Journal of Theoretical and Applied Information Technology
30th April 2024. Vol.102. No 8

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMIZED IRDN-IBDA: AN OPTIMAL FEATURE SELECTION AND RECOGNITION FOR NETWORK INTRUSION DETECTION SYSTEM

APPALARAJU GRANDHI [1*], SUNIL KUMAR SINGH[2]

School of Computer Science and Engineering, VIT-AP University, Vijayawada 522237, Andhra Pradesh, India;
Email 1*: rajugrandhi.21phd7068@vitap.ac.in, sunil.singh@vitap.ac.in
*Correspondence: rajugrandhi.21phd7068@vitap.ac.in

ABSTRACT

The demand for an efficient intrusion detection system has grown as attackers continue to create new attacks and network sizes expand. Recently, many techniques have been released for network intrusion detection systems (NIDSs). However, new threats are constantly developing and outside existing systems reach. The intrusion detection algorithms high error rate, significant dimensionality, false alarm rate, redundancy, meaningless data, and false negative rate now in use are only a few of the many issues with them. Given its exceptional performance in various detection and recognition tasks, we present a novel and efficient deep learning-based NIDS in this research. Initially in preprocessing data encoding and normalization are performed using raw input data. After preprocessing, the pre-processed data are fed into the feature extraction phase. The features are extracted by utilizing the SE-ResNeXt-101 approach. Then, the essential features are selected with the help of an Improved Binary Dandelion Algorithm (IBDA). The presented novel Improved Residual Dense Network (IRDN) is employed to identify attacks which enhance security and privacy inside the network framework. The lyrebird optimization technique is used to tune further the hyperparameters derived from the IRDN approach to increase performance. The Modified Generator GAN (MG-GAN) algorithm also solves the data imbalance issue. The research shows that the suggested technique outperforms current NIDS methods regarding assessment metrics. Additionally, this method is more suitable for complicated detection of network intrusion requirements.

Keywords: *Network Intrusion Detection Systems (Nidss), SE-Resnext-101, Improved Binary Dandelion Algorithm (IBDA), Modified Generator GAN (MG-GAN), Improved Residual Dense Network (IRDN).*

Symbols	Abbreviation
Network Intrusion Detection Systems	NIDS
Improved Binary Dandelion Algorithm	IBDA
Improved Residual Dense Network	IDRN
Modified Generator GAN	MG-GAN
Hierarchical Adversarial Attack	HAA
Graph Neural Network	GNN
Adaptive Synthetic	ADASYN
Long Short-Term Memory	LSTM
one-side selection	OSS
Deep Neural Network	DNN
n	the total number of specimens
x	the number of samples

i	Loop counter
Kullback–Leibler divergence	KL
Jensen–Shannon divergence	JSD
P_o and P_s	likelihood distributions
squeeze-and-excitation	SE
u_R	Parameter value for R filter
Core dandelions	CD
r and e	wilting and growth factors
Fe_{max}	maximum assessment time
f_{avg}	overall mean value of fitness of dandelions
SN	the total of all dandelions and seeds
$Scope(t)_i$	the radius of seeding in the i^{th} dimension
r	random integer
Global Average Pooling	GAP
Lyrebird optimization algorithm	LOA
Xi	possible outcome
lb and ub	lower and upper bounds
F_i	function objective

1. INTRODUCTION

The importance of network security has increased due to the swift development of big data, cloud computing, and other related innovations and data, along with the growing dependence of our everyday communications on networked services [1]. These factors have rendered networks indispensable. Every weakness or danger will impact the network as a whole. Traditional security measures like firewalls and encryption approaches struggle in an environment where attackers always create more sophisticated attacks [2, 3]. Additionally, to provide safe networks, cybersecurity researchers discovered how crucial it is to develop effective network intrusion detection systems (IDSs) [4]. Availability, confidentiality, and integrity are the goals of IDSs, which work to stop illegal access to networks, safeguard the data and communication systems within them, and above all identify suspected and unidentified risks and attacks with a low false alarm rate and high precision [5–7].

The two methods by which IDS operate are signature and anomaly-based methods [8]. Anomaly-based detection recognizes the assault based on peculiar user behaviour designs. In contrast, signature-based detection employs a known set of criteria or indications from the equipment's assault databases to determine if the behaviour is malicious or not [9, 10]. It may be possible to identify an attack if users engage in strange behaviour.

