MATHEMATICAL MODELING AND COMPUTING, Vol. 11, No. 4, pp. 930–945 (2024)

XIDINTV: XGBoost-based intrusion detection of imbalance network traffic via variational auto-encoder

Abdulganiyu O. H.[1], Ait Tchaoucht T.[1], Ezziyyani M.[2], Benslimane M.[3]

[1] *Euromed University of Fes, UEMF, Morocco*
[2] *Mathematical Laboratory and Applications, Abdelmalek Essaadi University Faculty of Science and Technology, Tangier, Morocco*
[3] *Laboratory of Sciences, Engineering and Management, Sidi Mohamed Ben Abdellah University, Morocco*

(Received 8 January 2024; Revised 15 August 2024; Accepted 17 August 2024)

In networks characterized by imbalanced traffic, detecting malicious cyber-attacks poses a significant challenge due to their ability to blend seamlessly with regular data volumes. This creates a formidable hurdle for Network Intrusion Detection Systems (NIDS) striving for accurate and timely identification. The imbalance in normal and attack data, coupled with the diversity among attack categories, complicates intrusion detection. This research proposes a novel approach to address this issue by combining Extreme Gradient Boosting with variational autoencoder (XIDINTV). The methodology focuses on rectifying class imbalance by generating diverse rare-class attack data while maintaining similarities with the original samples. This enhances the classifier's ability to discern differences during training, improving classification performance. Evaluations on NSL-KDD and CSE-CIC-IDS2018 datasets demonstrate the effectiveness of XIDINTV, particularly when compared to SMOTE sampling technique and traditional classification models, with Xtreme Gradient Boosting excelling in detecting rare instances of attack traffic.

Keywords: *intrusion detection system; imbalance network traffic; extreme gradient boosting; variational autoencoder; anomaly detection; data augumentation.*

2010 MSC: 68T01, 68M10, 68U01 **DOI:** 10.23939/mmc2024.04.930

1. Introduction

The global proliferation of internet usage has precipitated a substantial increase in the transmission of vital, sensitive, and confidential personal and business data across networks. In response to this surge, threat actors have made concerted efforts to exploit vulnerabilities in network security measures, seeking unauthorized access to sensitive data, thereby potentially disrupting system functionality and compromising the confidentiality and availability of data [1]. Consequently, the field of cybersecurity has risen to paramount importance, offering essential mechanisms to combat cyberattacks and mitigate their associated costs and damages [2]. In this con-text, the Intrusion Detection System (IDS) has emerged as a fundamental component of network security architecture, serving as a tool for the detection of diverse forms of intrusions. IDS operates by scrutinizing network traffic, identifying suspicious and malicious activities, as well as violations of security policies. This functionality empowers network administrators to maintain vigilant surveillance over contemporary threats [3, 4].

Notably, IDS can be categorized into host-based and network-based variants, contingent on their deployment environment and placement [5, 6]. Host-based IDS is installed on individual hosts or devices, where it exclusively inspects data packets for signs of suspicious actions and security policy breaches. However, this approach is encumbered by the requisite installation of an IDS on each host to be protected, potentially resulting in heightened processing demands for each host, ultimately impacting performance negatively. In contrast, Network-based IDS is deployed across the entire computer network, functioning to monitor, capture, and analyze network traffic with the goal of detecting ma-

This work was supported by EuroMed University of Fes.

licious activities. IDSs employ two primary approaches for attack detection, namely anomaly-based and signature-based methods [7, 8]. In signature-based IDS, also known as "misuse intrusion detection" or "knowledge-based intrusion detection", the system operates reactively by seeking patterns and signatures corresponding to known attacks, which are maintained in a database. These patterns are subsequently compared with network data, and any matching activity is flagged as malicious. This approach exhibits ease of development and is adept at identifying known attacks with minimal false positives. However, it lacks the capability to detect novel or zero-day attacks since it relies exclusive-ly on its existing database of known attacks [9]. Furthermore, the signature-based approach consumes significant resources due to the maintenance, updates, and constant comparison of an extensive signature database with incoming data packets for potential attacks. In contrast, anomaly-based intrusion detection systems, often referred to as "behavior-based IDS", model normal system behavior, subsequently identifying attacks by detecting deviations from this established normal pattern [10]. The strength of this approach lies in its capacity to detect new, previously unknown attacks. However, it frequently results in a higher rate of false positives, limiting its practical application.

In the real world of cybersecurity, typical activities make up the majority, resulting in a prevalence of normal traffic data, while malicious cyberattacks represent only a small portion, leading to a significant category imbalance. This severe imbalance, coupled with redundant network traffic data, places immense pressure on intrusion detection. Malicious cyberattacks have the capacity to effectively blend into the extensive volume of regular network traffic, posing a significant challenge for machine learning algorithms in comprehending the distribution of minority categories and increasing the likelihood of misclassification. This issue of data imbalance directly impacts the performance of intrusion detection classifiers. For example, in the NSL-KDD dataset, which consists of 125 973 training data points, 67 343 are categorized as normal data, while 58 630 are categorized as attack data. This attack data can be further subdivided into four primary categories, with the R2L and U2R attack making the rarest category, occurring only 52 times. Cyberattacks with R2L (Remote-to-Local) and U2R (User-to-Root) characteristics are frequently linked to computer security and network systems. In a remote-to-local (R2L) attack, an unauthorized person tries to access a local system or network, while an attacker trying to gain local access to a system tries to escalate their privileges in order to obtain root or administrative access in a U2R attack. Typically, rare-class attack data constitutes a minuscule fraction compared to normal data. When employing a single ma-chine learning algorithm for classification, the classifier tends to favor the majority class, leading to misclassification of rare-class attack data as normal. To mitigate this challenge, common solutions involve the utilization of oversampling techniques such as Synthetic Minority Over-sampling Technique (SMOTE), Borderline-SMOTE, Adaptive Synthetic Sampling (ADASYN), Generative Adversarial Net-works (GAN), among others. These techniques aim to generate additional instances of rare-class attack data. However, traditional oversampling methods, while partial-ly addressing the imbalance issue, tend to produce attack samples that lack diversity and often result in the generation of low-quality data.

Numerous endeavors have been made by researchers to devise effective Intrusion Detection System (IDS) techniques for the efficient detection of cyberattacks. However, despite the extensive body of research in the field of IDS, the prevailing IDS methods continue to exhibit notable shortcomings, including a pronounced issue of high false alerts [11–15], limited detection rates [15–18] particularly concerning minority attack classes, and the persistent challenge of imbalanced data classes [13, 15, 19–21]. It therefore becomes imperative to have an approach that can provide promising security for Zero-day/novel types of attacks, and also mitigate the aforementioned limitations of the existing approaches, as well as serve as a good candidate to be implemented as a product. Therefore, this manuscript seeks to introduce machine learning algorithms based on Intrusion Detection Systems (IDS) for the efficient identification of minority network attacks. The primary contributions of this study can be summarized as follows:

1. We have devised the Minimum-Maximum (Min-Max) normalization meth-od to guarantee uniform scaling of all feature values.

2. We have designed a variational auto-encoder to counteract the issue of class imbalance in the NSL-KDD and the more recent CSE-CIC-IDS2018 datasets by augmenting the underrepresented minority samples. This approach ad-dresses the class imbalance challenge in intrusion detection, facilitating improved learning of distinctions during classifier training. Subsequently, we conducted a performance comparison of these sampling techniques.
3. We use the classification model Extreme Gradient Boosting comparing with other methods which include Support Vector Machine (SVM), Logistic Regression, KNN, Decision Tree, Random Forest(RF).
4. We conducted a comparison of the suggested model with SMOTE and other methods, evaluating it through metrics including accuracy, precision, f-measure, recall, false alarm rate, and execution time.

