8.13 |
| GReaT | 53.15 | 45.72 | 1.12 | 52.84 | 46.03 | 0.00 | 1.14 | 99.81 | 0.00 | 0.19 |

instances, while GReaT generated values that didn't exist for that feature and class in the original data. Considering the average F1-score among all classifiers for such class, the data generated by RealTabFormer (single value) was more beneficial than the out-of-range values generated by GReaT.

Indeed, the difference between the performance provided by RealTabFormer and GReaT is evident in the results presented in Table 11. We evaluated five classifiers and five data augmentation configurations for each classifier. RealTabFormer led to the best F1-Score using RF, the second-best performance in two classifiers (DT and TabNet), and the third-best F1-Score in the other two classifiers (XGBoosT and MLP). GReaT led to the worst performance in three classifiers (DT, XGBoost, and MLP) and didn't lead to top performance in any classifier.

GPT-based methods also generated inconsistent values for categorical features in several classes. Tables and present, for the Fingerprinting and MITM classes, respectively, the percentage of instances for each value in categorical features, considering the original data and also the data generation algorithm. It is possible to verify, for example, that for the features *http.request.method* and *http.request.version*, the GPT-based algorithms generated data with two different representations for the same category (e.g., '0' and '0.0'), which can negatively affect the performance of classifiers. Also, the distribution of values across categories in the *dns.qry.name.len* feature follows the opposite proportion of that on the original data. Furthermore, the GReaT method generated data with values like 'GET' (for *http.request.method*) and 'HTTP/1.1' and 'HTTP/1.0' (for *http.request.version*), which are non-existent values of the features for such classes in the original data.

Although the generation of invalid data is not exclusive to GPT-based algorithms (SMOTE, for example, does not adequately handle categorical variables and generates intermediate values for categories that have values "1" and "0"), the type and amount of inconsistencies GReaT generates contributed to the low performance of classifiers trained with data generated by this algorithm.

Indeed, a high number of features can pose challenges for data augmentation in generating meaningful new data, and data augmentation is not always a guaranteed solution. Generating additional samples may not necessarily lead to a substantial improvement in class detection. Therefore, it is crucial to thoroughly evaluate the potential benefits and drawbacks of chosen data augmentation technique in the specific context before implementation.

## V. CONCLUSION AND FUTURE WORK

In this work, we assessed the effectiveness of GPT and interpolation-based data augmentation techniques when generating IIoT network traffic. We used four techniques from the literature (i.e., SMOTE, SMOTE-NC, REalTabFormer, and GReaT) to create synthetic data representing normal and malicious operations. Then, we trained classification models using the non-augmented and several augmented datasets and compared the multiclass classification performance in 25 scenarios.

The analysis of data generated by GPT-based methods (i.e., REalTabFormer and GReaT) showed these methods failed to capture class-specific feature values. Indeed, some data generated by such models are valid network traffic but not for the class (i.e., attack type) they were assigned to. Such data reduced the class-specific performance of intrusion detection algorithms trained with data generated by REalTabFormer and, especially, by GReaT. Indeed, SMOTE also generated some invalid values (especially for categorical features), but such values are statistically close to real ones and still improve the performance of some classifiers. For instance, the F1-score of the MLP trained with data augmented by SMOTE increased in 8 classes and decreased in 3 classes if compared to the model's performance when trained with

the non-augmented dataset. On the other hand, the F1-score of the MLP trained with data created by GReaT decreased in 8 classes and increased in 2 classes if compared to the model's performance when trained with the non-augmented dataset.

Furthermore, the evaluation showed that, while some intrusion detection solutions benefit more from data augmentation (e.g., Random Forest) and others had uncertain behaviour with performance gains and losses depending on the class evaluated (e.g., Tabnet), XGBoost was practically indifferent to the use of data augmentation.

XGBoost was the classification that achieved the best results for intrusion detection. The macro-average classification recall and F1-score of XGBoost were higher when trained with non-augmented data than with augmented data. Precision was the only performance metric for XGBoost that increased (approximately 3%) when training the model with a dataset with synthetic data, which indicates that depending on the used intrusion detection solution and performance objectives, the effort of applying data augmentation techniques may be worthless.

As future work, we intend to evaluate the use of GANs for data augmentation in the context of intrusion detection in IIoT. We also plan to assess the impact of data augmentation on classification algorithms based on other IIoT network traffic datasets.

## REFERENCES

[1] J. Leber. (Apr. 2020). *General Electric Pitches an Industrial Internet*. [Online]. Available: https://www.technologyreview.com/2012/11/28/114725/general-electric-pitches-an-industrial-internet/

[2] R. Ali Laghari and S. Mekid, "Comprehensive approach toward IIoT based condition monitoring of machining processes," *Measurement*, vol. 217, Aug. 2023, Art. no. 113004, doi: 10.1016/j.measurement.2023.113004.