Behaviour is used in conjunction with various ML and data mining methods to provide the highest performance for identifying attack activities in anomaly-based detection systems (IDSs) [11–13]. These techniques shrewdly recognize and offer a new viewpoint on the various attacks plaguing worldwide computer networks [14, 15]. However, there are still several issues with applying the machine learning approach in IDSs. Developing a suitable model to describe datasets is the most significant obstacle facing machine learning techniques [16–18]. Numerous methods and learning strategies have been used to build models for efficient systems detecting intrusions. On the other hand, current models show poor accuracy, low false alarm rates, and low detection rates [19, 20].

The selection of the NIDS problem resulted from a thorough analysis of the Cybersecurity environment, which is defined by the rise in complexity and variety of cyber threats directed at network infrastructures. We have concentrated on NIDS because they protect network resources and identify malicious activity, unauthorized access, and unusual network traffic. It is imperative to recognize, though, that the efficacy of NIDS can differ based on the particular kinds of intrusions targeted and the features of the network environment. While NIDS can detect a wide range of intrusions, including network-based attacks like DDoS, network scanning, and DoS, their effectiveness in identifying particular intrusions may be impacted by traffic volume, network topology, and attack complexity. Thus, even though NIDS are essential to network defence strategies, their effectiveness and applicability to various intrusions require careful thought and customization to match different network environments unique needs and difficulties.

To develop an intrusion detection system that works, many models have been and are still being developed based on different approaches and learning techniques. Current models exhibit low detection, high false alarm rates, and poor precision. Deep-learning-model-based NIDSs have been shown to outperform machine learning models in accuracy. However, because of class imbalances in the benchmark datasets, they cannot identify attacks with lower traffic. Contemporary benchmark datasets for intrusion detection feature class imbalances, wherein the amount of normal traffic significantly exceeds that of attacks, thereby simulating real-world network traffic. Specific attacks appear far more frequently than others, even among the various attack types. As a result, the NIDS performs worse overall and has trouble identifying some kinds of attacks. The unbalanced data has received insufficient attention in recent NIDS research despite the fact that it negatively impacts the NIDS's ability to detect attacks accurately. This paper attempts to address these concerns and develop a more effective detection model. The novel improved residual dense network is employed to identify assaults, which enhances detection accuracy. The essential features are extracted and selected based on SE-ResNeXt-101 and the improved binary dandelion algorithm (IBDA). Our findings showed that deep learning methods will increase the accuracy of the model and resistance to threats and attacks by increasing its rate of detection and efficiency.

The remaining work portions are arranged as follows: Section 2 briefly summarizes the prior works. This section outlines the suggested intrusion detection system's technique. In Section 4, experimental findings are described. Section 5 provides a final presentation of the findings and recommendations for future study.

Our Contributions

An effective optimal security solution is suggested for an IDS with a novel deep learning technique to improve cloud computing. The following summarizes the primary contributions of our proposed work:

- A modified generator GAN (MG-GAN) approach is proposed in this paper to tackle the class imbalance issue.
- The feature extraction is carried out using the SE-ResNeXt-101 approach. The NIDS improves overall network security by enhancing its capacity to identify and react to unauthorized or suspect activity by analysing extracted features and comparing them with known attack signatures.
- This work presents a feature selection method that uses the enhanced binary dandelion algorithm to choose features to address the problem of feature redundancy.
- The novel improved residual dense network is utilized to categorize the IDS, in which parameters are tuned using the Lyrebird optimization algorithm.
- Lastly, we evaluate the results of many existing IDS approaches and evaluate the effectiveness of our innovative methodology using three benchmark datasets: WSN-DS, BoT-IoT, and CICDDoS2019.

2. RELATED WORKS

This part examines the prior research that is most pertinent to our study, such as adversarial deep learning instances and current IDS intrusion attempts.

To achieve level-aware black-box adversarial assault tactics, Zhou et al. [21] introduced a Hierarchical Adversarial Attack (HAA) generation approach that targets the Graph Neural Network (GNN)-based IDS in IoT systems with a constrained budget. By creating a shadow GNN framework, an intelligent mechanism utilizing a saliency map technique is formulated

to produce adversarial examples by efficiently identifying and modifying essential feature elements with minimal disturbances. A hierarchical node selection method was created to choose a collection of more attack-prone nodes with a high assault priority.