The structure of this paper is as follows: Section 2 delves into the relevant literature; Section 3 outlines the proposed methodology; Section 4 provides an account of the results and subsequent discussion. Lastly, Section 5 brings the paper to a conclusion.

2. Literature review

Within the domain of machine learning, addressing the issue of class imbalance has consistently presented a significant challenge. Consequently, intrusion detection confronts substantial difficulties when dealing with highly imbalanced categories in network traffic. As a result, numerous researchers have embarked on investigations to enhance the accuracy of intrusion detection for imbalanced network traffic data. Notably, an Information Entropy Deep Belief Networks (IE-DBN) model was introduced [22] for network Intrusion Detection Systems (IDS), employing the KDDCup99 dataset. The study employed Information gain (IG) for dimensionality reduction by eliminating redundant features. Information entropy (IE) was used to determine the DBN network's depth and hidden neuron count. Furthermore, in addressing the issue of uneven class distribution, we utilized the Synthetic Minority Oversampling Technique (SMOTE) algorithm. This approach yielded a 0.76 percent false positive rate and an impressive 98.76 percent detection accuracy, as evidenced by the obtained results. However, class imbalance within the data remains a concern, since the study showed a reduction in detection efficiency for assaults on minority classes, particularly when dealing with huge datasets, even when employing SMOTE. Furthermore, the researchers did not explore putting their model to the test with con-temporary forms of attack or using evaluation measures beyond false positive and accuracy. Piyasak introduced a method aimed at enhancing the accuracy of minority classification [23]. This method addresses imbalanced data classification challenges by combining the Complementary Neural Network (CMTNN) with the Synthetic Minority Over-sampling Technique (SMOTE). Tests carried out on the UCI dataset show that this combination method successfully reduces issues with class imbalance.

Yan proposed an enhanced local adaptive composite minority sampling algorithm (LA-SMOTE) to handle the issue of network traffic imbalance and incorporated a deep learning GRU neural network to identify network traffic anomalies [24]. In another study [25] the researchers addressed imbalanced dataset CIDDS001 through data Upsampling and Downsampling methods. To assess the datasets, they used Random Forest, Deep Neural Networks, Voting, Variational Autoencoder, and Stacking Machine Learning classifiers. Their approach resulted in an accuracy rate of up to 99.99 percent.

Recently, a study [26] involved training a deep autoencoder to establish a data generation model for generating balanced datasets. This approach was shown to alleviate overfitting issues caused by imbalanced data and prevent misjudgment of new data types not present in the training dataset. In a different approach [27] introduced a novel Intrusion Detection System (IDS) based on the Siamese Neural Net-work (Siamese-NN). The suggested Siam-IDS could identify R2L and U2R assaults without using conventional class balancing methods like random under- and over-sampling. Siam-IDS outperformed comparable products in terms of recall values for both R2L and U2R attack categories.

The researchers [12] introduced an Intrusion Detection System (IDS) model, utilizing the adaptive synthetic (ADASYN) oversampling technique in conjunction with the LightGBM ensemble. The ap-

plication of ADASYN was employed to address the challenge of imbalanced data. Additionally, the classifier employed the ensemble method LightGBM to lessen computational complexity related to model training and speed up intrusion detection. UNSW-NB15, NSL-KDD, and CICIDS2017 were the three datasets utilized to evaluate the model's performance. The experimental findings showed that LightGBM had accuracy values of 83.98 percent, 89.79 percent, and 99.86 percent for the corresponding datasets. Additionally, the identification rate for minority classes was improved after using ADASYN oversampling, resulting in 85.89 percent, 92.57 percent, and 99.91 percent overall detection accuracy for the same datasets. But because the false positive rate is somewhat large, the model can be enhanced to deal with this issue. In order to distinguish between different forms of network attacks, researchers in a study [28] investigated five different machine learning techniques using the UNSW-NB15 dataset. These techniques included random forest, decision tree, logistic regression, K-Nearest Neighbors, and artificial neural networks. With an accuracy of 89.29 percent, the random forest classifier performed the best out of all the classifiers. Furthermore, significant gains in classification model accuracy were noted once the class imbalance problem was resolved by applying the synthetic minority oversampling technique (SMOTE). Surprisingly, the random forest classifier, which used 24 specific features from the principal component analysis approach achieved the best accuracy, hitting 95.1 percent. It is crucial to mention that the LR and ANN classifiers did not benefit from class balance; in fact, it led to a reduction in their accuracy, especially in managing the minority classes.

The uneven distribution of attacks poses a significant research challenge in the fields of intrusion detection and IoT security. To address this problem, researchers [29] used Borderline-SMOTE to oversample the minority class of rare-class attack traffic in the Internet of Things. This improved the detection accuracy of IoT attacks, particularly in situations where there is a class imbalance. Another study [30] used Generative Adversarial Networks (GAN) to learn the distribution of rare-class attack traffic and generate synthetic rare-class attack data in order to address the low detection rates of conventional classifiers for rare-class attack traffic in IoT settings. The detection effectiveness of intrusion detection models built on convolutional neural networks (CNNs) was somewhat improved by this method. Furthermore, researchers [31] generated artificial rare-class attack samples using a Wasserstein Conditional Generative Adversarial Network (WCGAN). After that, they used a balanced dataset to train an XGBoost classifier. The training data oversampled by WCGAN demonstrated improved classifier training and increased the recognition rate of intrusion detection models in testing on publicly accessible IoT datasets, particularly for rare-class intrusions.

In their study, [32] offer a unique method that uses an optimized kernel density estimation algorithm: a geometric synthetic minority oversampling technique. By learning the distribution of rare-class attack data, this method may generate a large variety of such data while maintaining closeness with the original sample features. The NSL-KDD and N-BaIoT datasets were used to evaluate the system's performance, and the results demonstrated a multi-classification accuracy of 86.39 percent and 99.94 percent, respectively.

Reference [33] addresses the issue of class imbalance by introducing a novel technique known as the Difficult Set Sampling Technique (DSSTE). Initially, the imbalanced training set is divided into two sets by the algorithm: an easy set and a challenging set, using the Edited Nearest Neighbor (ENN) approach. It then reduces the majority samples in the challenging set using the KMeans technique, hence decreasing their representation. The method also modifies the minority samples' continuous features in the challenging set to produce new samples that enhance the minority class. Lastly, a new training set is created by combining the augmentation samples from the easy set, the minority set in the difficult set, and the compressed majority set inside the difficult set. Through targeted data augmentation for the minority class and effective mitigation of the original training set's imbalance, this approach improves the classifier's capacity to identify distinctions during the training phase and ultimately leads to improved classification performance. The NSL-KDD dataset yielded an accuracy of 80.69 percent and an F-score of 79.34 for the DSSTE method, whereas the CSE-CIC-IDS2018 dataset yielded an accuracy and precision of 96.29 percent, according to experimental results.