[3] E. Fiestas, P. Linares, and S. Prado, "Integration of an IIoT platform with a deep learning based computer vision system for seedling quality control automation," in *Proc. IEEE 3rd Eurasia Conf. IoT, Commun. Eng. (ECICE)*, Oct. 2021, pp. 621–626.

[4] J. Rosales, S. Deshpande, and S. Anand, "IIoT based augmented reality for factory data collection and visualization," *Proc. Manuf.*, vol. 53, pp. 618–627, Jan. 2021.

[5] F. J. Valente and A. C. Neto, "Intelligent steel inventory tracking with IoT/RFID," in *Proc. IEEE Int. Conf. RFID Technol. Appl. (RFID-TA)*, 2017, pp. 158–163, doi: 10.1109/RFID-TA.2017.8098639.

[6] K. S. H. Ong, W. Wang, N. Q. Hieu, D. Niyato, and T. Friedrichs, "Predictive maintenance model for IIoT-based manufacturing: A transferable deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15725–15741, Sep. 2022.

[7] P. Jayalaxmi, R. Saha, G. Kumar, N. Kumar, and T.-H. Kim, "A taxonomy of security issues in industrial Internet-of-Things: Scoping review for existing solutions, future implications, and research challenges," *IEEE Access*, vol. 9, pp. 25344–25359, 2021, doi: 10.1109/ACCESS.2021.3057766.

[8] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6822–6834, Aug. 2019, doi: 10.1109/JIOT.2019.2912022.

[9] H. Huang, P. Ye, M. Hu, and J. Wu, "A multi-point collaborative DDoS defense mechanism for IIoT environment," *Digit. Commun. Netw.*, vol. 9, no. 2, pp. 590–601, Apr. 2023, doi: 10.1016/j.dcan.2022.04.008.

[10] S. Ma, H. Wang, L. Zhu, and Q. Zhang, "Joint security and resilience control in IIoT-based virtual control train sets under jamming attacks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 9, pp. 11196–11212, Sep. 2023, doi: 10.1109/TVT.2023.3266561.

[11] M. Nuaimi, L. C. Fourati, and B. B. Hamed, "Intelligent approaches toward intrusion detection systems for industrial Internet of Things: A systematic comprehensive review," *J. Netw. Comput. Appl.*, vol. 215, Jun. 2023, Art. no. 103637, doi: 10.1016/j.jnca.2023.103637.

[12] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40281–40306, 2022, doi: 10.1109/ACCESS.2022.3165809.

[13] M. N. Halgamuge, "Estimation of the success probability of a malicious attacker on blockchain-based edge network," *Comput. Netw.*, vol. 219, Dec. 2022, Art. no. 109402, doi: 10.1016/j.comnet.2022.109402. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622004364

[14] S. Latif, Z. Idrees, Z. E. Huma, and J. Ahmad, "Blockchain technology for the industrial Internet of Things: A comprehensive survey on security challenges, architectures, applications, and future research directions," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 11, p. e4337, Nov. 2021, doi: 10.1002/ett.4337.

[15] N. Prazeres, R. L. D. C. Costa, L. Santos, and C. Rabadão, "Engineering the application of machine learning in an IDS based on IoT traffic flow," *Intell. Syst. Appl.*, vol. 17, Feb. 2023, Art. no. 200189.

[16] N. Prazeres, R. L. D. C. Costa, L. Santos, and C. Rabadão, "Evaluation of ai-based malware detection in IoT network traffic," in *Proc. 19th Int. Conf. Secur. Cryptogr.*, vol. 1, 2022, pp. 580–585.

[17] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6822–6834, 2019, doi: 10.1109/JIOT.2019.2912022.

[18] K. DeMedeiros, A. Hendawi, and M. Alvarez, "A survey of AI-based anomaly detection in IoT and sensor networks," *Sensors*, vol. 23, no. 3, p. 1352, Jan. 2023.

[19] S. De, M. Bermudez-Edo, H. Xu, and Z. Cai, "Deep generative models in the industrial Internet of Things: A survey," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 5728–5737, Sep. 2022, doi: 10.1109/TII.2022.3155656.

[20] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.

[21] M. Zolanvari, M. A. Teixeira, and R. Jain, "Effect of imbalanced datasets on security of industrial IoT using machine learning," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Nov. 2018, pp. 112–117, doi: 10.1109/ISI.2018.8587389.