Liu et al. [22] suggest a NIDS built on LightGBM and Adaptive Synthetic (ADASYN) oversampling technologies. To prevent the influence of the lowest or highest based on the total features, they utilize data preprocessing to normalize and one-hot encode the original data. To address the issue of poor minority attack detection rate brought on by unbalanced training data, they secondly employ ADASYN oversampling technology to boost minority samples. Ultimately, the LightGBM ensemble learning method is used to minimize the system's temporal complexities further while maintaining the precision of detection.

For a more effective intrusion detection system, Al et al. [23] suggested using a hybrid deep learning (HDL) network made up of Long Short-Term Memory (LSTM) and a CNN. Furthermore, data imbalance processing was employed to lessen the impact of data imbalance on the system's efficiency. This processing included the SMOTE and Tomek links sampling methods known as STL.

Jiang et al. [24] provide a hybrid sampling and deep hierarchical network intrusion detection technique. Using the SMOTE model, they initially enhance the minority instances after reducing the noise in the majority categories using one-side selection (OSS). By establishing a balanced data set in this manner, the time needed for training the model may be significantly decreased and the system can fully understand the properties of minority instances. Secondly, they create a deep hierarchical network framework by extracting temporal characteristics using BiLSTM and spatial features using convolution neural networks (CNNs).

Kunang et al. [25] suggest utilizing a pretraining strategy with a deep autoencoder (PTDAE) in conjunction with a Deep Neural Network (DNN) to create a deep IDS. Hyperparameter optimization techniques were used to make the models. Through an autonomous hyperparameter optimization approach that incorporates grid search and random search methods, this study offers an alternative to DL framework systems. Finding the ideal category hyperparameter configuration and hyperparameter values to enhance the detection efficiency is made easier with the aid of the approach.

The efficacy of NIDS in protecting network environments is contingent upon resolving significant deficiencies delineated in contemporary scholarship. Research has highlighted several critical issues that NIDS must address, such as its limited capacity to scale in expansive and dynamic network environments [21], its inability to identify zero-day attacks with conventional signature-based and anomaly-based techniques [22], its high false positive rates that result in alert fatigue and decreased effectiveness [23, 24], and its requirement for more timely and adaptive detection mechanisms to keep up with evolving threats [25]. The problem statement is structured to address these particular gaps in light of these findings, highlighting the necessity of creating more reliable NIDS solutions to improve network security. The following study objectives were employed to address the issues mentioned above:

- The goal is to present the best preprocessing approach for normalization and data encoding from the raw data set. Furthermore, the proposed model simply requires real-time data usage; preprocessing is optional.
- Create a feature selection method that effectively prevents overfitting caused by the high dimensionality of feature space, lowering the IDS model's complexity.
- To develop a novel classifier with a low error rate and high accuracy for detecting and classifying the type of intrusion or assault.
- To demonstrate the efficacy of our proposed model, we will validate it using the three standard benchmark datasets BOT-IOT, CICDDoS2019 and WSN-DS.

3. PROPOSED METHODOLOGY

Network traffic is continuously monitored and analysed by network intrusion detection systems (NIDSs) to spot potentially harmful or security-threatening activity. The dynamic nature of cyber threats has made it necessary for

detection algorithms to undergo constant upgrades and enhancements, which could make some of the older NIDS research less adaptable and possibly obsolete. This research introduced a novel detection and classification approach based on an improved residual dense network (IRDN). Preprocessing, extracting features, selection of features, and classification are performed. The modified generator GAN (MG-GAN) algorithm tackles the class imbalance problem. The SE-ResNeXt-101 approach is employed to extract the essential features. Then, the features are selected based on the improved binary dandelion algorithm (IBDA). Finally, the hyperparameters present in the IRDN approach are fine-tuned by utilizing the lyrebird optimization algorithm (LOA). The architecture diagram depicts the proposed approach which is shown in figure 1.