Many researchers have balanced the training set and produced better experimental results by using interpolation, oversampling, encoder-generated data, and other data augmentation techniques. SMOTE [34, 35], Borderline-SMOTE, ADASYN, and various other algorithms [12, 36] represent traditional oversampling techniques. However, they share a common approach of generating synthetic samples along the line segments that connect instances of the minority class. This method presents challenges when attempting to enhance the minority class sample distribution. G-SMOTE [37], on the other hand, is a synthetic minority oversampling method designed to deal with the problem of imbalanced data. It is an improved variant based on the SMOTE algorithm. G-SMOTE differs from the previously stat-ed SMOTE-based algorithms in that it does not rely solely on the creation of artificial samples along line segments that link instances of minority classes. Rather, G-SMOTE introduces geometric modifications into the feature space, hence expanding the capabilities of the linear interpolation process. This novel method mimics the distribution of rare-class samples more accurately. Although these techniques provide data that is nearly identical to real data and successfully increase the minority class, problems may arise if test data distributions differ from the trained range. The classifier may struggle to accurately predict these distributions. To address this, we propose the Variational Autoencoder algorithm, which identifies challenging samples within the imbalanced training set, and adjusts the continuous attributes of the minority class. This method effectively reduces imbalance and generates data that aligns with the true distribution.

3. Methodology

Confronting the challenge of imbalanced network traffic, we introduce the variational autoencoder to equalize the number of minority samples with that of the majority samples, effectively reducing the imbalance in the training set. This adjustment enhances the classification accuracy attainable by the intrusion detection system. We employ the Extreme Gradient Boosting technique for classification, and its performance is compared with that of other classifiers such as Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Decision Tree classifiers in our classification models. Figure 1 illustrates the architecture of the proposed model. In our intrusion detection system, we initially perform data pre-processing, which includes tasks such as duplicate removal, outlier handling, and managing missing values. Following this, we partition the dataset into a test set and a training set. The training set undergoes Min-Max normalization to standardize the data, ensuring that all values fall within the same range, thus expediting the convergence process. Additionally, the preprocessed data is subjected to our proposed variational autoencoder algorithm to address data balancing. Subsequently, the processed training set is used to train the classification model, and the model's performance is assessed using the test set.

3.1. Proposed methods

The CSE-CIC-IDS2018 dataset has a major class imbalance, with around 83 percent of the data being benign network traffic [38]. Furthermore, in the NSL-KDD dataset, network traffic is skewed toward the majority class. To guarantee equitable class distribution, the SMOTE and VAE oversampling technique was applied and compared to see which produces the best results.

3.2. Dataset description

CSE-CIC-IDS2018 and NSL-KDD dataset are the two datasets employed for the study. There are 125 973 records in the NSL-KDD dataset, which are split up into 22 544 records for the training and testing sets. Each record in this dataset is com-posed of 41 attributes, which encompass 3 nominal, 6 binary, and 32 numeric attributes. The dataset encompasses instances of both normal network activity and different attack types. There are twenty-two distinct attack types in the training dataset and seventeen distinct attack types in the test dataset, for a total of 39 assault types. These attack types fall under four categories: Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). The CSE-CIC-IDS2018 dataset was created in 2018 by the Canadian Institute of Cyber Security (CIC) and the Communications Security Establishment (CSE). It stands as the most recent

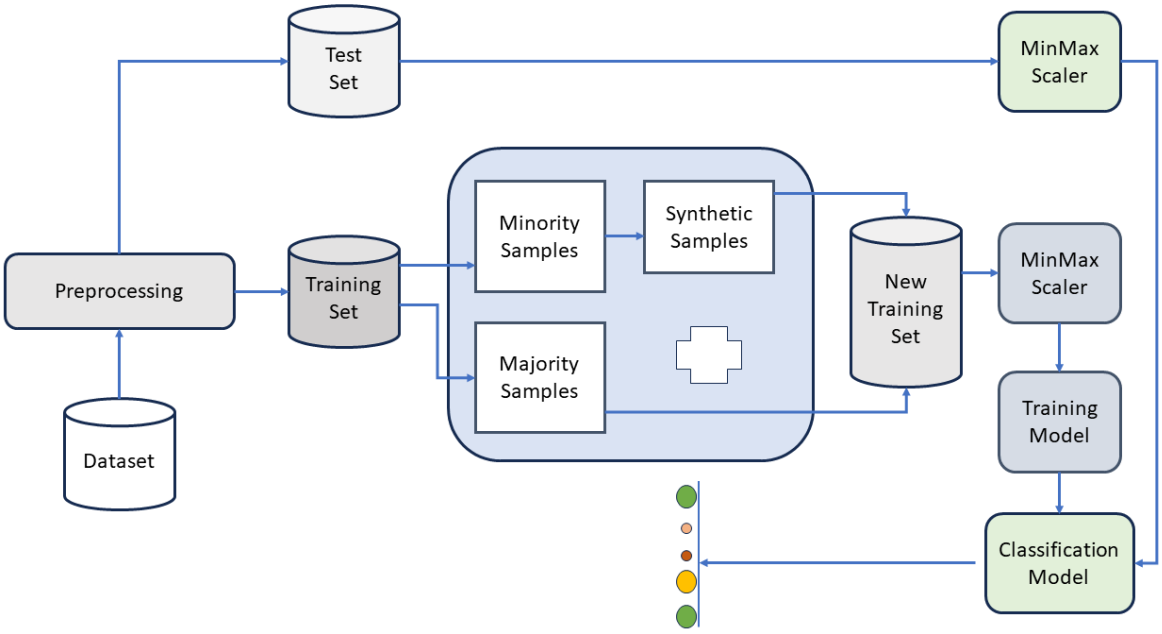

Fig. 1. Architecture of the Proposed Model.

and comprehensive publicly available dataset for intrusion detection. CSE-CIC-IDS2018 is designed for simulating real attack scenarios and represents an enhancement over its predecessor, the CSE-CIC-IDS2017 dataset. This dataset adheres to established standards for attack datasets and encompasses a variety of well-known attack types. Specifically, it includes a detailed breakdown of seven distinct assault scenarios, including brute force, online attacks, denial of service (DoS), distributed denial of service (DDoS), infiltration, and heartleech, which is a kind of DoS attack. CSE-CIC-IDS2018 dataset has a total of 83 samples, each of which has a unique collection of attributes. The network traffic distribution is shown in Table 1.

Table 1. Distribution of NSL-KDD and CSE-CIC-IDS2018 Dataset.

NSLKDD					**CSE-CIC-IDS2018**	
Classes	Train Data	Test Data	Attack Type	Total	Traffic Type	Count
DoS	45927	7458	apache2, mailbomb, processtable, up-dstorm, land, neptune, smurf, pod, back, teardrop,	11	Benign	83.07
Probe	11656	2421	mscan, saint, portsweep, ipsweep, satan, nmap	6	DDoS	7.786
R2L	995	2754	Warezclient, guess-passwd, ftp-write, waresmaster, imap, named, snmpgetattack, snmpguess, xlock, xsnoop, multihop, spy, and phf	15	DoS	4.031
U2R	52	200	rootkit, ps, sqlattack, loadmodule, perl, buffer-overflow, and xterm	7	Brute Force	2.347
Normal	67343	9711			Botnet Infiltration Web Attack	1.763 0.997 0.006
Total	125973	22544		39		100

3.3. Dataset preprocessing

We performed data preprocessing in this section to address various issues such as duplicate values, noisy data, missing data, infinity values, and the presence of categorical data requiring transformation. Specifically, in the case of the CSE-CIC-IDS2018 dataset, certain features like "Timestamp", "Destination Address", "Source Address", and "Source Port" were removed. Additionally, if the features "Init Bwd Win Byts" and "Init Fwd Win Byts" contained a -1 value, we introduced two binary check dimensions. A value of -1 was marked as 1, while any other value was marked as 0.

