



Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning

Ahmed Abdelkhalek¹ · Maggie Mashaly¹

Accepted: 20 January 2023 / Published online: 12 February 2023
© The Author(s) 2023

Abstract

Network intrusion detection systems (NIDS) are the most common tool used to detect malicious attacks on a network. They help prevent the ever-increasing different attacks and provide better security for the network. NIDS are classified into signature-based and anomaly-based detection. The most common type of NIDS is the anomaly-based NIDS which is based on machine learning models and is able to detect attacks with high accuracy. However, in recent years, NIDS has achieved even better results in detecting already known and novel attacks with the adoption of deep learning models. Benchmark datasets in intrusion detection try to simulate real-network traffic by including more normal traffic samples than the attack samples. This causes the training data to be imbalanced and causes difficulties in detecting certain types of attacks for the NIDS. In this paper, a data resampling technique is proposed based on Adaptive Synthetic (ADASYN) and Tomek Links algorithms in combination with different deep learning models to mitigate the class imbalance problem. The proposed model is evaluated on the benchmark NSL-KDD dataset using accuracy, precision, recall and F-score metrics. The experimental results show that in binary classification, the proposed method improves the performance of the NIDS and outperforms state-of-the-art models with an achieved accuracy of 99.8%. In multi-class classification, the results were also improved, outperforming state-of-the-art models with an achieved accuracy of 99.98%.

Keywords Class imbalance · Cybersecurity · Deep convolutional neural networks · Intrusion detection · Long-short-term memory

Ahmed Mashaly and Maggie Mashaly contributed equally to this work.

✉ Ahmed Abdelkhalek
ahmed.abdelhaleem@guc.edu.eg

Maggie Mashaly
maggie.ezzat@guc.edu.eg

¹ Networks Department, German University in Cairo, El-Tagamoa El-Khames, Cairo, Egypt

1 Introduction

1.1 Background and motivation

With the fast development of information technologies like smart devices, internet of things (IoT), cloud computing and big data, etc., the number of connected devices to the internet increased more than ever. Networks are getting larger and the risk of cyberattacks increases as networks become more difficult to monitor. A cyberattack begins with reconnaissance of the target and concludes with exploiting weaknesses to complete a malicious task [1]. Those cyberattacks cause an intrusion on the system which is defined as unauthorized access to the system resulting in a compromise in the confidentiality, integrity and availability (CIA) of security mechanisms of computer or network resources [2]. Many new cyberattacks have emerged in recent years like distributed denial of service or denial of service, brute force, botnet, cross-site scripting, etc. [3]. Those cyberattacks created a concern more serious than ever in cybersecurity. Many cloud servers and hosts have been under attack and have become botnets and Bitcoin Trojans malware vectors. According to the Internet Security Threat Report (ISTR), malware is found in one out of every thirteen web queries. Spam in e-mail has risen to more than 55%, internet dangers have risen to 46%, and ransomware has grown to 46% [4], incurring cyberattacks damages costing \$200,000 in 2019 according to CNBC [5]. Because of this expansion of networks and cyberattacks, a system to detect those attacks and provide network security is essential, namely intrusion detection systems (IDS).

An IDS, in its most basic form, is a software that monitors various sources and detects an intrusion on a system. IDS has proved to be an effective approach in detecting intrusions and has caught the attention of many researchers. It has the ability to detect known and unknown threats and intrusions by monitoring traffic data in computer systems and issuing alerts when these threats are detected [6]. According to the data sources monitored, IDSs are classified into Host IDS (HIDS) and Network IDS (NIDS). HIDS monitors data from logs, system calls, etc., but does not monitor network traffic, and thus, it can detect internal attacks not involving the network [7]. NIDS, on the other hand, collects and monitors data directly from the network through network monitoring equipment such as switches, routers and other network devices and thus can detect many types of network attacks.

According to the detection approach, IDSs are classified into misuse and anomaly detection. In misuse detection (also known as signature-based detection), any suspicious access is compared to a database of all known attacks, and the intrusion is detected based on that. This detects any previously known attacks successfully, but fails in detecting novel day-zero attacks. In addition, the database of known attacks should be regularly updated in order to keep up with the ever-increasing new attacks. On the other hand, anomaly-based intrusion detection system are capable of detecting both network and computer intrusions by monitoring system activity and classifying it as either normal or anomalous. The classification is based on heuristics or rules and not comparing to signatures. This has the theoretical potential of detecting day-zero new attacks, which has attracted the interest of many research fields.

Anomaly detection in intrusion detection systems was started in 1980 by Anderson [8] whose proposed method was system monitoring to detect anomalies. Since then, many techniques have been developed to implement anomaly-based network intrusion detection. Some of those techniques are computing based, data mining based, statistical based, machine learning, cognitive based or knowledge and user intention identification based, etc. [9]. One of the techniques used for anomaly detection is machine learning, which has shown ability to detect the differences between normal and anomaly traffic as shown in [10]. However, with the increase in network traffic and attack types, traditional machine learning models (shallow learners) are not keeping up with the needed performance [11]. To meet this large-scale demand, deep learning (DL) which is a branch of machine learning is now being used in NIDS. Studies have shown that deep learning outperforms traditional machine learning models in detecting anomalies due to its ability to extract information from massive amounts of data [12]. Deep learning-based approaches used in NIDS include deep neural networks (DNN), convolutional neural networks (CNN), long short-term memory (LSTM) and recurrent neural networks (RNN) [13]. A detailed literature review on machine learning- and deep learning-based NIDS is provided in Sect. 2.

1.2 Challenges

NIDSs based on deep learning models has proven to be an improvement upon machine learning models and achieving higher accuracy. However, they fail in detecting attacks with less traffic due to class imbalances in the benchmark datasets. Intrusion detection state-of-the-art benchmark datasets include class imbalances where the normal traffic is much more than the attacks' traffic (simulating real-world network traffic). Even among the different attack types, some attacks appear much more than others. This causes the NIDS to have difficulties detecting certain types of attacks and lowers the overall performance of it. Increased false alarm rate and decreased detection rate are signs of this lower performance. In recent NIDS researches, not enough attention was given toward the problem of imbalanced data despite its negative effect on the NIDS's accuracy of detecting attacks [14].

1.3 Contributions

This research aims to address the issue of class imbalances to improve the detection rate of minority classes using NIDS based on deep learning models. This is done by proposing a hybrid data resampling algorithm consisting of oversampling using Adaptive Synthetic Sampling (ADASYN) and undersampling using TomekLink. ADASYN is an oversampling technique that creates artificial samples of the minority classes, while TomekLink is an undersampling technique for cleaning up redundant samples that occur with ADASYN. A detailed explanation of the proposed method is included in Sect. 3. The main contributions of this paper can be summarized as follows:

- Addressing the class imbalances issue by using deep learning in combination with data resampling techniques. Both oversampling and undersampling techniques are applied to the dataset to increase the detection rate of minority classes. Oversampling is done using ADASYN which creates artificial data samples of the minority classes, while TomekLink is used to undersample any redundant data samples.
- Building four deep learning models to study the effect of data resampling on them and compare them with each other and with previous works. Those models include multi-layer perceptron (MLP), DNN, CNN and CNN-BLSTM. The effect of data resampling is observed on all models, and the improvement in false alarm rate and detection rate is noted.
- Testing the proposed data resampling method with the different deep learning models on the benchmark NSL-KDD dataset. The NSL-KDD is the most used NIDS dataset [15] while also suffering from the class imbalances problem.
- Evaluating the model performance by obtaining well-known performance metrics and comparing the models among each other's and with the related state-of-the-art previous models to show the superiority of the proposed models.

2 Literature review

With the increase in the need to detect network intrusions, the research into NIDS is gaining more and more attention. In addition, much progress has been achieved in both the machine learning and deep learning techniques and their applications in anomaly-based NIDS. In this section, we will review the most recent approaches in both techniques, focusing on the techniques applied on the benchmark NSL-KDD dataset presented in the literature which is divided into KDDTrain and KDDTest for training and testing, respectively.

2.1 Machine learning NIDS

Machine learning has been used as a technique to detect anomalies in network traffic as well as novel day-zero attacks without having its signature known before. The most common machine learning models used to detect anomalies in literature are support vector machines (SVM), decision trees (DT) and random forest (RF). The authors of [16] compare the accuracies of different machine learning models in classifying the KDDTest dataset into two classes (normal and anomaly). Reported results show that SVM achieved an accuracy of 69.52%, RF achieved an accuracy of 80.67%, and the J48 decision tree achieved an accuracy of 81.67%, all when tested on KDDTest. The authors of [17] proposed machine learning-based intrusion detection systems to classify attacks into normal, dos, probe, r2l and u2r. Their tests using DT algorithm achieved an accuracy of 75.22%, compared to 73.26% using SVM, and 62.73% using RF algorithm, all when tested on the KDDTest. In [18], an IDS was proposed based on preprocessing, feature selection, clustering and then classification. This was done by using fuzzy rule-based system to analyze the features

followed by a decision tree to select important features. The data are then clustered using K-means to minimize the number of the data sets used in training in order to bring down the complexities in the computation and processing. The SVM classifier is then used to categorize the intrusion on the network. This achieves an average accuracy of 97% on the KDD-NSL dataset. In [19] Firat et al. applied SVM, K-nearest neighbors (KNN) and DT algorithms on the NSL-KDD dataset and evaluated by splitting the KDDTrain dataset into train and test sets. Building on this work, Soheily et al. [20] proposed a hybrid NIDS based on K-means and RF(KM-RF) and evaluated the proposed algorithm on the NSL-KDD dataset. Similar work using an enhanced KNN algorithm along with local outlier factor (LOF) was done by authors of [21] and tested on the CICIDS2017 dataset of predicting zero day attacks with an accuracy of 92.74%.