[22] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, Nov. 2022, Art. no. 100258, doi: 10.1016/j.array.2022.100258.

[23] K. Alomar, H. I. Aysel, and X. Cai, "Data augmentation in classification and segmentation: A survey and new strategies," *J. Imag.*, vol. 9, no. 2, p. 46, Feb. 2023, doi: 10.3390/jimaging9020046.

[24] Z. Yang, R. O. Sinnott, J. Bailey, and Q. Ke, "A survey of automated data augmentation algorithms for deep learning-based image classification tasks," 2022, *arXiv:2206.06544*.

[25] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, "A survey of data augmentation approaches for NLP," 2021, *arXiv:2105.03075*.

[26] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, Dec. 2019, doi: 10.1186/s42400-019-0038-7.

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[28] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *Proc. VLDB Endowment*, vol. 11, no. 10, pp. 1071–1083, Jun. 2018, doi: 10.14778/3231751.3231757.

[29] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[30] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein, "SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training," 2021, *arXiv:2106.01342*.

[31] A. V. Solatorio and O. Dupriez, "REaLTabFormer: Generating realistic relational and tabular data using transformers," 2023, *arXiv:2302.02041*.

[32] V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci, "Language models are realistic tabular data generators," 2022, *arXiv:2210.06280*.

[33] T. Chen and C. Guestrin, "XGBoost," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[34] S. O. Arik and T. Pfister, "TabNet: Attentive interpretable tabular learning," 2019, *arXiv:1908.07442*.

[35] D. M. F. da Silva and J. M. B. Frade, "Cibersegurança para a indústria 4.0," Bachelor's Thesis, ESTG, Polytech. Leiria, Leiria, Portugal, 2022.

[36] E. Vanpoucke, A. Vereecke, and S. Muylle, "Leveraging the impact of supply chain integration through information technology," *Int. J. Oper. Prod. Manag.*, vol. 37, no. 4, pp. 510–530, Apr. 2017.

[37] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on SCADA systems," in *Proc. Int. Conf. Internet Things, 4th Int. Conf. Cyber, Phys. Social Comput.*, 2011, pp. 380–388, doi: 10.1109/iThings/CPSCom.2011.34.

[38] P. Aitken, B. Claise, and B. Trammell, *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*, document RFC 7011, RFC Editor, Request for Comments, Sep. 2013, p. 76. [Online]. Available: https://www.rfc-editor.org/info/rfc7011, doi: 10.17487/RFC7011.

[39] B. Andreas, J. Dilruksha, and E. McCandless, "Flow-based and packet-based intrusion detection using BLSTM," *SMU Data Sci. Rev.*, vol. 3, no. 3, p. 8, 2020.

[40] T.-T.-H. Le, Y. E. Oktian, and H. Kim, "XGBoost for imbalanced multiclass classification-based industrial Internet of Things intrusion detection systems," *Sustainability*, vol. 14, no. 14, p. 8707, Jul. 2022, doi: 10.3390/su14148707.

[41] I. P. Turnipseed, "A new SCADA dataset for intrusion detection system research," Dept. Elect. Comput. Eng., Mississippi State Univ., MS, USA, 2015. [Online]. Available: https://scholarsjunction.msstate.edu/td/209/

[42] M. Teixeira, T. Salman, M. Zolanvari, R. Jain, N. Meskin, and M. Samaka, "SCADA system testbed for cybersecurity research using machine learning approach," *Future Internet*, vol. 10, no. 8, p. 76, Aug. 2018, doi: 10.3390/fi10080076.

[43] Á. L. P. Gómez, L. F. Maimó, A. H. Celdrán, F. J. G. Clemente, C. C. Sarmiento, C. J. D. C. Masa, and R. M. Nistal, "On the generation of anomaly detection datasets in industrial control systems," *IEEE Access*, vol. 7, pp. 177460–177473, 2019, doi: 10.1109/ACCESS.2019.2958284.

[44] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165130–165150, 2020, doi: 10.1109/ACCESS.2020.3022862.

[45] M. Zolanvari, Z. Yang, K. Khan, R. Jain, and N. Meskin, "TRUST XAI: Model-agnostic explanations for AI with a case study on IIoT security," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 2967–2978, Feb. 2023, doi: 10.1109/JIOT.2021.3122019.

[46] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, "X-IIoTID: A connectivity-agnostic and device-agnostic intrusion data set for industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3962–3977, Mar. 2022, doi: 10.1109/JIOT.2021.3102056.

[47] M. Bayer, M.-A. Kaufhold, and C. Reuter, "A survey on data augmentation for text classification," *ACM Comput. Surveys*, vol. 55, no. 7, pp. 1–39, Jul. 2023, doi: 10.1145/3544558.