3.4. One hot encoding

We utilized the One Hot Encoding technique to transform the categorical feature data (as seen in NSLKDD, protocol type, service, and flag, for instance) into numerical data. As seen in equation (1). This process involved creating a novel binary feature for each potential category and attributing a value of 1 to the feature of each sample that corresponds to its original category,

$$e_i = 1_A(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{if } x \notin A. \end{cases} \tag{1}$$

For each vector e_i in the standard base, where e_i represents a vector with 1 in the ith position and 0s elsewhere, we have a set A. If x is an element within A, the function returns 1; otherwise, it returns 0. Consequently, A represents the set of instances where a 1 is assigned to the encoding vector. Essentially, one-hot encoding is a vectorized version of this indicator function, applied component-wise. Consider three protocol types in the NSLKDD data: "TCP", "UDP", and "ICMP". Before one-hot encoding, they were functions of these three categories. However, after one-hot encoding, they are transformed into binary vectors $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, respectively. The protocol type function for the NSLKDD data was split into three groups, while the flag function included 11 categories, and the service function contained 70 categories. Therefore, after applying one-hot encoding, the initial 41-dimensional feature vector expands to 122 dimensions.

3.5. Min-Max normalization

We used Min-Max normalization to rescale the input data within the range of $(0, 1)$ as a technique for feature scaling. The features in the dataset have different value ranges, so feature scaling was used to normalize these characteristics and make sure they all fall inside the same range. This approach prevents the model we constructed from favoring specific features solely because they have larger values. We performed feature scaling through the Min-Max technique, as described in equation (2):

$$x = \frac{(x_i - x_{\min})}{x_{\max} - x_{\min}}. \tag{2}$$

In this context, x_i represents the numerical feature value of the ith sample, while $x_{\min}$ and $x_{\max}$ indicate, respectively, the lowest and highest values for each numerical attribute. This technique is employed to enhance the convergence of gradient descent and to achieve an improved regularization effect.

3.6. Extreme gradient boosting

XGBoost is an iterative ensemble learning technique designed to combine the results of several decision trees in order to increase prediction accuracy. It continuously creates trees and optimizes their characteristics to minimize a cost function. This cost function is composed of the regularization terms and separate loss functions that help control the complexity of the model. The algorithm's strength is in its ability to identify complex correlations in the data without causing overfitting through regularization. XGBoost has been widely used in machine learning competitions and practical applications because to its efficiency and speed. In actual use, it is really effective. It has a good predictive power and can handle both regression and classification problems. With n samples and m features, our labeled dataset D (NSL-KDD and CSE-CIC-IDS2018) is shown as

$$D = (x_i, y_i) \tag{3}$$

for i in the interval $[1, n]$, where the target value is denoted by y_i and the feature vector of the ith sample is represented by x_i. A weighted sum of each sample's unique loss functions plus regularization terms makes up the objective function in XGBoost:

$$L(\Theta) = \sum_{i=1}^{n} \big[L(y_i, \hat{y}_i) + \Omega(f_i)\big], \tag{4}$$

where $L(\Theta)$ is the individual loss function that measures the difference between the true target and the predicted target $\hat{y}_i$ for the ith sample. $\Omega(f_i)$ is the regularization term for each tree f_i in the ensemble, which helps control model complexity. Θ represents the entire set of model parameters, including the parameters of individual trees and the weights. An ensemble of decision trees is constructed via XGBoost. The weighted sum of the predictions made by each individual tree determines the expected goal for a particular sample:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \tag{5}$$

where K is the total number of individual trees, $f_k(x_i)$ represents the prediction of the k-th tree for the i-th sample. Two different kinds of regularization terms are added to the goal function by XGBoost. L1 Regularization: by including the absolute values of the tree node weights, this promotes sparsity in the feature selection,

$$\Omega(f_i) = \gamma \sum_{j=1}^{L} |\theta_j|, \tag{6}$$

where L is the total number of leaves in the tree, θ_j represents the weight of the j-th leaf, γ is the regularization parameter. L2 Regularization: adds the squares of the weights to control the complexity of the tree,

$$\Omega(f_i) = \gamma \sum_{j=1}^{L} |\theta_j^2|, \tag{7}$$

where L is the total number of leaves in the tree, θ_j represents the weight of the j-th leaf, γ is the regularization parameter. By adding a single tree at a time, XGBoost optimizes the objective function. It accomplishes this by employing strategies like gradient boosting to minimize the goal function iteratively. A fresh tree is trained to roughly match the loss function's gradient in each iteration, making the tree well-trained. XGBoost integrates the forecasts of each individual tree in the ensemble, taking into account their weighted contributions, to provide predictions for future samples.

3.7. Variational autoencoder

The CSE-CIC-IDS2018 dataset has a major class imbalance, with around 83 percent of the data being benign network traffic [38]. Furthermore, in NSL-KDD, there is an imbalance in the distribution of network traffic, with a bias toward the majority class. To ensure a fair distribution among classes, the Variational AutoEncoder (VAE) was used to generate synthetic observations. The objective is to approximate the true distribution of the minority class, denoted as, $x \sim P_\theta(x)$, and then randomly sample from this approximated distribution to create new synthetic observations, represented as, $\tilde{x}$. Finally, a classification model is trained on the updated dataset, which now has a more balanced class distribution, including the synthetic minority class observations. In this process, the input data x and the latent variables z are introduced, and consideration is given to the joint distribution of x and z which is influenced by the parameter θ. This parameter is optimized during the training phase:

$$P_\theta(x, z). \tag{8}$$

The latent variables, denoted as z, can be regarded as a representation of the most essential information from x, as they exclude any additional information. Given the goal of approximating, a practical approach would involve integrating the joint distribution with respect to z,

$$P_\theta(x) = \int P_\theta(x|z)\, P_\theta(x)\, dz. \tag{9}$$

Nonetheless, it is worth noting that $P_\theta(x|z)$ is challenging to compute [8]. This is where the encoder, denoted as $Q_\phi(x|z)$ with parameter ϕ comes into play. The reason for approximating the encoder is twofold: it enables the estimation of $P_\theta(x|z)$ and ensures that the process remains computationally feasible,

$$Q_\phi(z|x) \approx P_\theta(z|x). \tag{10}$$

Furthermore, it is assumed that the encoder's distribution follows a multivariate normal distribution, where the mean and standard deviation are determined based on the input variables x,

$$Q_\phi(z|x) \sim N(\mu(x), \mathrm{diag}(\sigma(x))) \tag{11}$$

As previously mentioned, the decoder is responsible for the reconstruction aspect of the VAE model, utilizing z as input to recreate x. The decoder can be determined by assessing the disparity between the encoder, denoted as $Q_\phi(z|x)$ and $P_\theta(z|x)$. To estimate $P_\theta(z|x)$ using the encoder, the Kullback–Liebler (KL) divergence is introduced. The KL-divergence serves to quantify the difference between the encoder and the distribution, $P_\theta(z|x)$, that it aims to estimate,

$$D_{KL}(Q_\phi(z|x)||P_\theta(z|x)) = E_{z\sim Q_\phi}(\log(Q_\phi(z|x)) - \log(P_\theta(z|x))). \tag{12}$$