Although machine learning techniques achieve good detection accuracies, they cannot face the problems in a real-world network environment. This is because traditional machine learning techniques rely heavily on feature engineering to extract information from network traffic [12]. On the other hand, deep learning can extract and learn features from the data due to their deep structure [22]. Thus, deep learning techniques are more suitable for massive amounts of data.

2.2 Deep learning NIDS

Deep learning techniques are typically used in more complex tasks like image recognition and natural language processing. This makes deep learning more suitable for a complex task like intrusion detection especially for intrusions that have not been seen before. Many researchers have studied the application of deep learning for intrusion detection and proved that it outperforms other techniques.

In [12], a recurrent neural network (RNN) model was proposed and evaluated on the NSL-KDD dataset. It achieved accuracies of 83.28% and 81.29% in binary and multi-class classification, respectively, and thus outperforming traditional machine learning models. In [13], long short-term memory (LSTM) layers were used to build a model and evaluated on the same dataset achieving a higher accuracy of 93.88%. The authors of [23] used a sequential feedforward neural network (SEQ-FFNN) to detect and classify attack packets and evaluated the model on the KDDTrain dataset achieving accuracies of 98.97% for Tanh activated layers and 99.59% for the Sigmoid activated model. In [24], a multi-layer perceptron (MLP) with particle swarm optimization algorithm (PSO) model was proposed (MLP-PSO) and was shown to achieve an accuracy of 83.27% on binary classification. The authors in [25] suggested a BAT-MC model for NIDS using NSL-KDD that outputs a 122-feature vector and uses a min-max scaler in the data preprocessing layer. The traffic data are then converted into traffic images in the multiple convolution layers stage, which contains three two-dimensional convolutional layers that extract spatial features, followed by the bidirectional long short-term memory (BLSTM) layer, which connects the forward and backward LSTM and learns the time-series features in the data packet. Each data packet can produce a packet vector, which forms a network flow vector. Then, to learn features on the network flow vectors and pay more attention

to key features, attention layers are used. These proposed enhancements achieved 84.25% accuracy in KDDTest in five category classification. The authors in [26] implemented a CNN2D model and achieved an accuracy of 86.95% on binary classification. Finally, in [27], a CNN model was used and evaluated after splitting the KDDTrain dataset to achieve an accuracy of 98.63 % which is the highest compared to other models. In [28], an intrusion detection system based on fusion convolutional neural network (FCNN) for feature extraction and stacked ensemble (SE) for classification was proposed. The FCNN uses a 1D CNN with a 2D CNN to extract features from the dataset. This method produced a new dataset of 256 features. The classification was then carried out by an SE learner where a combination of K-nearest neighbors (KNN), decision tree (DT) and Naive Bayesian (NB) was used. The experiments on the NSL-KDD dataset with tenfold validation resulted in an average accuracy of 98.9%. In [29], an IDS based on LSTMs and gated recurrent unit (GRU) called LTSMGRU is proposed. The steps of this IDS included feature selection using Pearson correlation method after scaling the data using MinMaxScaler then using the model to predict the attack. This was evaluated on the CICIDS2018 dataset by splitting it into train and test sets and testing with tenfolds. The highest average accuracy achieved was recorded to be 99.76%. Another IDS is proposed in [30], and it is based on gated recurrent neural network (GRU-RNN) on a software-defined network (SDN) called DeepIDS. This was tested on the NSL-KDD dataset and achieved an accuracy of 80.7% and 90% in binary detection for a DNN and GRU-RNN, respectively. In [31], feature selection was implemented using sequence forward selection (SFS) algorithm and decision tree (DT) model before anomaly detection. Anomaly detection was carried out by using RNN, LSTM and GRU. This IDS achieves the highest average accuracy of 92%.

Deep learning techniques show an obvious improvement over traditional machine learning models by achieving very high accuracies. However, those high accuracies achieved are due to the class imbalances in the NSL-KDD dataset. Most models are able to detect majority classes easily, which results in high accuracies, while failing to detect minority classes.

2.3 Class Imbalances

When a class is underrepresented in a dataset, this causes the dataset to be imbalanced. Detecting the minority class becomes a difficulty which lowers the performance of the intrusion detection system [32]. The NSL-KDD dataset has much more normal samples than the attack samples, just like in real-world network. This causes the class imbalances problem to appear and be one of the most common challenges in intrusion detection [11]. In [33], the performance of DT and RF was improved by using CatBoost with random oversampling and undersampling. This was evaluated on the CIC-IDS-2018 dataset and achieved an accuracy of 91.95% in multi-class classification. In [34], an improved NSGS-III called I-NSGA-III feature selection algorithm was proposed for solving the imbalance problem in NSL-KDD. A better detection rate was reported, but not a higher accuracy. In [35], random oversampling was used on the minority classes, and random undersampling was used on the

majority class in the NSL-KDD dataset to improve intrusion detection. However, in this work, random oversampling is known to cause overfitting [36], and only the accuracy was reported. In [37], two tree-based and one deep learning-based classifiers were tested with different sampling rates; it proved that sampling improves the detection of different classes. In [38], Zhang et al. propose SMOTE combined with edited nearest neighbors (SMOTE-ENN) and deep neural network (DNN) algorithm to evaluate on the NSL-KDD dataset. A cost-sensitive deep learning model was combined with ensemble algorithms to deal with the imbalance problem in [39]. An IDS consisting of three layers. The first layer uses cost-sensitive deep neural network to separate normal traffic from suspicious traffic. Suspicious traffic is then fed into an XGBoost model to classify into a majority class or a collection of minority classes. The collection of minority classes is then classified into a particular minority class using a random forest. This achieves high detection rates for both majority and minority attacks but it consumes heavy resources and takes a lot of time. A deep metric learning that combines autoencoders and triplet networks to detect attacks was proposed in [40]. The triplet network is trained to get the embedding vectors of the network flows and detect the presence of a malicious activity. However, this achieved good results mainly on binary classification of the NSL-KDD giving accuracy of 93.5%. In [41], seven machine learning models were tested on the NSL-KDD dataset for binary classification including k-means clustering, k-NN, RF, IF, b-XGBoost, DNN and CNN. Both of the b-XGBoost and DNN models were the top performers. For multi-class classification, two classifiers, namely m-XGBoost and Siamese-NN, were evaluated, and m-XGBoost was chosen. The model was tested on the NSL-KDD dataset. The accuracy was reported to be 80% in binary classification. In multi-class, an average F-score is 70%. In [42], oversampling was done using SMOTE, while the oversampled data were undersampled using TomekLink to remove redundant data samples. This was evaluated on many datasets including the NSL-KDD with a stratified tenfold cross-validation, while the classification was carried out with a LSTM model. This method achieved an accuracy of 99%.

Related works in NIDS that consider the imbalance problem highly improve the performance of NIDS compared to others that do not address imbalances problem. However, few of the related works evaluated their methods on NSL-KDD dataset; also to the best of our knowledge, none of the studies have analyzed the combination of ADASYN and TomekLink algorithms together for undersampling and oversampling, respectively.

3 Proposed method

In this paper, it is aimed to deal with the NSL-KDD dataset imbalances problems in order to achieve not only good accuracies but also improved detection rate of the minority classes. This can be achieved by mitigating the imbalanced classes' problem and then proceeding with intrusion detection.

To deal with the imbalance problem, a data preprocessing method is proposed. This method consists of oversampling the minority classes using Adaptive Synthetic Sampling (ADASYN) followed by undersampling using TomekLink to remove any

redundant data samples generated by ADASYN. ADASYN basic idea is using a weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data are generated for minority class examples that are harder to learn compared to those minority examples that are easier to learn [43]. Some of the generated samples may overlap with the majority classes, and that is why undersampling is used after oversampling. Undersampling is done using TomekLinks which deletes some of the majority class to remove redundancies. This proposed method is shown to improve the performance of NIDS, and to the best of our knowledge, ADASYN + TomekLinks method has not been tested in the previous literature. After the resampling method, the classification is then carried out by more than one deep learning model, and the performances of the different models are compared. After the data resampling, the models are trained on both binary and multi-class classification problems. The performance of the models is evaluated on the KDDTest dataset and a part of KDDTrain dataset, where in both cases, the test dataset remains unseen by the training model. A detailed explanation of the dataset, data preprocessing as well as the purposed architectures will be included in this section. A diagram of the purposed architecture is shown in Fig. 1.

3.1 Dataset description

The NSL-KDD is a publicly available dataset purposed by M. Tavallae in [44] as an improvement over the well-known KDD'99 dataset. It is available for researchers as a benchmark dataset to evaluate different intrusion detection methods. The NSL-KDD dataset contains KDDTrain and KDDTest which are partitioned into different difficulty levels. The NSL-KDD improved many of the shortcomings of the KDD'99 dataset by providing the following advantages:

- It does not include redundant records in the train set, so that the training would not be biased toward a certain record [44].
- It does not include duplicate records, which helps the training to be more accurate [44].

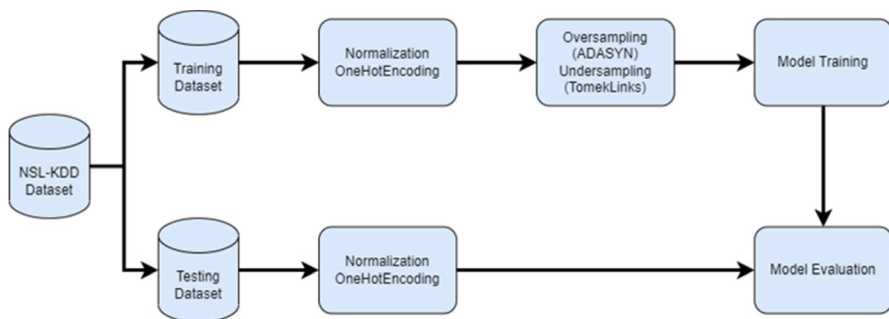


Fig. 1 Purposed architecture

- The selected records of each difficulty level is proportional to the original KDD'99 dataset in order not to affect the training [44].
- The number of records in the train and test sets is not very large, which makes it affordable to run experiments on the whole dataset without needing to select a portion of the dataset randomly. This enables a more accurate comparison between intrusion detection algorithms [44].