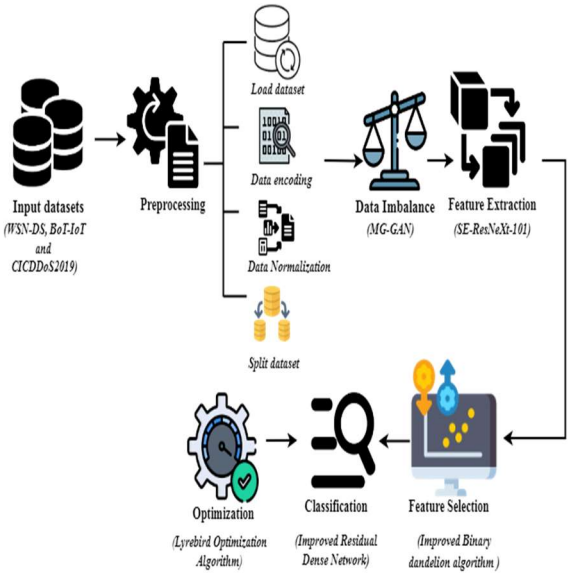


Figure 1. Overall Framework for the proposed methodology.

3.1. Data Preprocessing

The preprocessing stage aims to provide a smoothed and organized dataset, which will serve as the basis for more precise and effective intrusion detection in the NIDS's subsequent phases of analysis.

3.1.1. load datasets

The datasets that we utilized were openly accessible. The information is kept in a pcap-formatted CSV file. This stage involved scanning the specifics of every dataset using the Pandas package and then cleaning each dataset to remove any null and redundant information to set it up for the following step.

3.1.2. data encoding

Label encoding in datasets is carried out at this stage. Working with deep learning approaches involves interacting with numerical values. All dataset labels are not numerical values; we encoded the label column by converting benign or malicious values to integers using the one-hot encoder.

3.1.3. data normalization

One preprocessing method for optimizing within-range features is to normalize data. The learning effectiveness will be impacted by the variability of information read from CSV file, which contains distinct standard derivations and means. We used a standard scalar to scale the input data into our model. The datasets were normalized using the standard scalar based on the "sklearn.preprocessing" library.

3.1.4. data splitting

The modelling data sets are split into two categories: testing and training. To further enhance the algorithm's effectiveness during training, we separate the training data into sets for training and validation.

3.1.5. MG-GAN for data imbalance

Comparing GAN to earlier DL methods, the latter is far more sophisticated. In deep learning, we attempt to classify, cluster, or forecast after training the algorithm with the available data. New things are made in GAN. There are two primary components of GAN. The discriminator attempts to distinguish between the original sample and the created sample; the generator creates instances without comprehending the characteristics of the provided dataset. The samples generated and training instances are used independently by the discriminator. We train the discriminator and the generator in a GAN with the feed-forward network and dropout method. The two blocks cooperate, albeit antagonistically, to better one another. With great attention, the discriminator attempts to understand the original specimens and provides input to the generator regarding the synthetic samples made. As a result of feedback, the generator attempts to produce new samples that closely resemble the original dataset.

The probability factor of error is defined as

Where x represents the number of samples produced by the generator, i serves as a loop counter, and n is the total number of specimens sent as inputs to the discriminator. The proximity between two samples is calculated, determining if the resulting sample is acceptable.

The average error is defined as

$$\sqrt{\frac{(p+q)^2-(p-q)^2}{n}} \quad (2)$$

The distinction among the created sampling and initial sample from a database with n instances is calculated using the data points$p$, q. Normal computation of the anticipated distribution of probabilities involves two measures. The Kullback–Leibler divergence formula (KL) is a conventional equation. Asymmetric KL divergence exists. Jensen–Shannon divergence (JSD), which determines how similar two probability distributions are, is used by GANs

.

The JSD is represented as

$$JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M) \quad (3)$$

Where M represents the average of P and Q. The distribution of probability P's divergence from the average is represented by the symbol. We suggest a modified generator GAN, in which, like in conventional GAN, the discriminator is trained using the initial data. Contrary to the fundamentals of a standard GAN, the generator receives the original data and multi-variate noise with distribution P_n as inputs. We use a Gaussian covariant distribution, where P_o and P_s represent the likelihood distributions of the synthesized and initial information, respectively. This distribution's dimension aids in defining the latent space. The latent space specifies the intrinsic range of variation. This facilitates the generator's ability to produce samples inside the allowed latent space. The discriminator recognizes the synthetic data as accurate because they are within the defined area and closely resemble the actual data. The generator can be trained without repeatedly iterating the feedback loop. This reduces the need for computation and its expense. These two antagonistic blocks operate using a min-max strategy.