The loss function of the VAE model comprises two elements: the KL Loss and the reconstruction loss. In terms of parameter optimization, the decoder component indicates that maximizing the decoder results in minimizing the reconstruction loss. Meanwhile, the KL loss component reveals that as the encoder improves its ability to encode z, the loss between the encoded z and its prior z decreases. During the optimization of the VAE model, gradients of the loss function with respect to its parameters are calculated. To reiterate, the loss function of the VAE model is composed of two elements: the KL loss and reconstruction loss,

$$L_{\mathrm{recon}}(x, \phi, \theta) = -E_{Q_\phi(z|x)}[\log(P_\theta(z|x))], \tag{13}$$

$$L_{\mathrm{KL}}(x, \theta) = \frac{1}{2}\sum 1 + Z_{\log_{\mathrm{var}}} - Z^2_{\mathrm{mean}}, \tag{14}$$

$$L_{\mathrm{VAE}}(x, \phi, \theta) = L_{\mathrm{recon}}(x, \phi, \theta) + \beta \cdot L_{KL}(x, \theta). \tag{15}$$

3.8. Synthetic Minority Oversampling Technique (SMOTE)

The SMOTE [39] over-sampling technique works on the following principle: Consider Z as the size of a relatively small class, consisting of samples denoted as i, each with its associated feature vector x_i, where i ranges from 1 to Z

1. Locate k neighbors for the sample x_i from among all Z samples within this relatively small class (for instance, by employing the Euclidean Distance), and designate them as $x_i(near)$, where $near$ spans from 1 to k.
2. From the k neighbors, a sample denoted as $x_i(nn)$ will be chosen randomly. Subsequently, a random number ζ_1, ranging from 0 to 1, is generated to create a new sample x_{i1} using the following equation: $x_{i1} = x_i + \zeta_1 \cdot (x_i(nn) - (x_i))$;
3. Repeat the process described in step b N times to generate N new samples: $x_{i\,new}$, where new ranges from 1 to N.

3.9. Machine learning algorithm

When evaluating our classifier's architecture against other models, we employed Logistic Regression, K-Nearest Neighbor, Random Forest, SVM, and Decision Tree for the training and testing processes, as elaborated in the subsequent section.

K-Nearest Neighbor (KNN). For identifying data samples, the K-nearest neighbor (k-NN) method is a basic and simple nonparametric technique. Using this method, an unlabeled data point is assigned to the class represented by its K-nearest neighbors after approximating the distances between different data points in input vectors. The parameter 'k' must be carefully chosen when building a k-NN classifier because different values of 'k' can result in differing classification performance. Longer classification durations and perhaps lower prediction accuracy can arise from using a high number of

neighbors in the prediction process if 'k' is sufficiently large. K-nearest neighbor sets itself apart from inductive learning techniques by being classified as an instance-based learning technique. In contrast to inductive techniques, k-NN does not have a separate model training phase; instead, it classifies new cases by only looking for examples that are comparable inside input vectors. Amazingly, K-nearest neighbor functions without any settings and usually uses the Euclidean distance metric to measure how different neighbors are from one another.

Support Vector Machines (SVM). The idea behind the mathematical description of SVM is to identify a hyperplane that minimizes classification errors while maximizing the margin between classes. This strategy has solid theoretical underpinnings and is supported by science. SVM aims to produce a strong decision boundary that performs well when applied to unknown data by optimizing the margin. Finding the hyperplane that optimally divides data into distinct classes while maximizing the margin between the two classes is the fundamental notion of SVM. A support vector machine (SVM) is used in the context of an intrusion detection system (IDS) to classify network traffic and identify possible intrusions or cyberattacks. SVMs are used by IDS to categorize network traffic data into categories such as legitimate and possibly harmful activity. They build a model that is able to discriminate between suspicious or malicious activity and typical network behavior by using information taken from network packets or logs. SVMs are trained using labeled datasets of historical network traffic data, where anomalies are identified as known intrusion instances. In order to distinguish between typical and abnormal network behavior, the SVM algorithm looks for a hyperplane. Real-time classification of incoming network traffic is possible with the SVM model once it has been trained. Because SVMs are well-known for handling high-dimensional data, they can be used to analyze a variety of network properties and identify intricate infiltration.

Decision Trees. A Decision Tree is an essential tool for categorization tasks in Intrusion Detection Systems (IDS). Using a sequence of interconnected judgments, each of which affects the next, Decision Trees classify network traffic or data samples. The visual representation of this decision-making process takes the shape of a tree structure, where the root node serves as the first classification point. In intrusion detection systems (IDS), decision trees are frequently used to evaluate network data and determine if it indicates potentially malicious or benign activities. A trained Decision Tree identifies pertinent characteristics from the network data to classify a specific data point which may represent a benign occurrence or an intrusion by selecting pertinent attributes. The objective in the context of IDS is to construct an ideal Decision Tree that contains the most amount of information while maintaining the shortest tree structure. The optimum tree effectively distinguishes between typical and non-normal network behavior. The optimal root node for the tree is deter-mined using a variety of measures, including the Gini index and Information Gain. This effectively divides the training dataset into several categories, such as normal and incursion. This procedure aids in locating any dangers and security holes inside a network. To put it briefly, decision trees are employed in intrusion detection systems to categorize network data by building a hierarchical tree of decisions depending on the data's characteristics. This makes it easier to identify abnormalities and cyberthreats.

Random Forest (RF). Among the supervised machine learning methods is Random Forest (RF). It is made by combining many Decision Trees (DTs) in order to improve the precision and robustness of categorization predictions. Each of these distinct Decision Trees is built at random and trained using a majority vote system to generate categorization results. Although they are parts of the Random Forest, Decision Trees are two different categorization algorithms. The primary distinction is that Random Forest builds a rule-subset using all of the Decision Trees that are members, as opposed to Decision Trees, which create a rule-set during training for the purpose of later classifying fresh samples. This novel method produces outputs that are more accurate and resilient to overfitting, requiring less input and doing away with the requirement for a separate feature selection procedure. Numerous research indicate that Random Forest is a good fit for applications related to anomaly and intrusion detection in network security.

Logistic Regression (LR). For jobs involving binary classification, statistical and machine learning techniques like logistic regression are employed. It estimates the likelihood that the outcome belongs to a specific class in order to represent the link between one or more independent variables (features) and a binary dependent variable (outcome). A linear combination of input data is transformed into a value between 0 and 1, which represents the likelihood of the positive class, by the logistic function (sigmoid function) in the logistic regression model. Simplicity, interpretability, and the capacity to generate probabilistic predictions are among the main features of logistic regression. When it comes to generating predictions and decisions, logistic regression is an effective technique for figuring out how input features relate to the possibility that an event will occur.

4. Result and discussion

Initially, we studied the classifier's performance. While Random Forest obtained the highest precision and Fscore of 87.23 and 81.37%, respectively, in the experimental results for the NSL-KDD dataset, XGBoost achieved the highest accuracy of 86.76% and the highest recall of 84.07%. However, XGBoost required more computational time to execute than Decision Tree. Random Forest obtained the maximum precision and F1 rate of 90.12% and 91.37%, whereas XGBoost achieved the highest accuracy and recall rate of 99.40% and 94.07% following oversampling with SMOTE. After oversampling with Variational Autoencoder, XGBoost achieved the highest accuracy rate of 99.79%, Decision Tree achieved the highest precision and recall rate of 99.63% and 89.07%, while Random Forest achieved the highest f1-score of 94.04%, XGBoost had the longest execution time while DT had the least execution time. For the CSE-CIC-IDS2018 dataset, Random Forest achieved the highest accuracy, precision, and f1-score of 96.66%, 89.11%, and 83.17% while XGBoost achieved the highest recall of 89.71% respectively, however SVM expended more computational time while Decision Tree has the least execution time. After oversampling with SMOTE, XGBoost achieved the highest accuracy and recall rate of 99.87% and 94.18%, and Random Forest achieved the highest precision and F1 rate of 90.23% and 91.48%, SVM had the longest execution time while DT had the least execution time. After oversampling with Variational Autoencoder, XGBoost achieved the highest accuracy rate of 99.89%, Decision Tree achieved the highest precision and recall rate of 99.74% and 90.08%, while Random Forest achieved the highest f1-score of 94.15%, XGBoost had the longest execution time while DT had the least execution time.