The dataset contains 41 attributes, with 1 column as the classification label. There are five labels, one label for the normal traffic, while the attacks are grouped into four different labels (DoS, Probing, R2L and U2R) according to the type of attack as shown in Table 1. An overview of each classification label is given below:

- Normal: This refers to normal traffic, and it is considered as the majority class of the dataset [45].
- Denial of Service Attack (DoS): This is an attack that aims to make a system extremely busy and consume its memory resources to hinder its ability to accept normal requests, preventing users' access to the system [45].
- Probing Attack (Probe): This attack is used to gather information about a network in order for its initiator to exploit the network's vulnerabilities [45].
- Remote to Local Attack (R2L): This is an attempt by an intruder to get access to a network as a user [45].
- User to Root Attack (U2R): This attack starts as a regular user account on the network and exploits vulnerabilities to gain root access to the system [45].

The 41 attributes of the dataset are shown in Table 2. There are three non-numerical attributes, and the remaining 38 attributes are all numerical [45]. The non-numerical attributes are the protocol-type which varies between 3 values, the Service which can have 70 different values and the Flag which has 11 different values as shown in Table 3 [45].

Table 1 NSL-KDD dataset attack types [45]

Class label	Attack type
DoS	mailbomb, neptune, apache2, back, land pod, processtable, teardrop, smurf, udpstorm worm
Probe	ipsweep, nmap, mscan, saint, portsweep satan
R2L	guess passwd, ftp write, httptunnel, multihop imap, named, phf, sendmail, Snmpgetattack spy, warezclient, snmpguess warezmaster, xsnoop, xlock
U2R	buffer overflow, loadmodule, perl, ps sqlattack, rootkit, xterm

Table 2 NSL-KDD dataset features [45]

Feature name	Feature type	Description
Duration	Numerical	Connection duration
Protocol-type	Non-numerical	Protocol used in the connection
Service	Non-numerical	Service used by destination network
Flag	Non-numerical	Status of the connection—normal or error
Src-bytes	Numerical	Number of data bytes transferred from source to destination in single connection
Dst-bytes	Numerical	Number of data bytes transferred from destination to source in single connection
Land	Numerical (Binary)	If source and destination IP addresses and port numbers are equal, then this variable takes value 1 else 0
Wrong-fragment	Numerical	Total number of wrong fragments in this connection
Urgent	Numerical	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated
Hot	Numerical	Number of hot indicators in the content such as entering a system directory, creating programs and executing programs
Num-failed-logins	Numerical	Count of failed login attempts
Logged-in	Numerical (Binary)	Login Status: 1 if successfully logged in and 0 otherwise
Numcompromised	Numerical	Number of compromised conditions
Root-shell	Numerical (Binary)	1 if root shell is obtained and 0 otherwise
Su-attempted	Numerical (Binary)	1 if “su root” command attempted or used and 0 otherwise
Num-root	Numerical	Number of root accesses or number of operations performed as a root in the connection
Num-filecreations	Numerical	Number of file creation operations in the connection
Num-shells	Numerical	Number of shell prompts
Num-access-files	Numerical	Number of operations on access control files
Num-outboundcmds	Numerical	Number of outbound commands in an ftp session
Is-host-login	Numerical (Binary)	1 if the login belongs to the hot list, i.e., root or admin and 0 otherwise
Is-guest-login	Numerical (Binary)	1 if the login is a “guest” login and 0 otherwise
Count	Numerical	Number of connections to the same destination host as the current connection in the past two seconds
Srv-count	Numerical	Number of connections to the same service (port number) as the current connection in the past two seconds

Table 2 (continued)

Feature name	Feature type	Description
Srv-error-rate	Numerical	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count
Srv-serror-rate	Numerical	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv-count (24)
Rerror-rate	Numerical	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)
Srv-rerror-rate	Numerical	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv-count (24)
Same-srv-rate	Numerical	The percentage of connections that were to the same service, among the connections aggregated in count (23)
Diff-srv-rate	Numerical	The percentage of connections that were to different services, among the connections aggregated in
Srv-diff-host-rate	Numerical	The percentage of connections that were to different destination machines among the connections aggregated in srvcnt (24)
Dst-host-count	Numerical	Number of connections having the same destination host IP address
Dst-host-srv-count	Numerical	Number of connections having the same port number
Dst-host-samesrv-rate	Numerical	The percentage of connections that were to the same service, among the connections aggregated in dst-host-count (32)
Dst-host-diff-srv-rate	Numerical	The percentage of connections that were to different services, among the connections aggregated in dst-host-count (32)
Dst-host-samestrc-port-rate	Numerical	The percentage of connections that were to the same source port, among the connections aggregated in dst-host-srvcnt (33)
Dst-host-srv-diffhost-rate	Numerical	The percentage of connections that were to different destination machines, among the connections aggregated in dsthost-srv-count (33)
Dst-host-serror-rate	Numerical	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst-host-srv-count (32)
Dst-host-srv-serror-rate	Numerical	The percent of connections that have activated the flag (4)s0, s1, s2 or s3, among the connections aggregated in dsthost-srv-count (33)

Table 2 (continued)

Feature name	Feature type	Description
Dst-host-terror-rate	Numerical	The percentage of connections that have activated the flag(4) REJ, among the connections aggregated in dst-host-srvcount(32)
Dst-host-srv-terror-rate	Numerical	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst-host-srvcount(33)

Table 3 Non-numerical features values [45]

Non-numerical features	Values
Protocol-type	tcp, udp, icmp
Service	aol, auth, bgp, courier, csnet-ns, ctf, daytime, discard, X11, Z39-50, domain, domain-u, echo, eco-i, ecr-i, efs, exec, finger, ftp, systat, vmnet, ftp-data, gopher, harvest, hostnames, http, http-2784, http-443,red-i, http-8001, imap4, IRC, iso-tsap, klogin, kshell, ldap, link, login, supdup, mtp, name, netbios-dgm, netbios-ns, netbios-ssn, netstat, nnspp, whois, nntp, ntp-u, other, pm-dump, pop-2, pop-3, printer, private, urp-i, uucp, uucp-path, remote-job, rje, shell smtp,sql-net, ssh, sunrpc, telnet, tftp-u, tim-i, time, urh-i
Flag	OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1, S2, S3, SF, SH.

The distribution of the labels in the train set and the test set is shown in Table 4. The distribution of the train and test sets can be further visualized in Fig. 2.

As shown in both Table 4 and Fig. 2, the NSL-KDD suffers from a very obvious class imbalance problem. This is intended to simulate real-network traffic where the normal traffic is much more than the attacks traffic. However, this causes the NIDS to achieve good accuracy by detecting the normal traffic and can negatively affect the detection rate of some attacks. The percentages of the normal, DoS, Probe, R2L and U2R traffic in the KDDTrain dataset are 53.45%, 36.45%, 9.25%, 0.78% and 0.04%, respectively. The presence of attacks is much lower than normal traffic especially both R2L and U2R attacks which are more difficult to detect [46]. This provides the motivation behind resampling the training dataset using ADASYN and TomekLinks.

Table 4 Distribution of samples for each class in NSL-KDD dataset

Class label	Train	Test
Normal	67,343	9710
DoS	45,927	7458
Probe	11,656	2421
R2L	995	2887
U2R	52	67
Total	125,973	22,543

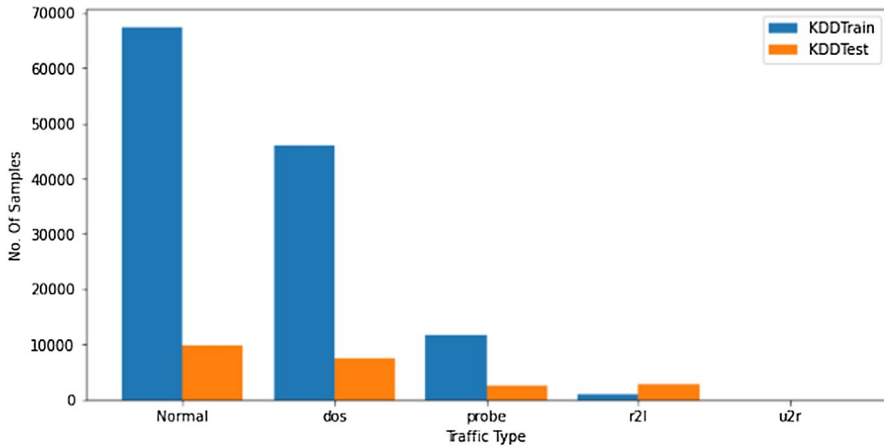


Fig. 2 Distribution of samples for each class in NSL-KDD dataset

3.2 Data preprocessing

Before using the data for the deep learning model, data preprocessing is essential to make it understandable and learnable by the model. The NSL-KDD dataset is a clean dataset without any empty or Not a Number (NaN) entries. Nan entries are not understood by a machine learning model and thus should be removed before training a model. So in this work, data preprocessing consists of two steps: encoding and normalization.

3.2.1 Encoding

Deep learning models only understand numeric values, that's why the non-numeric attributes have to be encoded in some way to be understood by the model. In this work, the three non-numeric attributes were converted into numerical ones using one-hot encoding. One-hot encoding is the most common algorithm for addressing multi-labeled data due to its simplicity [47]. One-hot encoding is converting each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. Each integer value is represented as a binary vector. All the values are zero, and the index is marked with a 1 as shown in Fig. 3 [47]. This transformation results in increasing the number of attributes from 41 to 122. The classification labels are also transformed using one-hot encoding. This converts the 1 column classification into five columns representing the normal class and the four attack types. This encoding is achieved with the OneHotEncoder class from the scikit-learn library [48]. Various encoding techniques were tested in this work, and one-hot encoding was chosen as it provided best results.