In supervised learning scenarios, inequalities in classes are an issue that can be strategically addressed with MG-GAN, which is

$$lik(\theta) = \prod_{i=1}^{n} f(x_i|\theta) \quad (1)$$

mainly intended for generating realistic data instances. An important tactic is to use MG-GANs for data augmentation, particularly for minority classes. Unbalanced datasets are greatly enhanced by MG-GANs, helping the model better broaden to minority classes with sparse real-world samples. The generator is trained to produce artificial samples of underrepresented classes to accomplish this. Additionally, MG-GANs can be used in conjunction with minority class oversampling to create a class population that is more equally distributed. Concentrated MG-GAN training ensures that samples produced closely resemble specific classes, especially minority categories, and concurrently trains discriminators to distinguish between produced and actual samples.

3.2. Feature Extraction Using SE-ResNeXt-101

The process of extracting pertinent data from raw network traffic and turning them into useful features is known as feature extraction in network intrusion detection systems. These features pick up on these traits and patterns connected to malicious and benign network activity. Because of its deep architecture, which consists of squeeze-and-excitation (SE) blocks of data, residual learning with skipped connections, and an ensemble method via its "Next" parameter, SE-ResNeXt-101 was chosen for feature extraction over other CNNs. The model's depth allows it to identify intricate hierarchical characteristics, and the SE blocks enhance feature recalibration by emphasizing the importance of relevant data. The inclusion of ensemble learning via cardinality further improves its capacity to identify a wide range of patterns. SE-ResNeXt-101 is well-known for its cutting-edge performance and models that have been trained, providing a strong basis for transfer learning. In light of the stated objectives, the model is a strong contender for efficient feature extraction due to its widespread acceptance in the research community and capacity to satisfy task-specific requirements.

Utilizing a pretrained SE-ResNeXt-101 model, characteristics were extracted from the input data. A ResNeXt101-32x4d variant with an extra squeeze-and-excitation component was called SE-ResNeXt-101-32x4d. The computing unit that could be created from the process of transformation, E_{ks}, which converts the input data

$Z \in S^{B' \times Y' \times R'}$ to feature mappings $V \in S^{B \times Y \times R}$, was a squeeze-and-excitation block. The kernels were represented by $U = [u_1, u_2, \ldots, u_R]$ where u_R stands for the parameter value of the R[th] filter. E_{ks} was a convolutional operator. Formula (1) can thus describe the outcome as $V = [v_1, v_2, \ldots, v_R]$.

$$v_r = u_r * Z = \sum_{m=1}^{R'} u_r^m * Z^m \quad (4)$$

In this case, $U = [u_r{}^1, u_r{}^2, \ldots, u_r{}^{R'}]$, $Z = [z^1, z^2, \ldots, z^{R'}]$, and $v_r \in S^{B \times y}$ are represented by the symbol *. The 2D spatial kernel, um r, acts on the corresponding Z channels and represents a single u_r^m channel. Bias terms were removed to simplify the terminology. Channel connections were automatically encoded as u_r because the result was the sum of all the channels; however, they became entangled with the local spatial relationships that the filters had gathered. One potential solution to the problem of channel dependency exploitation is to compress global geographical data into channel descriptors. Global average pooling was used to generate channel-specific data to achieve this. In formal terms, the statistic $a \in S^R$ was produced by reducing V over dimensions of space B × Y. This allowed for determination of the r[th] element using formula (5):

$$a_r = E_{md}(v_r) = \frac{1}{B \times Y} \sum_{j=1}^{B} \sum_{i=1}^{Y} v_r(j, i) \quad (5)$$

An additional operation was carried out to fully collect channel-wise dependencies using the data collected during the squeeze operations.

An adaptation of ResNet that bore a striking resemblance to the inception paradigm was the ResNeXt modules. Both follow the merge approach, except in this version, the outcome of distinct routes was combined rather than depth concatenated as in the inception approach. Research showed that boosting cardinality produced more accuracy than expanding or deepening the search. The results of this layer were then depth concatenated and fed to a 1 × 1 convolutional layer.

$$e(Z) = \sum_{j=1}^{R} K_j(z) \quad (6)$$

$K_j(z)$ in (6) is a function of any kind. Projecting z into a (perhaps low-dimensional) space, K_j modifies and embeds it, resembling a primary neuron. The size of the set of transformations to be aggregated is indicated by the letter R in Formula (3). The word “cardinality” is used to characterize R. The level of cardinality determines the number of intricate transformations. The combined transform in Formula (7) is the residual function.