Table 2. Comparison of the Proposed Model with other related work.

Model	NSLKDD					CSE-CIC-IDS2018				
	Acc	**Pre**	**Rec**	**F1**	**Time**	**Acc**	**Pre**	**Rec**	**F1**	**Time**
LR	83.2	61.4	79.2	59.4	23.6	93.1	62.1	87.1	61.1	23.4
KNN	86.3	79.8	79.0	71.6	319	96.2	80.5	87.1	73.3	338
SVM	84.4	69.8	79.5	61.1	617	94.4	70.6	88.7	62.4	639
DT	86.5	83.1	77.0	75.2	6.44	96.5	84.0	83.1	79.1	8.35
RF	86.6	87.2	79.0	81.3	12.4	96.6	89.1	81.5	83.1	14.2
XGB	86.7	81.3	84.0	69.8	432	96.3	82.0	89.7	71.7	459
SMOTE+LR	96.2	68.3	89.2	64.4	26.8	96.4	68.4	89.3	64.5	24.7
SMOTE+KNN	99.3	84.6	89.0	81.6	449	99.5	84.8	89.1	81.7	449
SMOTE+SVM	97.5	70.7	89.5	66.0	839	97.6	70.8	89.6	67.0	839
SMOTE+DT	99.6	86.0	87.0	85.1	9.53	99.7	86.1	87.1	85.3	6.44
SMOTE+RF	99.7	90.1	89.0	91.3	18.6	99.8	90.2	89.1	91.4	15.4
SMOTE+XGB	99.4	84.1	94.0	79.8	559	99.8	84.3	94.1	79.9	549
VAE+LR	98.7	82.8	80.8	86.0	9.34	98.8	82.9	80.9	86.1	7.34
VAE+KNN	99.3	98.1	84.1	89.7	287	99.4	98.2	84.2	89.8	277
VAE+SVM	99.2	98.2	84.3	89.3	821	99.3	98.3	84.4	89.4	823
VAE+DT	99.7	99.6	89.0	89.3	11.6	99.8	99.7	90.0	89.4	9.66
VAE+RF	99.7	99.5	87.3	94.0	26.2	99.8	99.6	87.4	94.1	16.2
VAE+XGB	**99.7**	**99.1**	**84.4**	**79.0**	**1412**	**99.8**	**99.1**	**84.5**	**80.0**	**1400**

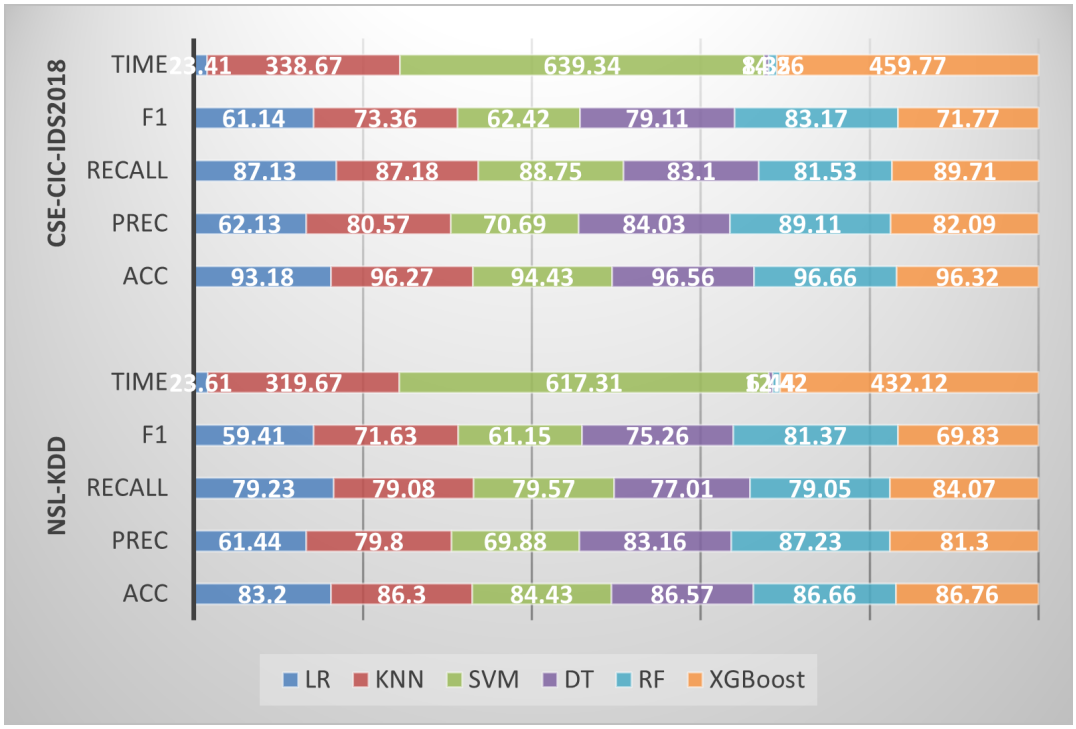

Fig. 2. Comparison of methods without Sampling Technique.

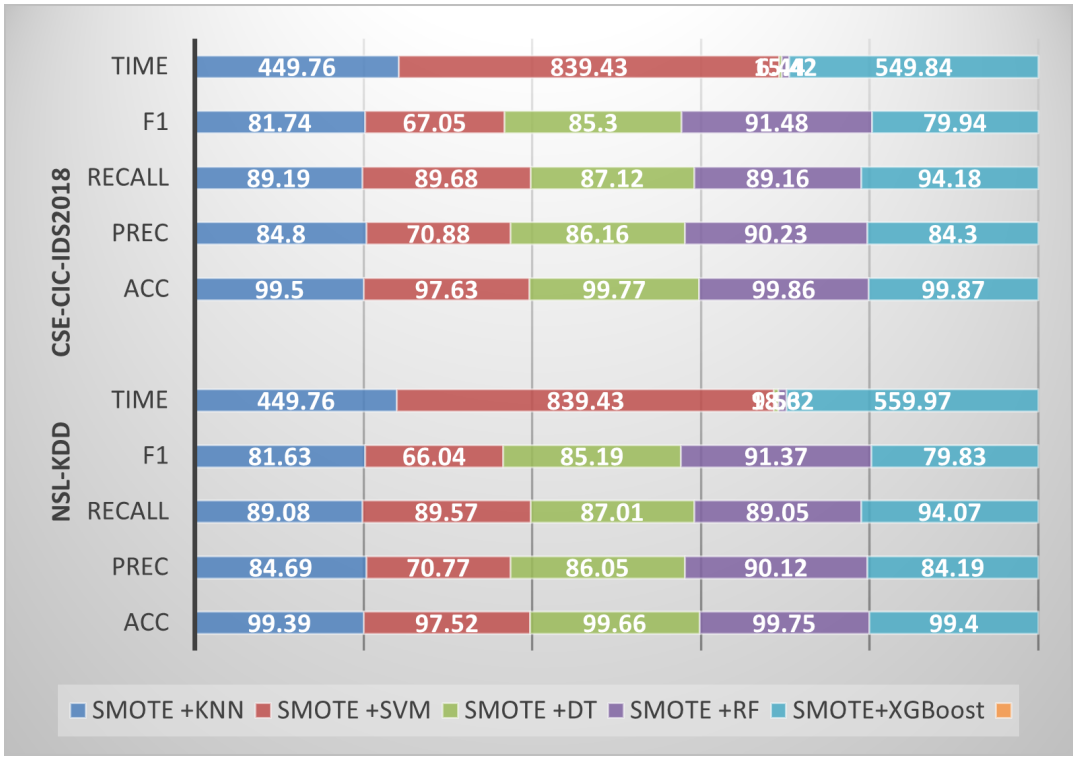

Fig. 3. Comparison of methods combined with Smote Sampling Technique.