Type		Type	AA_Onehot	AB_Onehot	CD_Onehot
AA	Onehot encoding 	AA	1	0	0
AB		AB	0	1	0
CD		CD	0	0	1
AA		AA	0	0	0

Fig. 3 One-hot encoding [49]

3.2.2 Normalization

Scaling the data is an important part of data preprocessing in deep learning by mapping the values into a specific range. This process helps the deep learning model and speeds up the training process. Scaling can be achieved using normalization, where one of the most common normalizers is the MinMaxScaler from the scikit-learn library which was used in this work. The normalization is done by subtracting each value from the minimum value in its column and dividing by the range (Max value–Min value) as shown in Eq. 1. Several normalizers were tested in the process of this research; eventually, MinMaxScaler was chosen for providing best results.

$$x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)} [50]. \quad (1)$$

After applying one-hot encoding and normalization on the train dataset, two approaches are implemented. The first approach is to train on the entire KDDTrain dataset and use the KDDTest as the testing dataset. The second approach is to split the KDDTrain dataset into a 75%, 25% parts and use the 75% for training and 25% for testing. In both cases, the test dataset is a set of samples that have not been seen while training the model.

3.2.3 Class balancing

As shown in the dataset description subsection, the NSL-KDD suffers from the class imbalance problem. This appears in how the normal traffic compromises 53.45%, while the U2R traffic compromises 0.04% of the dataset. This problem creates a false high accuracy because the model will be trained to classify the majority class while ignoring the minority classes. Achieving high accuracies that does not correctly reflect the performance of the model is called an accuracy paradox [51]. This imbalance problem is well-known to be dealt with by oversampling and undersampling [52]. In this work, an advanced approach based on the work of [42] is proposed by oversampling the dataset using ADASYN instead of SMOTE and undersampling the output using TomekLinks with the addition of dropping the normal traffic in the multi-class classification case. This proposed data resampling method will be referred to as ADASYN + TomekLinks.

3.2.3.1 Dropping normal traffic In the case of multi-class classification, the normal traffic is dropped from both the train and test datasets. This is implemented to let the model focus more on classifying the attack type without confusing it with a normal sample. This is done under the assumption of having two-stage NIDS which will first identify an attack then identify its type. The distribution of the dataset samples after dropping normal traffic is shown in Fig. 4.

3.2.3.2 ADASYN Using ADASYN algorithm generates synthetic samples according to the level of difficulty in learning a specific minority class samples. Thus, more synthetic samples are generated for the minority classes that are relatively harder to learn due to their small count. It generates samples along the line segments between the k minority classes nearest neighbors [53]. The ADASYN algorithm can be further clarified as explained in Algorithm 1. However, ADASYN does not take into consideration that neighboring examples may come from other classes. This may create an overlap between classes that can be solved by undersampling. Although ADASYN was designed as an improvement to SMOTE, the literature of comparisons between them does not unanimously favor either of the two [53]. The distribution of the dataset samples after dropping normal traffic and oversampling is shown in Fig. 5.

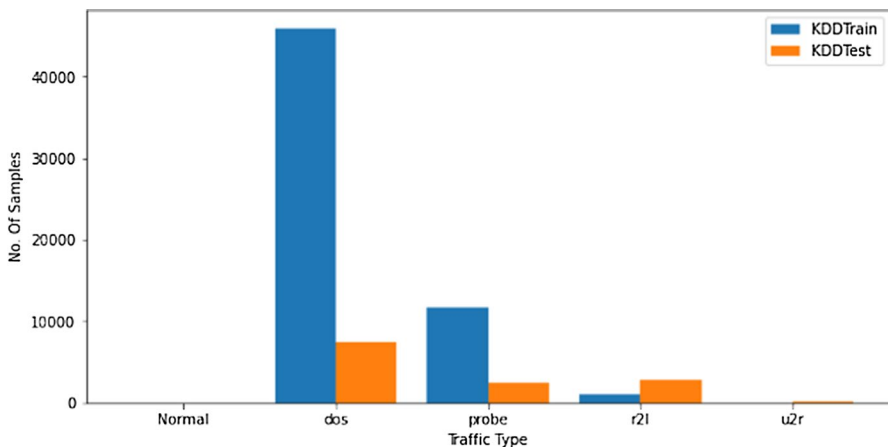


Fig. 4 Distribution of samples in each class in NSL-KDD dataset after dropping normal traffic

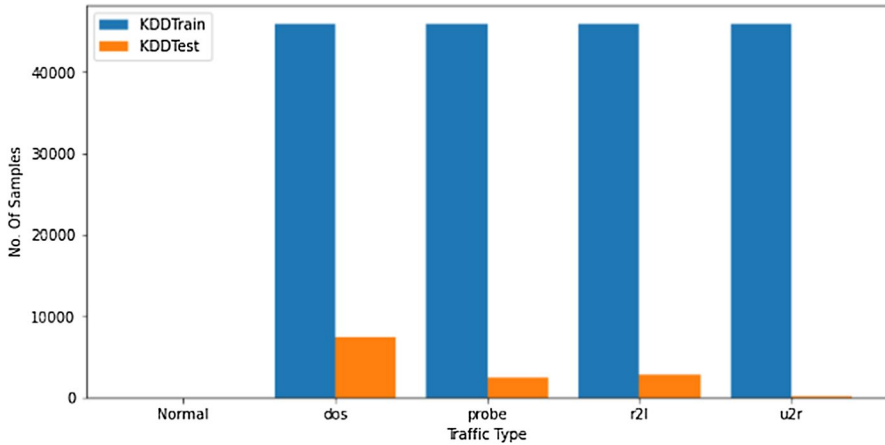


Fig. 5 Distribution of samples in each class in NSL-KDD dataset after dropping normal traffic and over-sampling

Algorithm 1 Algorithm - ADASYN

Require:

- (a) D_{tr} = Training imbalanced dataset with m samples;
- (b) m_s = Number of samples in minority class;
- (c) m_l = Number of samples in majority class;
- (d) d_{th} = Preset threshold for maximum tolerated degree of class imbalance ratio;
- (e) $\beta \in [0, 1]$ = A parameter used to specify the desired level after generation of the synthetic data;
- (f) δ_i = The number of examples in the K nearest neighbor that belong to any class other than the minority class;

Ensure: D_{tr} is balanced dataset

Calculate degree of class imbalance: $d_c = m_s/m_l$ where $d \in [0, 1]$

while $d_c < d_{th}$ **do**

1. $G_c = (m_l - m_s) * \beta$

2. For each sample in minority class, find K nearest neighbours based on Euclidean distance and calculate $r_i = \delta_i/K$

3. Normalize $\hat{r}_i = r_i / \sum_{i=1}^{m_s} r_i$

4. Calculate the number of samples to be generated $g_i = \hat{r}_i \times G_c$

5. For each minority class data example, generate g_i synthetic data examples.

end while

3.2.3.3 TomekLinks TomekLinks are a pair of samples belonging to different classes, but they are each other's nearest neighbor [54]. This is done by going over minority class samples and calculated the neighbor with the lowest Euclidean distance. If this nearest neighbor turns out to belong to the majority class, it is removed [54]. The presence of these pairs makes it ambiguous to distinct between classes so removing it helps in training the model [55].

3.2.3.4 ADASYN+TomekLinks In this paper, a combination of ADASYN and TomekLinks is used to oversample the data. TomekLinks is used after ADASYN to remove the noise that is generated with the oversampling. This removes neighbors of different classes to make the learning process easier. The proposed algorithm is shown in Algorithm 2. The distribution of the dataset samples after dropping normal traffic and oversampling+undersampling is shown in Fig. 6.

Algorithm 2 Algorithm - ADASYN+TomekLinks

- (a) Apply ADASYN until the number of minority samples increase with the desired proportion as shown in Algorithm 1;
 - (b) Loop over minority class samples;
 - (c) Find the nearest neighbor to the sample;
 - (d) Remove the nearest neighbor if it belongs to a majority class;
-

The distribution of the dataset in all preprocessing stages is shown in Table 5. It should be noted that the above-mentioned data resampling algorithms are only applied on the training dataset and not the test dataset. Applying those techniques

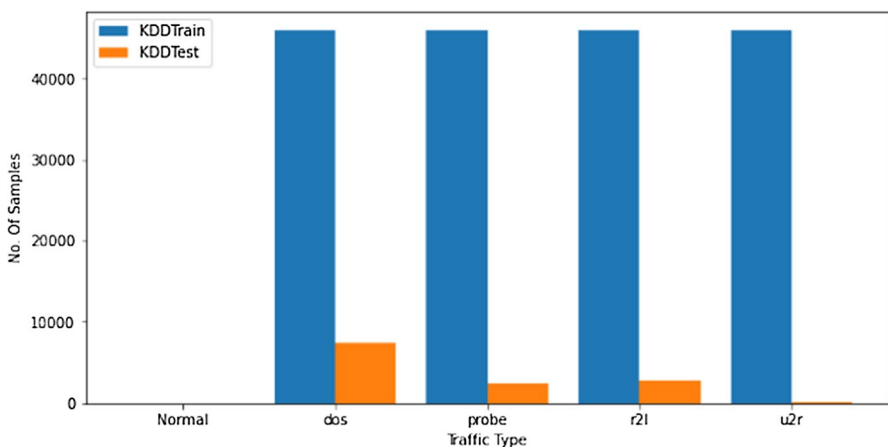


Fig. 6 Distribution of samples in each class in NSL-KDD dataset after dropping normal traffic and oversampling+undersampling

Table 5 Distribution of samples in each class in KDDTrain

Class label	Dropping normal	Oversampling	Oversampling+Undersampling
Normal	0	0	0
DoS	45,927	45,927	45,927
Probe	11,656	45,918	45,918
R2L	995	45,925	45,864
U2R	52	45,924	45,863
Total	125,973	183,694	183,572

on the test dataset would generate similar patterns in both training and test datasets, which will result in overestimated results [56].