The outcome was denoted by f.

$$f = Z + \sum_{j=1}^{R} K_j(Z) \quad (7)$$

The residual learning and attention mechanisms underpinning SE-ResNeXt-101's operation make it suitable for extracting feature tasks such as those found in NIDSs. It uses residual connections, which are an extension of the ResNet design, to record the distinction between input and output, making deep network training easier. Squeeze-and-excitation (SE) blocks improve discriminative power by adaptively scaling channel significance, further refining feature maps. The network can now capture a variety of patterns thanks to the addition of a cardinality parameter called "Next," which generates an ensemble of paths within every block. The deep structure and high model capacity of SE-ResNeXt-101 enable it to learn complex hierarchical features, making it an excellent choice for extracting features in network intrusion detection systems.

3.3. Feature Selection

The appropriate characteristics for detecting network intrusions are chosen using the second percentile approach and recursive feature removal approach after the data have been preprocessed. The system's computational complexity is significantly decreased by selecting the best features. The features are selected using the improved binary dandelion algorithm.

3.3.1. binary dandelion algorithm (BDA)

Using the binary dandelion algorithm (BDA) in a discrete search space was suggested. The entire search domain in BDA is called “land,” and it can be classified as fertile or impoverished. Core dandelions (CDs) are dandelions that grow on fertile land; assistant dandelions (ADs) are

dandelions that grow on poor terrain. The seeds that a dandelion sows will be dispersed over the surrounding area. The process comprises four stages: selection, normal seeding, mutation seeding, and initiation.

step 1: initialization.

Every seed in the characteristic selection puzzles represents the chosen characteristic. We encode the seeds given a D-dimensional data collection $X = (x_1, x_2 \ldots, x_D)$, where i = 1, 2, …, D and $x_i \in \{0,1\}$. If $x_i = 1$, the ith feature is chosen. If not, there is no selection for the ith feature. N dandelions will be randomly created at the start of BDA and f indicates the fitness function. The seeds are $\{X_1, X_2, \ldots, X_n\}$. Assistant dandelions are still present.

step 2: normal sowing.

Every dandelion's fitness rating is correlated with the quantity of seeds it produces in BDA. The following equation estimates precisely how many seeds a dandelion will yield:

In the formula, $f_{\max_{i=1,2,..n} i}$ and $f_{\min_{i=1,2,..n} i}$, where max and min denote the highest and lowest amount of seeds, respectively, and ε is a small constant that keeps the value of the denominator from going to 0. Every dandelion may only seed inside its seeding radius, which is the specific range surrounding it. A core dandelion (CD's) seeding radius differs from an AD's. The equation that follows is used to determine the CDs' seeding radius:

$$R_{CD}^{t}(i,j) = \begin{cases} Bound & t = 1 \\ R_{CD}^{t-1}(i,j) \times r + r_1 & a = 1 \\ R_{CD}^{t-1}(i,j) \times e + r_2 & \mathrm{a} \neq 1 \end{cases} \quad (9)$$

In this case, the greatest seeding radius is Bound. The wilting and growth factors are denoted by r and e, correspondingly, with the following value varying: $e \in [1,1.1] r \in [0.9,1]$. There are two randomized numbers in the range [−0.5, 0.5] called r_1 and r_2. a is computed using an additional equation to ascertain when the present generation has discovered a higher fitness level than the preceding generation:

$$a = \frac{f_{CD}(t)+\varepsilon}{f_{CD}(t-1)+\varepsilon} \quad (10)$$

In the t[th] generation, the value of the fitness of the CD is represented by $f_{CD}(t)$. To keep the denominator from going to zero, ε is a very tiny constant. The equation used to determine the seeding range of ADs is:

$$R_{AD}^{t}(i,j) = \{ Bound \quad t = 1 \\ w \times R_{AD}(t-1) + r_3 \times (r_4 \times X_{CD}^{t}(i,j) - X_{AD}^{t}(i,j)) \quad t > 1 \quad (11)$$

When the factor of the weight is denoted by w, it is determined using the subsequent equation:

$$w = 1 - \frac{Fe}{Fe_{max}} \quad (12)$$

Fe_{max} is the maximum assessment time, while Fe is the present assessment time. The seeding radius determines every seed's location.

$$M_i = \begin{cases} \max \times \dfrac{f_{\max} - f(X_i) + \varepsilon}{f_{\max} - f_{\min} + \varepsilon} & M_i > \min \\ \min & M_i \leq \min \end{cases} \quad (8)$$

We employ the transfer function to translate the seeding perimeter into a measure of the location of a vector element's reversal likelihood, p(i, j), because the procedure is used in discrete space.