As seen in Table 3, the accuracy outcome of our study effort exhibits greater performance when compared to other pertinent related work, however it exhibits high resource and time complexity.

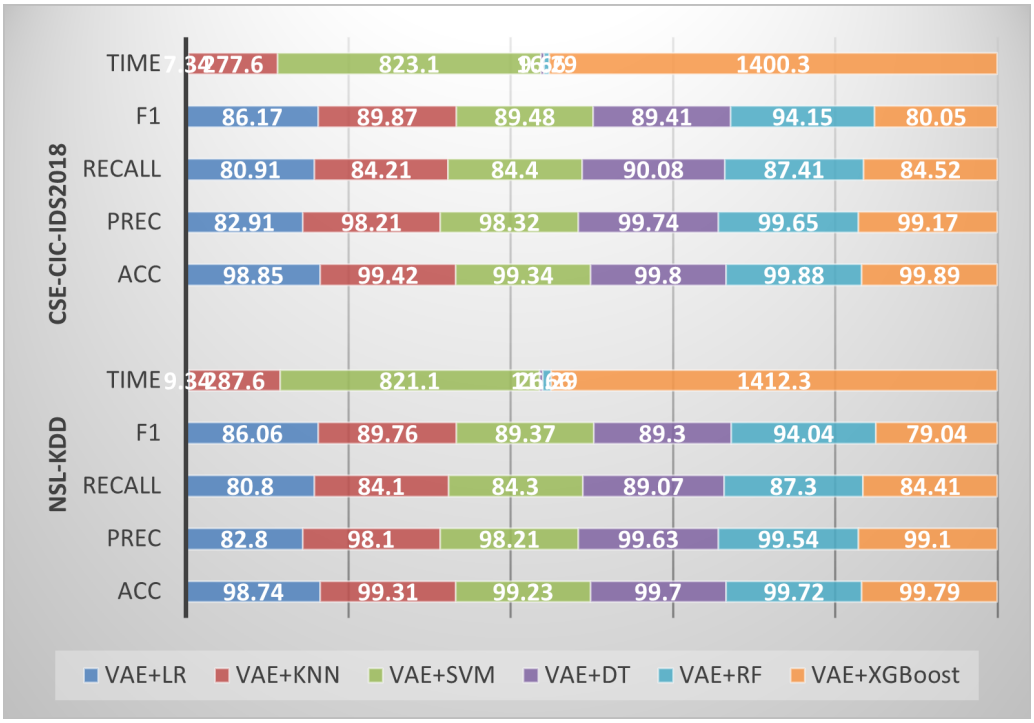

Fig. 4. Comparison of methods combined with Variational Autoencoder Sampling Technique.

Table 3. Comparison of the Proposed Model with other related work.

Research Papers	Accuracy	Precision	Recall	F-Score	Execution Time (sec)
Ref. [40]	98.1	99.1	N/A	N/A	0.18
Ref. [21]	99.4	N/A	N/A	N/A	N/A
Ref. [22]	98.8	N/A	N/A	N/A	52.64
Ref. [18]	76.6	96.0	61.6	75.0	1220
Ref. [41]	87.0	87.0	83.0	83.2	N/A
Ref. [42]	96.1	N/A	N/A	N/A	N/A
Proposed	**99.7**	**99.1**	**84.4**	**79.0**	**1400**

5. Conclusion

The burden on network intrusion detection is growing as network intrusion continues to change. Cyberspace security is particularly vulnerable because of issues with unbalanced network traffic, which make it harder for intrusion detection systems to anticipate the dispersion of harmful attacks. In order to increase the learning of imbalanced network data by the classification model, this research suggested a variational autoencoder. In order to improve classification accuracy, a deliberate increase in the number of minority samples that must be learned might lessen network traffic imbalance and reinforce minority learning under difficult samples. In order to compare XGBoost with five traditional machine learning and deep learning classification algorithms, we paired it with SMOTE and variational autoencoder sampling methods. Tests demonstrate that our approach can more successfully detect attacks and precisely identify the samples in the unbalanced network traffic that require expansion. The study's findings were able to show how important it is to handle class imbalance by employing the oversampling technique, as well as how this affects the overall effectiveness of intrusion detection. After sampling the imbalanced training set samples with the variational autoencoder algorithm, we discovered in the experiment that XGBoost performs more accurately than other machine learning methods, despite having a higher computational complexity than other methods. Consequently, we want to apply feature selection approaches in the following stage to shorten the execution time while enhancing the model's overall performance.

[1] Abdulganiyu O. H., Ait Tchakoucht T., Saheed Y. K. A systematic literature review for network intrusion detection system (IDS). International Journal of Information Security. **22** (5), 1125–1162 (2023).

[2] Mallouk I., Abou el Majd B., Sallez Y. A generic model of the information and decisional chain using Machine Learning based assistance in a manufacturing context. Mathematical Modeling and Computing. **10** (4), 1023–1036 (2023).

[3] Khoroshchuk D., Liubinskyi B. Machine learning in lung lesion detection caused by certain diseases. Mathematical Modeling and Computing. **10** (4), 1084–1092 (2023).

[4] Merzouk S., Gandoul R., Marzak A., Sael N. Toward new data for IT and IoT project management method prediction. Mathematical Modeling and Computing. **10** (2), 557–565 (2023).

[5] Bridges R. A., Glass-Vanderlan T. R., Iannacone M. D., Vincent M. S., Chen Q. A Survey of Intrusion Detection Systems Leveraging Host Data. ACM Computing Surveys. **52** (6), 1–35 (2020).

[6] Abdulganiyu O. H., Tchakoucht T. A., Saheed Y. K. Towards an efficient model for network intrusion detection system (IDS): systematic literature review. Wireless Networks. **30**, 453–482 (2024).

[7] Aldweesh A., Derhab A., Emam A. Z. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. Knowledge-Based Systems. **189**, 105124 (2020).

[8] Kayode Saheed Y., Harazeem Abdulganiyu O., Ait Tchakoucht T. A novel hybrid ensemble learning for anomaly detection in industrial sensor networks and SCADA systems for smart city infrastructures. Journal of King Saud University – Computer and Information Sciences. **35** (5), 101532 (2023).

[9] Masdari M., Khezri H. A survey and taxonomy of the fuzzy signature-based Intrusion Detection Systems. Applied Soft Computing. **92**, 106301 (2020).

[10] Masdari M., Khezri H. Towards fuzzy anomaly detection-based security: a comprehensive review. Fuzzy Optimization and Decision Making. **20** (1), 1–49 (2021).

[11] Gu J., Wang L., Wang H., Wang S. A novel approach to intrusion detection using SVM ensemble with feature augmentation. Computers & Security. **86**, 53–62 (2019).

[12] Liu J., Gao Y., Hu F. A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. Computers & Security. **106**, 102289 (2021).

[13] Nazir A., Khan R. A. A novel combinatorial optimization based feature selection method for network intrusion detection. Computers & Security. **102**, 102164 (2021).