3.3 Model architectures

Several model architectures were used in this work; we now describe those deep learning models. The models are based on MLP, DNN, CNN and CNN-BLSTM architectures. Those models were chosen for their proved high performance in many domains [57–59].

3.3.1 Multi-layer perceptron (MLP)

Multi-layer perceptron is a feed forward neural network which consists of three layers: the input layer, the output layer and the hidden layer. The input layer is the layer that receives the input signal which needs to be processed by the neural network. The output layer is the layer that performs the classification and produces an output. Any number of hidden layers can be included in between the input and output layers. They are the engine of the computations between the input and output layers. MLP is a feed forward neural network, meaning that the data flows from the input layer to the output layer. MLP neurons are trained using the back propagation algorithm. Some of the uses of MLP are pattern classification, recognition, prediction and approximation [60].

In this work, an enhanced MLP model is suggested. The model is the simple and fast one consisting of an input layer, an output layer and two hidden layers. After several trials and experiments, the number of neurons in each layer was found to provide the highest accuracy with values of 102 and 50, respectively. The activation function was chosen as the ReLU in both hidden layers. For the output layer, the activation function was either a sigmoid or a softmax depending on whether it is a binary or multi-class classification. No dropout was used in this model. The details of the model are shown in Table 6. Finally, the model was compiled with the Adam optimizer and categorical cross-entropy or binary cross-entropy as a loss function. The hyperparameters of the model are shown in Table 7.

The second model is of a more complex architecture than the first one, which will be referred to as the DNN (deep neural network) model. It consists of an

Table 6 MLP model layers

Block	Layers	Number of Neurons	Activation
Input block	Input layer	122	–
Hidden block 1	Dense layer	102	ReLU
Hidden block 2	Dense layer	50	ReLU
Output block	Output layer	1/4 ¹	Sigmoid/Softmax ¹

¹Depending on binary or multi-class classification

Table 7 MLP model hyperparameters

Parameter	Binary classifier	Multi-class classifier
Batch size	128	128
Learning rate	0.001/0.001 ¹	0.001/0.001 ¹
Standard deviation	0.001/0.1 ¹	0.01/0.1 ¹
Optimizer	Adam	Adam
Loss function	Binary cross-entropy	Categorical cross-entropy
Metric	Accuracy	Accuracy

¹Depending on splitting the KDDTrain or not

Table 8 DNN model layers

Block	Layers	Number of neurons	Activation
Input block	Input layer	122	–
Hidden block 1	Dense layer	1024	ReLU
	Dropout layer	0.01	–
Hidden block 2	Dense layer	768	ReLU
	Dropout layer	0.01	–
Output block	Output layer	1/4 ¹	Sigmoid/Softmax ¹

¹Depending on binary or multi-class classification

input layer, an output layer and two hidden layers. After several experiments, the number of neurons in each hidden layer was found to be 1024 and 768. The activation function was chosen to be ReLU function for all the hidden layers. A dropout layer was used after each hidden layer as well, with a probability of 0.01. For the output layer, the activation function was either a sigmoid or a softmax depending on whether it is a binary or multi-class classification. The details of the model are shown in Table 8. Finally, the model was compiled with the Adam optimizer and categorical cross-entropy or binary cross-entropy as a loss function. The hyperparameters of the model are shown in Table 9.

Table 9 DNN model hyperparameters

Parameter	Binary classifier	Multi-class classifier
Batch size	128	128
Learning rate	0.0003/0.001 ¹	0.001/0.001 ¹
Standard deviation	1/0.1 ¹	1/0.1 ¹
Optimizer	Adam	Adam
Loss function	Binary cross-entropy	Categorical cross-entropy
Metric	Accuracy	Accuracy

¹Depending on splitting the KDDTrain or not

3.3.1.1 Convolutional neural networks (CNN) The convolutional neural network (CNN) is a deep learning model widely used in computer vision. CNN is composed of a convolutional layer, pooling layer and dense layer. While the two-dimensional CNN has been successfully applied in image recognition, the one dimensional CNN (1D CNN) is more suitable for sequence data [61].

In this work, we use a 1D CNN model to implement an intrusion detection system. We design a model with seven layers, including the input and output layers. There are three 1D convolutional layers and two fully connected layers for the classification. The three convolutional layers are followed by a 1D max pooling layer to reduce the spatial size, thus reducing complexity and avoiding overfitting. Each layer is followed by a 10% dropout layer. The number of convolutional filter is 62, 62 and 124 with a kernel size of 2, 4 and 8. The max pooling layers has a pool size of 2, 4 and 8 with a stride of 1. The two fully connected layers consist of 256 and five neurons. All of those values were found after experiments and trials. The details of the model are shown in Table 10. Finally, the model was compiled with an Adam optimizer and categorical cross-entropy or binary cross-entropy as a loss function. The hyperparameters of the model are shown in Table 11.

3.3.1.2 Recurrent neural networks (RNN) Recurrent neural networks (RNNs) are a class of neural networks which can deal with sequential data [62]. They extend the capabilities of a traditional neural network by having a looped back connection in the hidden layers that is able to reuse data from previous times. A major problem that hinders the learning process of RNNs is vanishing and exploding gradients. This led to developing long short-term memory (LSTM) [63]. The LSTM model is proposed by Hochreiter et al. [63] to solve the vanishing gradient problem of RNN. It is composed of a cell, an input gate, output gate and forget gate. Bidirectional long short-term memory (BLSTM) is a more advanced application of LSTM where there are two LSTMs; one that takes the input in the forward direction and the other in the backward direction. This is done so that the BLSTM output can depend on both next and previous time steps.

In this work, we use a CNN-BLSTM model to implement an intrusion detection system. We design a model with six layers, including the input and output layers. There are three 1D convolutional layers and a BLSTM layer. The three

Table 10 CNN model layers

Block	Layers	Layer size	Activation	Kernel	Stride
Input block	Input layer	122	–	–	–
Hidden block 1	1D CNN layer	62	ReLU	2	1
	1D MaxPooling layer	2	–	–	1
	Dropout layer	0.1	–	–	–
Hidden block 2	1D CNN layer	62	ReLU	4	1
	1D MaxPooling layer	4	–	–	1
	Dropout layer	0.1	–	–	–
Hidden block 3	1D CNN layer	124	ReLU	8	1
	1D MaxPooling layer	8	–	–	1
	Dropout layer	0.1	–	–	–
Hidden block 4	Dense layer	256	ReLU	–	–
	Dropout layer	0.1	–	–	–
Hidden block 5	Dense layer	5	ReLU	–	–
	Dropout layer	0.1	–	–	–
Output block	Output layer	1/4 ¹	Sigmoid/Softmax ¹	–	–

¹Depending on binary or multi-class classification

Table 11 CNN model hyperparameters

Parameter	Binary classifier	Multi-class classifier
Batch size	128	128
Learning rate	0.1/0.001 ¹	0.001/0.001 ¹
Standard deviation	0.1/0.001 ¹	1/0.001 ¹
Optimizer	Adam	Adam
Loss function	Binary cross-entropy	Categorical cross-entropy
Metric	Accuracy	Accuracy

¹Depending on splitting the KDDTrain or not

convolutional layers are followed by a 1D max pooling layer to reduce the spatial size, thus reducing complexity and avoiding overfitting. Each layer is followed by a 10% dropout layer. The number of convolutional filter is 40, 60 and 80 with a kernel size of 2, 3 and 4. The maxpooling layers have a pool size of 2, 3 and 4 with a stride of 1. All of those values were found after experiments and trials. The details of the model are shown in Table 12. Finally, the model was compiled with an Adam optimizer and categorical cross-entropy or binary cross-entropy as a loss function. The hyperparameters of the model are shown in Table 13.

Table 12 CNN-BLSTM model layers

Block	Layers	Layer size	Activation	Kernel	Stride
Input block	Input layer	122	–	–	–
Hidden block 1	1D CNN layer	40	Tanh	2	1
	1D MaxPooling layer	2	–	–	1
	Dropout layer	0.1	–	–	–
Hidden block 2	1D CNN layer	60	Tanh	3	1
	1D MaxPooling layer	3	–	–	1
	Dropout layer	0.1	–	–	–
Hidden block 3	1D CNN layer	80	Tanh	4	1
	1D MaxPooling layer	4	–	–	1
	Dropout layer	0.1	–	–	–
Hidden block 4	Forward BLSTM layer	50	Tanh	–	–
	Backward BLSTM layer	50	Tanh	–	–
Output block	Output layer	1/4 ¹	Sigmoid/Softmax ¹	–	–

¹ Depending on binary or multi-class classification

Table 13 CNN-BLSTM model hyperparameters

Parameter	Binary classifier	Multi-class classifier
Batch size	128	128
Standard deviation	0.1/0.001 ¹	1/0.001 ¹
Optimizer	Adam	Adam
Loss function	Binary cross-entropy	Categorical cross-entropy
Metric	Accuracy	Accuracy

¹ Depending on splitting the KDDTrain or not

4 Experiments and results

In this section, several experiments were conducted to evaluate the performance of the proposed models with the data resampling method ADASYN+TomekLink. Firstly, the evaluation metrics are introduced and defined. Secondly, the performance of the different proposed models is compared with and without data resampling and also compared with each other's performances and with some state-of-the-art models in intrusion detection. The experiments show that our model outperforms state-of-the-art models in anomaly detection.

4.1 Experiment setup

The models were implemented using TensorFlow and Keras in a Google Collab platform. The hardware of the experiments is a Windows 10 with an Nvidia GeForce

GTX 1050. All the data resampling is done on the training set only. The test dataset was a set of samples never seen by the model, either the KDDTrain dataset or 25% of the KDDTrain dataset. The training of the models was done using 500 epochs with an early stopping condition monitoring val_accuracy metric to maximize with a patience of 20.