The V-shaped curve $F(x) = \left| erf\left(\frac{\sqrt{\pi}}{2} x\right) \right|$ is selected as the BDA function of transfer based on research. As a result, the equation that follows yields the flip likelihood:

$$P^t(i,j) = \begin{cases} F(R_{CD}^{t}(i,j)) the\ seed\ is\ core\ dandelion \\ F(R_{AD}^{t}(i,j) the\ seed\ is\ assis\ tan\ t\ \ dandelion \end{cases} \quad (1)$$

Therefore, the equation for spreading seedlings is:

$$X^{t+1}(i,j) = \begin{cases} 1 - X^t(i,j) & rand() \leq p^t(i,j) \\ X^t(i,j) & rand() > p^t(i,j) \end{cases} \quad (2)$$

step 3: mutation seeding.

We put the seeds through a mutation seeding step to boost population variety and

enhance the capacity to eliminate the optimal locale. Within every generation, a particular amount of seeds undergo mutations, which are represented by the following Equation (15):

$$X_M(i,j) = 1 - X(i,j) \quad (15)$$

step 4: strategy selection.

Only a portion of the seeds and dandelions in every generation can make it to the following round. Naturally, there is a greater chance that seeds with higher fitness values will enter the following cycle. We use an additional equation to determine every seed's likelihood of survival:

$$w_i = \frac{f_i}{\sum_{n=1}^{SN} f_n} \quad (16)$$

$$f_i = \left| f(X_i) - f_{avg} \right| \quad (17)$$

In the present generation, f_{avg} denotes the overall mean value of fitness of dandelions and seeds, while SN indicates the total of all dandelions and seeds.

BDA Utilizing Enhanced Seeding Strategy (SBDA)

In our earlier work, we presented a BDA that depends on chaos and seeding methods (SBDA) to enhance the efficiency of BDA further. The development and wilting factors are eliminated in SBDA, and the prior best populations are used to reorganize the core dandelions. The array's initial seed will be modified if it is filled. The following equation updates the CDs' seeding radius before the K[th] iteration:

$$Scope(t+1)_i = Scope(t)_i + r \ \ sin\, sin\, (r) \quad (18)$$

In the t[th] formation of core dandelions, $Scope(t)_i$ represents the radius of seeding in the i[th] dimension, and r is a random integer. Here is the equation used to adjust the CD's seeding radius following the K[th] iteration:

$$Scope(t+1)_i = scope(t)_i + r_1 \times (2 \times r_2 \times average(i) - 1) \quad (19)$$

In the following equation, $average(i)$ reflects the mean of all seeds of the i[th] dimensional historical optimum individuals while r_1, and r_2 represent two randomized numerals:

$$average(j) = \frac{\sum_{i=1}^{k} Best(i,j)}{K} \quad (20)$$

We also incorporate chaotic individuals into SBDA. The chaotic map that we select is called a tent, and its equation is outlined below:

$$p(i+1,j) = \begin{cases} 2 \times p(i,j) & 0 < p(i,j) \le 0.5 \\ 2 - 2 \times p(i,j) & 0.5 < p(i,j) < 1 \end{cases} \quad (21)$$

3.4. Classification

To enhance intrusion detection reliability and effectiveness, the architecture of deep learning is used in the categorization of network intrusion detection systems (NIDSs) using an improved residual dense network (IRDN).

improved residual dense network

One deep learning framework that is well-known for its efficiency in various tasks related to computer vision is the residual dense network (RDN). An RDN is a deep convolutional neural network that uses dense and residual connections to rebuild data at super-resolution.

In a traditional CNN, the input of each layer is the result of the preceding layer. However, issues like gradients exploding and disappearing could arise from this. To address this issue, residual connections are incorporated into the RDN. Residual connections, which add the inputs to each residual block's final result, allow the network within each residual block to learn the difference between its input and its outcome. This improves training equilibrium and permits a deeper network depth.

On the other hand, dense connections combine the inputs and outputs of each block, allowing the network to train on all of the data from previous levels. This fixes the gradient vanished problem and enhances the network's ability to extract characteristics. As a result, the RDN may broaden the network and increase efficiency. The residual and dense links framework, which includes an integrated function with batch normalization (BN), ReLU, Conv, and

other operations, make up the fundamental units of the RDN, the residual dense block (RDB).