[48] L. Li and M. Spratling, "Data augmentation alone can improve adversarial training," 2023, *arXiv:2301.09879*.

[49] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[50] Z. Zhong, "Random erasing data augmentation," in *Proc. AAAI*, vol. 34, 2020, pp. 13001–13008.

[51] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," 2017, *arXiv:1711.04340*.

[52] J. Lemley, S. Bazrafkan, and P. Corcoran, "Smart augmentation learning an optimal data augmentation strategy," *IEEE Access*, vol. 5, pp. 5858–5869, 2017, doi: 10.1109/ACCESS.2017.2696121.

[53] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," in *Proc. NIPS*, Vancouver, BC, Canada. Red Hook, NY, USA: Curran, 2020, Art. no. 525.

[54] J. Fonseca and F. Bacao, "Research trends and applications of data augmentation algorithms," 2022, *arXiv:2207.08817*.

[55] Q. Zheng, P. Zhao, Y. Li, H. Wang, and Y. Yang, "Spectrum interference-based two-level data augmentation method in deep learning for automatic modulation classification," *Neural Comput. Appl.*, vol. 33, no. 13, pp. 7723–7745, Jul. 2021.

[56] B. Li, J. Springer, G. Bebis, and M. Hadi Gunes, "A survey of network flow applications," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 567–581, Mar. 2013, doi: 10.1016/j.jnca.2012.12.020.

[57] M. Nakhwan and R. Duangsoithong, "Comparison analysis of data augmentation using bootstrap, GANs and autoencoder," in *Proc. 14th Int. Conf. Knowl. Smart Technol. (KST)*, 2022, pp. 18–23, doi: 10.1109/KST53302.2022.9729065.

[58] K. Gupta, D. K. Sharma, K. D. Gupta, and A. Kumar, "A tree classifier based network intrusion detection model for Internet of Medical Things," *Comput. Electr. Eng.*, vol. 102, Sep. 2022, Art. no. 108158, doi: 10.1016/j.compeleceng.2022.108158.

[59] H. Jmila and M. I. Khedher, "Adversarial machine learning for network intrusion detection: A comparative study," *Comput. Netw.*, vol. 214, Sep. 2022, Art. no. 109073, doi: 10.1016/j.comnet.2022.109073.

[60] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6 doi: 10.1109/CISDA.2009.5356528.

[61] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. MilCIS*, Nov. 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[62] J. Liu, Y. Gao, and F. Hu, "A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM," *Comput. Secur.*, vol. 106, Jul. 2021, Art. no. 102289.

[63] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jun. 2008, pp. 1322–1328, doi: 10.1109/IJCNN.2008.4633969.

[64] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–9.

[65] I. Sharafaldin, A. H. Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISS*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[66] Y. Zhang and Q. Liu, "On IoT intrusion detection based on data augmentation for enhancing learning on unbalanced samples," *Future Gener. Comput. Syst.*, vol. 133, pp. 213–227, Aug. 2022.

[67] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intell. Comput.*, Aug. 2005, pp. 878–887.

[68] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[69] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014, *arXiv:1406.2661*.

[70] R. Sauber-Cole and T. M. Khoshgoftaar, "The use of generative adversarial networks to alleviate class imbalance in tabular data: A survey," *J. Big Data*, vol. 9, no. 1, p. 9, Aug. 2022, doi: 10.1186/s40537-022-00648-6.

[71] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.

[72] Z. Li, S. Chen, H. Dai, D. Xu, C.-K. Chu, and B. Xiao, "Abnormal traffic detection: Traffic feature extraction and DAE-GAN with efficient data augmentation," *IEEE Trans. Rel.*, vol. 72, no. 2, pp. 498–510, Jun. 2023, doi: 10.1109/TR.2022.3204349.

[73] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018, *arXiv:1802.09089*.

[74] A. A. Alqarni and E.-S. M. El-Alfy, "Improving intrusion detection for imbalanced network traffic using generative deep learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 4, pp. 959–967, 2022, doi: 10.14569/IJACSA.2022.01304109.

[75] Z. Wang, Z. Li, J. Wang, and D. Li, "Network intrusion detection model based on improved BYOL self-supervised learning," *Secur. Commun. Netw.*, vol. 2021, pp. 1–23, Oct. 2021.

[76] R. A. Fisher and F. Yates, *Statistical Tables for Biological, Agricultural and Medical Research*. New York, NY, USA: Hafner, 1953.

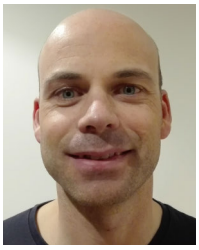
[77] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. Guo, and M. G. Azar, "Bootstrap your own latent—A new approach to self-supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 21271–21284.