4.2 Evaluation metrics

A commonly used table in evaluating supervised machine learning models is the confusion matrix. The confusion matrix is a table that shows information about the predicted classes vs the actual classes [64]. It helps in calculating several metrics to evaluate the performance of the model using some terminologies. Some of the confusion matrix terminologies are:

- True Positive (TP): The data instances correctly predicted to be positive.
- False Negative (FN): The data instances wrongly predicted to be negative.
- True Negative (TN): The data instances correctly predicted to be negative.
- False Positive (FP): The data instances wrongly predicted to be positive.

The first metric used and the most intuitive one is accuracy, which can be calculated as Eq. 2

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} [28]. \quad (2)$$

When data are unbalanced, accuracy does not provide the most optimal evaluation of the model. Therefore, in addition to accuracy, other metrics are often used to evaluate the model. Those metrics are Precision, Recall and the F-score. The precision is the number of true positive results divided by the number of all positive results, including those not identified correctly also known as positive predictive value. It is calculated as shown in Eq. 3. The recall is the number of true positive results divided by the number of all samples that should have been identified as positive and is computed using Eq. 4. And the F-score is the harmonic mean of the precision and recall which is calculated according to Eq. 5 [65].

$$Precision = \frac{TP}{TP + FP} [28]. \quad (3)$$

$$Recall = \frac{TP}{TP + FN} [28]. \quad (4)$$

$$Fscore = \frac{2 * precision * recall}{precision + recall} [65]. \quad (5)$$

According to the metrics definitions, the objective here is to maximize accuracy, recall, precision and F-score.

Table 14 MLP performance metrics in binary classification

Metric	NoSplit				Split			
	Acc. ¹	Prec. ²	Recall	F-score	Acc. ¹	Prec. ²	Recall	F-score
No Resampling	77.2%	0.67	0.92	0.77	99.7%	0.99	0.99	0.99
Over	81.4%	0.73	0.90	0.80	99.8%	0.99	0.99	0.99
Over+ Under	85.1%	0.76	0.95	0.84	99.8%	0.99	0.99	0.99
Random Sampling	82.6%	0.72	0.96	0.82	99.8%	0.98	0.98	0.98

¹Accuracy²Precision**Table 15** DNN performance metrics in binary classification

Metric	NoSplit				Split			
	Acc. ¹	Prec. ²	Recall	F-score	Acc. ¹	Prec. ²	Recall	F-score
No Resampling	84.1%	0.75	0.94	0.83	99.6%	0.99	0.99	0.99
Over	85.6%	0.77	0.93	0.84	99.6%	1	0.99	0.99
Over+ Under	86.1%	0.77	0.95	0.85	99.6%	0.99	0.99	0.99
Random Sampling	85.6%	0.77	0.94	0.84	99.7%	0.97	0.97	0.97

¹Accuracy²Precision

4.3 Experimental results

The conducted experiments are mainly divided into two parts. In the first part, the proposed models were trained on the KDDTrain dataset and tested on the KDD-test dataset. This part will be referred to as the NoSplit part. In the second part, the proposed models were trained on 75% of the KDDTrain dataset while being test on the remaining 25% of the KDDTrain dataset. This part will be referred to as the split part. All the proposed models were tested in the two parts on both binary and multi-class classification. The effect of data resampling is observed on all the proposed models, then their performance is compared to each other's. Moreover, the proposed models performances are compared to recent NIDS works in the literature.

The baseline performance of every model without any data resampling is compared with the performance of the same model using oversampling (ADASYN) and oversampling+undersampling (ADASYN+TomekLink). The performance is also compared with random over- and undersampling. The evaluation metrics of each of the ML, DNN, CNN and CNN-BLSTM models in the binary classification case are shown in Tables 14, 15, 16 and 17, respectively.

From Tables 14, 15, 16 and 17, the effect of the proposed ADASYN+TomekLinks algorithm can be observed on the binary classification. In most models, an improvement in all accuracy, precision, recall and F-score can be observed. Among the

Table 16 CNN performance metrics in binary classification

Metric	NoSplit				Split			
	Acc. ¹	Prec. ²	Recall	F-score	Acc. ¹	Prec. ²	Recall	F-score
No resampling	74.8%	0.72	0.96	0.82	99.7%	0.99	0.99	0.99
Over	82.3%	0.72	0.96	0.82	99.7%	0.99	0.99	0.99
Over+Under	93.3%	0.76	0.95	0.84	99.8%	0.99	0.99	0.99
Random sampling	75.7%	0.65	0.92	0.76	99.8%	0.98	0.98	0.98

¹Accuracy²Precision**Table 17** CNN-BLSTM performance metrics in binary classification

Metric	NoSplit				Split			
	Acc. ¹	Prec. ²	Recall	F-score	Acc. ¹	Prec. ²	Recall	F-score
No Resampling	80.0%	0.70	0.92	0.79	99.7%	0.99	0.99	0.99
Over	83.5%	0.74	0.94	0.82	99.5%	0.99	0.99	0.99
Over+ Under	82.2%	0.73	0.91	0.81	99.7%	1	0.99	0.99
Random sampling	78.8%	0.68	0.94	0.78	99.7%	0.97	0.97	0.97

¹Accuracy²Precision**Table 18** MLP performance metrics in multi-class classification

Metric	NoSplit				Split			
	Acc. ¹	Prec. ²	Recall	F-score	Acc. ¹	Prec. ²	Recall	F-score
No Resampling	83.3%	0.83	0.83	0.83	99.7%	0.99	0.99	0.99
Over	87.3%	0.87	0.87	0.87	99.9%	0.99	0.99	0.99
Over + Under	85.5%	0.85	0.85	0.85	99.9%	0.99	0.99	0.99
Random sampling	84.4%	0.84	0.84	0.84	99.9%	0.99	0.99	0.99

¹Accuracy²Precision

proposed models, the CNN model achieves the highest accuracy of 87.8% in the NoSplit case and 99.8% in the Split case.

Furthermore, the evaluation metrics of each of the ML, DNN, CNN and CNN-BLSTM models in the multi-class classification case are shown in Tables 18, 19, 20 and 21, respectively.

The effect of the proposed ADASYN+TomekLinks algorithm can be observed on the multi-class case from Tables 18, 19, 20 and 21. In most cases, an improvement in all accuracy, precision, recall and F-score can be observed with oversampling

Table 19 DNN performance metrics in multi-class classification

Metric	NoSplit				Split			
	Acc. ¹	Prec. ²	Recall	F-score	Acc. ¹	Prec. ²	Recall	F-score
No Resampling	85.7%	0.85	0.85	0.85	99.8%	0.98	0.98	0.98
Over	83.6%	0.83	0.83	0.83	99.9%	0.99	0.99	0.99
Over + Under	82.9%	0.82	0.82	0.82	99.9%	0.9	0.99	0.99
Random sampling	83.0%	0.83	0.83	0.83	99.7%	0.97	0.97	0.97

¹Accuracy²Precision**Table 20** CNN performance metrics in multi-class classification

Metric	NoSplit				Split			
	Acc. ¹	Prec. ²	Recall	F-score	Acc. ¹	Prec. ²	Recall	F-score
No Resampling	76.8%	0.76	0.76	0.76	99.7%	0.99	0.99	0.99
Over	77.8%	0.77	0.77	0.77	99.5%	0.99	0.99	0.99
Over+ under	81.8%	0.81	0.81	0.81	99.7%	1	0.99	0.99
Random sampling	81.3%	0.81	0.81	0.81	99.7%	0.97	0.97	0.97

¹Accuracy²Precision**Table 21** CNN-BLSTM performance metrics in multi-class classification

Metric	NoSplit				Split			
	Acc. ¹	Prec. ²	Recall	F-score	Acc. ¹	Prec. ²	Recall	F-score
No Resampling	85%	0.84	0.84	0.84	99.6%	0.96	0.96	0.96
Over	80.5%	0.80	0.80	0.80	99.9%	0.99	0.99	0.99
Over + Under	80.6%	0.80	0.80	0.80	99.9%	0.99	0.99	0.99
Random Sampling	82.8%	0.82	0.82	0.82	99.7%	0.97	0.97	0.97

¹Accuracy²Precision

only. This is due to dropping the normal traffic before the resampling process, which helps the model not get confused with the normal traffic and decreases the presence of any TomekLinks. Among the proposed models, the MLP model achieves the highest accuracy of 87.25% in the NoSplit case and 99.9% in the Split case.

4.4 Discussion

The experimental results show the following:

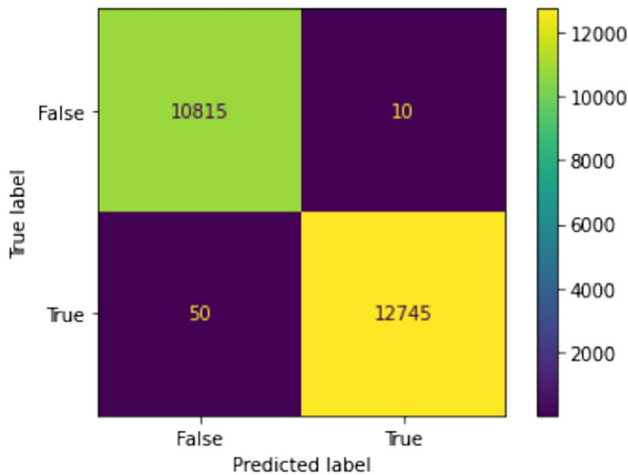


Fig. 7 CNN binary classification confusion matrix

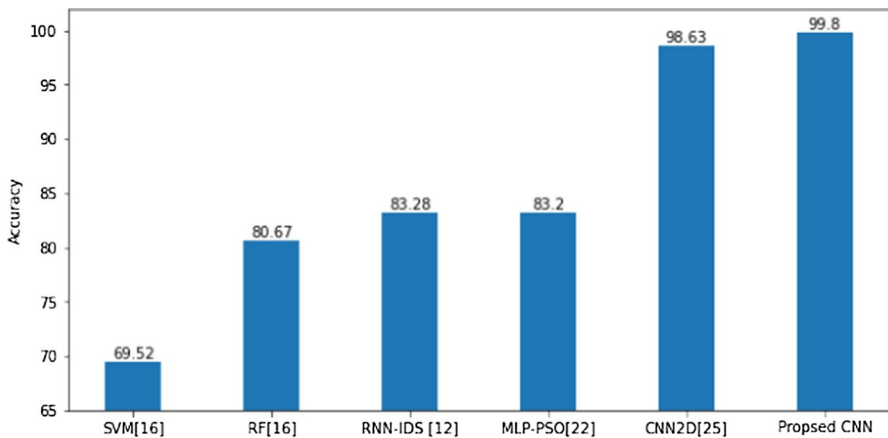


Fig. 8 Proposed CNN versus binary classifiers

1. Among the proposed binary models, the CNN model achieves the best accuracy, precision, recall and F-score. The CNN model extracts important features from the inputs which helps in detecting attacks. The un-sequential nature of the inputs makes RNN not useful. It was also found that the increased complexity of the model just causes overfitting in the case of DNN model. The confusion matrix is shown in Fig. 7 as well as the metrics in Table 16. In Fig. 8, its performance is compared to other recent NIDS in binary classification.
2. Among the multi-class models, the MLP model achieves the best accuracy, precision, recall and F-score. This shows that the added complexity to the model does not benefit the classification of the attack type. Increasing the complexity in the case of DNN causes overfitting. Because there is no sequential flow between the