[14] Sohi S. M., Seifert J. P., Ganji F. RNNIDS: Enhancing network intrusion detection systems through deep learning. Computers & Security. **102**, 102151 (2021).

[15] Zhang J., Ling Y., Fu X., Yang X., Xiong G., Zhang R. Model of the intrusion detection system based on the integration of spatial-temporal features. Computers & Security. **89**, 101681 (2020).

[16] Selvakumar B., Muneeswaran K. Firefly algorithm based feature selection for network intrusion detection. Computers & Security. **81**, 148–155 (2019).

[17] Mebawondu J. O., Alowolodu O. D., Mebawondu J. O., Adetunmbi A. O. Network intrusion detection system using supervised learning paradigm. Scientific African. **9**, e00497 (2020).

[18] Wang Z., Liu Y., He D., Chan S. Intrusion detection methods based on integrated deep learning model. Computers & Security. **103**, 102177 (2021).

[19] Ashiku L., Dagli C. Network Intrusion Detection System using Deep Learning. Procedia Computer Science. **185**, 239–247 (2021).

[20] Bhati B. S., Rai C. S., Balamurugan B., Al-Turjman F. An intrusion detection scheme based on the ensemble of discriminant classifiers. Computers and Electrical Engineering. **86**, 106742 (2020).

[21] Gu J., Lu S. An effective intrusion detection approach using SVM with naΓÏve Bayes feature embedding. Computers & Security. **103**, 102158 (2021).

[22] Jia H., Liu J., Zhang M., He X., Sun W. Network intrusion detection based on IE-DBN model. Computer Communications. **178**, 131–140 (2021).

[23] Jeatrakul P., Wong K. K. W., Fung L. C. C. Classification of Imbalanced Data by Combining the Complementary Neural Network and SMOTE Algorithm. Neural Information Processing. Models and Applications. 152–159 (2010).

[24] Yan B., Han G. LA-GRU: Building Combined Intrusion Detection Model Based on Imbalanced Learning and Gated Recurrent Unit Neural Network. Security and Communication Networks. **2018**, 6026878 (2018).

[25] Abdulhammed R., Faezipour M., Abuzneid A., Abumallouh A. Deep and Machine Learning Approaches for Anomaly-Based Intrusion Detection of Imbalanced Network Traffic. IEEE Sensors Letters. **3** (1), 7101404 (2019).

[26] Chuang P.-J., Wu D. Y. Applying Deep Learning to Balancing Network Intrusion Detection Datasets. 2019 IEEE 11th International Conference on Advanced Infocomm Technology (ICAIT). 213–217 (2019).

[27] Bedi P., Gupta N., Jindal V. Siam-IDS: Handling class imbalance problem in Intrusion Detection Systems using Siamese Neural Network. Procedia Computer Science. **171**, 780–789 (2020).

[28] Hafiza Anisa A., Anum H., Narmeen Zakaria B. Network intrusion detection using oversampling technique and machine learning algorithms. PeerJ Computer Science. **8**, e820 (2022).

[29] Zhang Y., Liu Q. On IoT intrusion detection based on data augmentation for enhancing learning on unbalanced samples. Future Generation Computer Systems. **133**, 213–227 (2022).

[30] Andresini G., Appice A., Rose L. D., Malerba D. GAN augmentation to deal with imbalance in imaging-based intrusion detection. Future Generation Computer Systems. **123**, 108–127 (2021).

[31] Kumar V., Sinha D. Synthetic attack data generation model applying generative adversarial network for intrusion detection. Computers & Security. **125**, 103054 (2023).

[32] Yang Y., Gu Y., Yan Y. Machine Learning-Based Intrusion Detection for Rare-Class Network Attacks. Electronics. **12** (18), 3911 (2023).

[33] Liu L., Wang P., Lin J., Liu L. Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning. IEEE Access. **9**, 7550–7563 (2021).

[34] Balla A., Habaebi M. H., Elsheikh E. A. A., Islam M. R., Suliman F. M. The Effect of Dataset Imbalance on the Performance of SCADA Intrusion Detection Systems. Sensors. **23** (2), 758 (2023).

[35] Talukder M. A., Hasan F., Islam M., Uddin M. A., Akhter A., Yousuf M., Alharbi F., Moni M. A. A dependable hybrid machine learning model for network intrusion detection. Journal of Information Security and Applications. **72**, 103405 (2023).

[36] Lavanya T., Rajalakshmi K. Heterogenous ensemble learning driven multi-parametric assessment model for hardware Trojan detection. Integration. **89**, 217–228 (2023).

[37] Douzas G., Bacao F. Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE. Information Sciences. **501**, 118–135 (2019).

[38] Zhu M., Ye K., Wang Y., Xu C.-Z. A Deep Learning Approach for Network Anomaly Detection Based on AMF-LSTM. Network and Parallel Computing. 137–141 (2018).

[39] Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P. SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research. **16** (1), 321–357 (2002).

[40] Onah J. O., Abdulhamid S. M., Abdullahi M., Hassan I. H., Al-Ghusham A. Genetic Algorithm based feature selection and Naïve Bayes for anomaly detection in fog computing environment. Machine Learning with Applications. **6**, 100156 (2021).

[41] Gupta N., Jindal V., Bedi P. LIO-IDS: Handling class imbalance using LSTM and improved one-vs-one technique in intrusion detection system. Computer Networks. **192**, 108076 (2021).

[42] Li X., Yi P., Wei W., Jiang Y., Tian L. LNNLS-KH: A Feature Selection Method for Network Intrusion Detection. Security and Communication Networks. **2021**, 8830431 (2021).

XIDINTV: виявлення вторгнень в незбалансований мережевий трафік на основі XGBoost за допомогою варіаційного автокодера

Абдулганію О. А.[1], Айт Чаухт Т.[1], Еззіяні М.[2], Бенсліман М.[3]

[1] *Університет Euromed у Фесі, UEMF, Марокко*
[2] *Математична лабораторія та застосування, Факультет науки та технологій Університету Абдельмалека Ессааді, Танжер, Марокко*
[3] *Лабораторія наук, інженерії та управління, Університет Сіді Мохамеда Бен Абделла, Марокко*

У мережах, які характеризуються незбалансованим трафіком, виявлення зловмисних кібератак становить значну проблему через їх здатність плавно поєднуватися зі звичайними обсягами даних. Це створює серйозну перешкоду для систем виявлення вторгнень у мережу (NIDS), які прагнуть до точної та своєчасної ідентифікації. Дисбаланс у звичайних і атакуючих даних в поєднанні з різноманітністю категорій атак ускладнює виявлення вторгнень. Це дослідження пропонує новий підхід до вирішення цієї проблеми шляхом поєднання екстремального градієнтного посилення з варіаційним автокодером (XIDINTV). Методологія зосереджена на виправленні класового дисбалансу шляхом генерації різноманітних даних атаки рідкісного класу, зберігаючи при цьому схожість з оригінальними зразками. Це покращує здатність класифікатора розпізнавати відмінності під час навчання, покращуючи ефективність класифікації. Оцінки на основі наборів даних NSL-KDD і CSE-CIC-IDS2018 демонструють ефективність XIDINTV, особливо в порівнянні з технікою вибірки SMOTE і традиційними моделями класифікації, при цьому Xtreme Gradient Boosting відмінно виявляє рідкісні випадки атак трафіку.

Ключові слова: *система виявлення вторгнень; дисбаланс мережевого трафіку; екстремальне градієнтне посилення; варіаційний автокодувальник; виявлення аномалії; доповнення даних.*