[78] A. Verma and V. Ranga, "Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning," *Proc. Comput. Sci.*, vol. 125, pp. 709–716, Jan. 2018, doi: 10.1016/j.procs.2017.12.091.
[79] V. Kumar, A. Choudhary, and E. Cho, "Data augmentation using pre-trained transformer models," 2020, *arXiv:2003.02245*.
[80] M. Glass, X. Wu, A. Rajaram Naik, G. Rossiello, and A. Gliozzo, "Retrieval-based transformer for table augmentation," 2023, *arXiv:2306.11843*.
[81] *CIIC-C-T-Polytechnic-of-Leiria/Dataaugmentationtests*, Accessed: Jun. 12, 2023. [Online]. Available: https://github.com/CIIC-C-T-Polytechnic-of-Leiria/dataAugmentationTests
[82] *Edge-IIoTset Cyber Security Dataset of IoT & IIoT | Kaggle*. Accessed: Jun. 1, 2023. [Online]. Available: https://www.kaggle.com/datasets/mohamedamineferrag/edgeiiotset-cyber-security-dataset-of-iot-iiot

**FRANCISCO S. MELÍCIAS** received the B.S. degree in computer science, in 2022. He is currently pursuing the master's degree in computer engineering-mobile computing. He is a Passionate Computer Scientist and a Researcher with IPLeiria, Leiria, Portugal. His research interests include the field of machine learning in cybersecurity, with a focus on exploring and contributing to advancements in the domain.

**TIAGO F. R. RIBEIRO** received the bachelor's degree in electrical engineering and computers and the master's degree in data science from the School of Technology and Management (ESTG), Polytechnic of Leiria. Currently, he is a Researcher affiliated with the Computer Science and Communication Research Centre (CIIC), with a focus on machine learning, image processing, segmentation, and the application of these techniques to fields, such as earth sciences and cybersecurity, with a special interest in generative models.

**CARLOS RABADÃO** received the B.Sc. degree in electrical engineering (specialization in telecommunications and electronics) from the University of Coimbra, Portugal, in 1989, the M.Sc. degree in electronics and telecommunications engineering from the University of Aveiro, Portugal, in 1996, and the Ph.D. degree in computer science engineering from the University of Coimbra, in 2007. He is currently a Coordinator Professor with the Department of Computer Science Engineering, School of Technology and Management (ESTG), Polytechnic of Leiria. He is the Head of the Computer Science and Communication Research Centre (CIIC), Polytechnic of Leiria, and the Chair of the Technical-Scientific Council, ESTG. He has more than 24 years of teaching and research experience in computer engineering, namely in the areas of cybersecurity and computer networks. He has published around 50 papers in international conferences and journals in the areas of cybersecurity, computer science, and data communications. He has participated in more than 20 national and international research and development projects, having coordinated five of these. His major research interests include information and network security, information security and privacy management, security incident response systems for industry 4.0, next-generation networks, and services and wireless networks.



**LEONEL SANTOS** received the B.Sc. degree in computer engineering (specialization in communication networks and systems) from the School of Technology and Management (ESTG), Polytechnic of Leiria, Portugal, in 2006, and the Ph.D. degree in computer science from Universidade de Trás-os-Montes e Alto Douro, Portugal, in 2020. He is currently an Assistant Professor with the Department of Computer Science Engineering, ESTG, Polytechnic of Leiria. He is a Full Member with the Computer Science and Communication Research Centre (CIIC), Polytechnic of Leiria, a Forensic Computer Expert with the Laboratory of Cybersecurity and Computer Forensics-LabCIF, Polytechnic of Leiria, and a member with the Scientific-Pedagogical Committee. He has more than 14 years of teaching and research experience in computer engineering, namely in the areas of cybersecurity and computer networks. He has published papers in journals and international conferences in the areas of cybersecurity, information and network security, and computer science. He has participated in national research and development projects. His major research interests include cybersecurity, information security, network security, the Internet of Things, intrusion detection systems, and computer forensics.

**ROGÉRIO LUÍS DE C. COSTA** received the Ph.D. degree in computer engineering from the University of Coimbra (UC), Portugal, in 2011, and the M.Sc. degree in informatics from Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Brazil, in 2002. He is currently a Researcher with the Polytechnic of Leiria, Portugal. He has more than 15 years of teaching experience. He participated in national and international research projects and published papers at leading journals and conferences. He also held technical and managerial positions in software development companies. His research interests include big data, machine learning, cybersecurity and privacy, and data integration and quality.