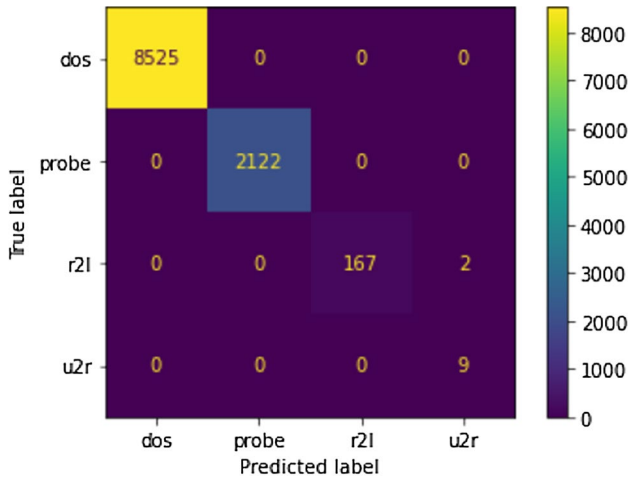


Fig. 9 MLP multi-class classification confusion matrix

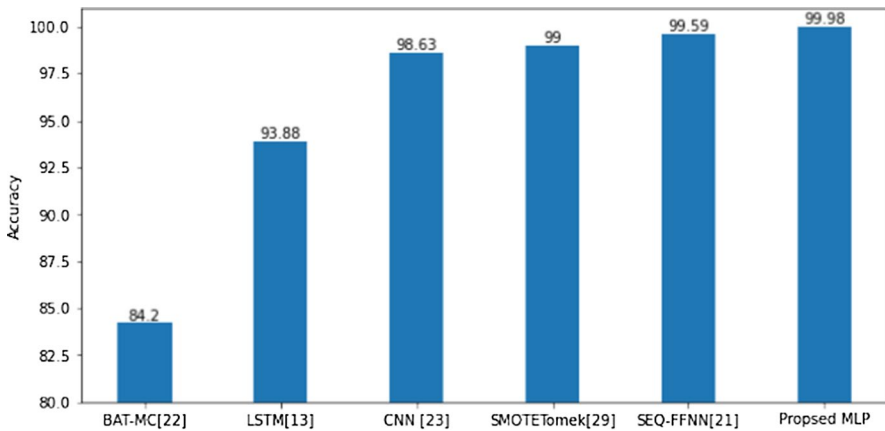


Fig. 10 Proposed MLP versus multi-class classifiers

data, RNNs do not increase the accuracy. A deeper look into the metrics is provided in Table 22, while the confusion matrix is shown in Fig. 9. From Table 22, it is shown that the MLP model successfully detects the minority attacks with good accuracy, precision and recall. Its performance is compared to recent multi-class NIDS in Fig. 10.

3. In both of the binary and multi-class classification detection, the proposed ADASYN+TomekLink or ADASYN only combined with deep learning improve the accuracy, precision and recall over the baseline models. This shows that data resampling techniques are a promising approach to solve the class imbalance problem in intrusion detection.

Table 22 MLP performance metrics in multi-class classification on different classes

Label	Acc. ¹	Prec. ²	Recall	F-score
dos	100%	1	1	1
probe	100%	1	1	1
r2 l	98.8%	1	0.98	0.98
u2r	100%	0.81	1	0.89

¹Accuracy²Precision

4. Dropping the normal traffic before training the attack type classification model improves the detection rate of minority classes. This is because the model no longer confuses an attack type sample with normal traffic sample.
5. A two-stage NIDS can be implemented using both binary and multi-class models. The first stage would detect the presence of an attack as opposed to normal traffic. After detecting the presence of an attack, the second stage would work on identifying the type of attack in order to be able to deal with it.

5 Conclusion

This paper proposes a framework to solve the class imbalance problem in the NSL-KDD dataset to improve the detection rate of minority class attacks. Data resampling techniques along with deep learning were proposed to solve the problem. The data were first oversampled using ADASYN then undersampled using TomekLinks to remove redundant data samples. This data resampling technique was used with four different deep learning models based on MLP, DNN, CNN and CNN-BLSTM architectures. Based on experimental results, ADASYN+TomekLinks technique was found to increase the detection rate of minority classes in comparison with no data resampling techniques. Moreover, in the binary classification, the proposed CNN model achieved an accuracy of 99.8% and detection rate of 99% which is better than state-of-the-art binary classifiers. Furthermore, in the multi-class classification, proposed MLP model achieved an accuracy of 99.9% and detection rate of 99% which is also better than state-of-the-art multi-class classifiers. These results provide a promising direction for NIDS and for improving the detection rate of minority classes in imbalanced datasets. This also opens up the possibility of implementing a two-stage NIDS to make use of the high accuracies in binary and multi-class classifiers using different models. Some of the future work recommendations will be below.

5.1 Future work

The proposed IDS in this paper provides promising results. However, the detection rates of minority classes can be improved even further, which is how this

work can be extended upon. By using different data resampling techniques and different combinations of oversampling and undersampling techniques and using different deep learning architectures, the detection rate can be further improved. This work can also be tested on different more imbalanced intrusion detection datasets like CIC-IDS2017 or UNSW-NB15, which both have more classes than the NSL-KDD dataset.

Author contributions All authors conceived the presented idea. All authors developed the theory and performed the computations. All authors verified the analytical methods. All authors discussed the results and contributed to the final manuscript.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB). Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data availability The NSL-KDD dataset is publicly available on <https://www.unb.ca/cic/datasets/nsl.html>, while the code is available for any entity of interest.

Declarations

Conflict of interest The authors declare that they have no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Conti M, Dargahi T, Dehghantanha A (2018) Cyber threat intelligence: challenges and opportunities, pp 1–6. https://doi.org/10.1007/978-3-319-73951-9_1
2. Faker O, Dogdu E (2019) Intrusion detection using big data and deep learning techniques. In: Proceedings of the 2019 ACM Southeast Conference. ACM SE '19, pp. 86–93. Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/3299815.3314439>
3. Kaur G, Habibi Lashkari A, Rahali A (2020) Intrusion traffic detection and characterization using deep image learning. In: 2020 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCoM/CyberSciTech), pp 55–62. <https://doi.org/10.1109/DASC-PiCom-CBDCoM-CyberSciTech49142.2020.00025>
4. Internet Security Threat Report. <https://docs.broadcom.com/doc/istr-23-2018-en>. Accessed: 2022-07-18
5. Cyberattacks now cost companies \$200,000 on average, putting many out of business. <https://www.cnbc.com/2019/10/13/cyberattacks-cost-small-companies-200k-putting-many-out-of-business.html>. Published: SUN, OCT 13 2019

6. Musa US, Chhabra M, Ali A, Kaur M (2020) Intrusion detection system using machine learning techniques: A review. In: 2020 International Conference on Smart Electronics and Communication (ICOSEC), pp 149–155
7. Khraisat A, Gondal I, Vamplew P, Kamruzzaman J (2019) Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity* 2. <https://doi.org/10.1186/s42400-019-0038-7>
8. Javaid A, Niyaz Q, Sun W, Alam M (2016) A deep learning approach for network intrusion detection system. *EAI Endorsed Trans Sec Safety*. <https://doi.org/10.4108/eai.3-12-2015.2262516>
9. Veeramreddy J, Prasad K (2019). Anomaly-Based Intrusion Detect Syst. <https://doi.org/10.5772/intechopen.82287>
10. Mahfouz AM, Venugopal D, Shiva SG (2019) Comparative analysis of ml classifiers for network intrusion detection. In: ICICT
11. Zhang H, Huang L, Wu CQ, Li Z (2020) An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset. *Comput Netw* 177:107315. <https://doi.org/10.1016/j.comnet.2020.107315>
12. Yin C, Zhu Y, Fei J, He X (2017) A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* 5:21954–21961. <https://doi.org/10.1109/ACCESS.2017.2762418>
13. Muhuri PS, Chatterjee P, Yuan X, Roy K, Esterline A (2020) Using a long short-term memory recurrent neural network (lstm-rnn) to classify network attacks. *Information*. <https://doi.org/10.3390/info11050243>
14. Bedi P, Gupta N, Jindal V (2020) Siam-ids: Handling class imbalance problem in intrusion detection systems using siamese neural network. *Proc Comput Sci* 171:780– 789. <https://doi.org/10.1016/j.procs.2020.04.085>. Third International Conference on Computing and Network Communications (CoCoNet'19)
15. Panigrahi R, Borah S (2018) A detailed analysis of cids2017 dataset for designing intrusion detection systems. *Int J Eng Technol* 7(3.24):479–482. <https://doi.org/10.14419/ijet.v7i3.24.22797>
16. Debicha I, Debatty T, Mees W, Dricot J (2021) Efficient intrusion detection using evidence theory. *CoRR arXiv: abs/2103.08585*
17. Dina AS, Siddique AB, Manivannan D (2022) Effect of balancing data using synthetic data on the performance of machine learning classifiers for intrusion detection in computer networks. *CoRR arXiv: abs/2204.00144*<https://doi.org/10.48550/arXiv.2204.00144>
18. Ammayappan S (2019) Enhanced soft computing approaches for intrusion detection schemes in social media networks. *J Soft Comput Paradigm*. 2019:69–79. <https://doi.org/10.36548/jscp.2019.2.002>
19. Kilincer IF, Ertam F, Sengur A (2021) Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Comput Netw* 188:107840. <https://doi.org/10.1016/j.comnet.2021.107840>
20. Soheily Khah S, Marteau P-F, Béchet N (2018) Intrusion detection in network systems through hybrid supervised and unsupervised machine learning process: A case study on the iscx dataset, pp 219– 226. <https://doi.org/10.1109/ICDIS.2018.00043>
21. Elmasri T, Samir N, Mashaly M, Atef Y (2020) Evaluation of cids2017 with qualitative comparison of machine learning algorithm. In: 2020 IEEE Cloud Summit, pp 46– 51. <https://doi.org/10.1109/IEEECloudSummit48914.2020.00013>
22. Xin Y, Kong L, Liu Z, Chen Y, Li Y, Zhu H, Gao M, Hou H, Wang C (2018) Machine learning and deep learning methods for cybersecurity. *IEEE Access* 6:35365–35381. <https://doi.org/10.1109/ACCESS.2018.2836950>
23. Aribisala A, Khan MS, Husari G (2021) Machine learning algorithms and their applications in classifying cyber-attacks on a smart grid network. In: 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp 0063– 0069. <https://doi.org/10.1109/IEMCON53756.2021.9623067>
24. Alhajar E, Maxwell P, Bastian N (2021) Adversarial machine learning in network intrusion detection systems. *Exp Syst Appl* 186:115782. <https://doi.org/10.1016/j.eswa.2021.115782>
25. Su T, Sun H, Zhu J, Wang S, Li Y (2020) Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset. *IEEE Access* 8:29575–29585. <https://doi.org/10.1109/ACCESS.2020.2972627>
26. Li Y, Xu Y, Liu Z, Hou H, Zheng Y, Xin Y, Zhao Y, Cui L (2020) Robust detection for network intrusion of industrial IoT based on multi-CNN fusion. *Measurements* 154:107450. <https://doi.org/10.1016/j.measurement.2019.107450>

27. Radhi M, Mohammed A (2022) A novel approach to network intrusion detection system using deep learning for sdn: Futuristic approach
28. Chen C, Song Y, Yue S, Xu X, Zhou L, Lv Q, Yang L (2022) Fcnn-se: An intrusion detection model based on a fusion CNN and stacked ensemble. *Appl Sci* 12(17):8601
29. Aldallal A (2022) Toward efficient intrusion detection system using hybrid deep learning approach. *Symmetry* 14(9). <https://doi.org/10.3390/sym14091916>
30. Tang TA, Mhamdi L, McLernon D, Zaidi SAR, Ghogho M, El Moussa F (2020) DeepIDS: deep learning approach for intrusion detection in software defined networking. *Electronics*. <https://doi.org/10.3390/electronics9091533>
31. Le T-T-H, Kim Y, Kim H (2019) Network intrusion detection based on novel feature selection model and various recurrent neural networks. *Appl Sci*. <https://doi.org/10.3390/app9071392>
32. Wang S, Yao X (2012) Multiclass imbalance problems: analysis and potential solutions. *IEEE Trans Syst Man Cybern Part B (Cybernetics)* 42(4):1119–1130. <https://doi.org/10.1109/TSMCB.2012.2187280>
33. Jumabek A, Yang SS, Noh YT (2021) CatBoost-based network intrusion detection on imbalanced CIC-IDS-2018 dataset. *Korean Soc Commun Commun J* 46(12):2191–2197
34. Zhu Y, Liang J, Chen J, Ming Z (2016) An improved nsga-iii algorithm for feature selection used in intrusion detection. *Knowl-Based Syst*. <https://doi.org/10.1016/j.knosys.2016.10.030>
35. Jiang J, Wang Q, Shi Z, Lv B, Qi B (2018) Rst-rf: A hybrid model based on rough set theory and random forest for network intrusion detection. In: *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*
36. Chawla N, Bowyer K, Hall L, Kegelmeyer W (2002) Smote: Synthetic minority over-sampling technique. *J Artif Intell Res (JAIR)* 16:321–357. <https://doi.org/10.1613/jair.953>
37. Alikhanov J, Jang R, Abuhamad M, Mohaisen D, Nyang D, Noh Y (2022) Investigating the effect of traffic sampling on machine learning-based network intrusion detection approaches. *IEEE Access* 10:5801–5823. <https://doi.org/10.1109/ACCESS.2021.313731>
38. Zhang X, Ran J, Mi J (2019) An intrusion detection system based on convolutional neural network for imbalanced network traffic. In: *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pp 456–460
39. Gupta N, Jindal V, Bedi P (2021) CSE-IDS: using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in Network-based intrusion detection systems. *Comput Secur* 112:102499. <https://doi.org/10.1016/j.cose.2021.10249>
40. Andresini G, Appice A, Malerba D (2021) Autoencoder-based deep metric learning for network intrusion detection. *Inf Sci* 569:706–727. <https://doi.org/10.1016/j.ins.2021.05.016>
41. Bedi P, Gupta N, Jindal V (2021) I-SiamIDS: an improved siam-IDS for handling class imbalance. *Network-Based Intrusion Detect Syst*. <https://doi.org/10.1007/s10489-020-01886-y>
42. Mbow M, Koide H, Sakurai K (2022) Handling class imbalance problem in intrusion detection system based on deep learning. *Int J Netw Comput* 12(2):467–492
43. He H, Bai Y, Garcia EA, Li S (2008) Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp 1322– 1328. <https://doi.org/10.1109/IJCNN.2008.4633969>
44. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the kdd cup 99 data set. In: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1– 6. <https://doi.org/10.1109/CISDA.2009.5356528>
45. Dhanabal L, Shantharajah S (2015) A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. *Int J Adv Res Comput Commun Eng* 4(6):446–452
46. Jeya PG, Ravichandran M, Ravichandran CS (2012) Efficient classifier for r2l and u2r attacks. *Int J Comput Appl* 45
47. Rodríguez P, Bautista MA, Gonzalez J, Escalera S (2018) Beyond one-hot encoding: Lower dimensional target embedding. *Image Vis Comput* 75:21–31
48. Jie L, Jiahao C, Xueqin Z, Yue Z, Jiajun L (2019) One-hot encoding and convolutional neural network based anomaly detection. *J Tsinghua Univ (Science and Technology)* 59(7):523–529
49. *Data Science in 5 Minutes: What is One Hot Encoding?* <https://www.educative.io/blog/one-hot-encoding>
50. Patro SG, Sahu D-KK (2015) Normalization: A preprocessing stage. In: *IARJSET*. <https://doi.org/10.17148/IARJSET.2015.2305>
51. Elmasry W, Akbulut A, Zaim AH (2019) Empirical study on multiclass classification-based network intrusion detection. *Comput Intell* 35:919–954

52. Bagui S, Li K (2021) Resampling imbalanced data for network intrusion detection datasets. *J Big Data*. <https://doi.org/10.1186/s40537-020-00390-x>
53. Brandt J, Lanzén E (2021) A comparative review of smote and adasyn in imbalanced data classification
54. Tomek I (1976) Two modifications of cnn. *IEEE Trans Syst Man Cybern* 6:769–772
55. Ma Y, He H (2013) Imbalanced learning: foundations, algorithms, and applications
56. Santos M, Soares J, Henriques Abreu P, Araujo H, Santos J (2018) Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches. *IEEE Comput Intell Mag* 13:59–76. <https://doi.org/10.1109/MCI.2018.2866730>
57. EL-Habil BY, Abu-naser SS (2022) Global climate prediction using deep learning. *J Theor Appl Inf Technol* 100(24)
58. Zhendong S, Jinping M (2022) Deep learning-driven MIMO: Data encoding and processing mechanism. *Phys Commun*. <https://doi.org/10.1016/j.phycom.2022.101976>
59. Xin Z, Chunjiang Z, Jun S, Kunshan Y, Min X (2022) Detection of lead content in oilseed rape leaves and roots based on deep transfer learning and hyperspectral imaging technology. *Spectroch Acta Part A Molecular Biomole Spectrosc*. <https://doi.org/10.1016/j.saa.2022.122288>
60. Abirami S, Chitra P (2020) Chapter fourteen - energy-efficient edge based real-time healthcare support system. In: Raj P, Evangeline P (eds.) *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*. *Advances in Computers*, vol. 117, pp. 339–368. Elsevier. <https://doi.org/10.1016/bs.adcom.2019.09.007>. <https://www.sciencedirect.com/science/article/pii/S0065245819300506>
61. Azizjon M, Jumabek A, Kim W (2020) 1d cnn based network intrusion detection with normalization on imbalanced data. In: *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, pp 218–224. <https://doi.org/10.1109/ICAIIIC48513.2020.9064976>
62. Vinayakumar R, Soman KP, Poornachandran P (2017) Evaluation of recurrent neural network and its variants for intrusion detection system (IDS). *Int J Inf Syst Model Des (IJISMD)* 8(3):43–63
63. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
64. Veeramreddy J, Prasad K (2019). Anomaly-Based Intrusion Detect Syst. <https://doi.org/10.5772/intechopen.82287>
65. Powers DMW (2011) Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *J Mach Learn Technol* 2(1):37–63

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.