To increase its capabilities, we can add more layers, such as spatial and channel attention. The layers above will assist the network to focus more intently on important spatial locations and channel-wise connections within the data.

The spatial attention layer is implemented after each residual dense block. The average information across geographical dimensions is first obtained by pooling global average operations. A dense layer triggered by a sigmoid is included after the vector has been resized to a 1 × 1 spatial dimension. This operation, which assigns weights for attention to different spatial regions, enables the network to highlight salient features.

Similarly, the channel attention layer comes after every residual dense block. It includes two global pooling procedures across channel dimensions: mean and maximum pooling. A sigmoid activates a dense layer that receives concatenated outcomes. This process records inter-channel correlations and enables the network to weigh channel-specific attributes adaptively. These attention layers come after every residual dense block in the RDN design, smoothly integrating with it. Once the attention operations are finished, a 1 × 1 layer of convolution is added to enable global learning residuals. This layer guarantees information flow from the source to the output and improves the model's capacity to capture global characteristics.

The characteristics are then combined using a global average pooling operation, and the categorization result is generated by a dense layer using a softmax activation function. The dense layer's class count should correspond to the task's categorization specifications.

The input is initially convolved in the RDB, and the tensor is batch-normalized following the convolution in the RDB. In this case, T1 is the residual combined tensor by T2 and the input layers, and T2 is activated by the LeakyReLU function.

$$T = N(C(I)) \tag{22}$$

The normalization action in Formula (22) is represented by operator N, operator C's convolution process, and I's input layer. The normalized tensor in the RDB is denoted as T. During the entire model life cycle, the initial RDB block.

T1 as the output tensor. This block is utilized to tensor to the subsequent RDB block transitively; this approach is helpful in SR. However, it has a significant weight in categorizing, which impacts the categorization's precision and effectiveness. To resolve this issue, we eventually use the tensor T2, which does not have residual concatenation in the RDB, for residual concatenation.

$$T1 = Concate(T, I) \tag{23}$$

$$T2 = L(T) \tag{24}$$

Concate stands for the residual concatenate operation in Equation (23). T2 is the tensor that comes after LeakyReLU with an alpha of 0.3. T1 is the tensor grouped among T and I, while L indicates the LeakyReLU operation.

The primary goal of the attention's spatial layer is to draw attention to significant spatial places in the characteristic maps. The following is the equation for the spatial attention mechanism:

The spatial attention S can be computed as follows given an input feature map X with dimensions H*W*C. At the same time, H indicates the height, the width is represented as W, and the amount of channels is indicated as C.

The Global Average Pooling (GAP) is represented as

$$G = GAP(X) = \frac{1}{H \times W} H_{i=1} W_{j=1} X(i, j, :) \tag{25}$$

The activation function of the sigmoid in the dense layer is employed as

$$S_{spatial} = \sigma(Dense(G_{reshape})) \tag{26}$$

The element multiplication of the combined layer is represented as

$$X_{attented} = X \bullet S_{spatial} \tag{27}$$

In this case, element-wise arithmetic is indicated by σ, which stands for the activation function of the sigmoid. The channel attention

layer captures the map of the feature inter-channel connections. The channel mechanism for attention can be expressed using the formula that follows:
The activation function of the sigmoid in the dense layer is employed as

$$S_{channel} = \sigma(Dense(G_{concat})) \quad (28)$$

The element multiplication of the combined layer is represented as

$$X_{attented} = X \bullet S_{channel} \quad (29)$$

The RDN design applies these layers of attention after every residual dense block, which helps the framework adaptively focus on pertinent spatial positions and channel-wise data throughout the NIDS categorization process. Subsequently, the residual connection operation produces a new tensor (T3), created by the block outcomes of the 3-layer RDB and the original input tensor.

$$T3 = R^3(C(C(I))) \quad (30)$$

Operator R^3 in Formula (30) denotes the 3-layer RDB operation. The input layer might be reloaded for the remaining connections to improve classification accuracy following three pooling processes. Equation (31) then shows that tensor T3 is used to perform the residual linked operation and the resultant tensor T4 prepares the data for categorization.

$$T4 = Concate(T3, P^3(C(I))) \quad (31)$$

P^3 represents the third pooling operation. We incorporate a thick layer for categorization. To avoid overfitting, the optimizer uses the Lyrebird optimization algorithm (LOA), the loss function is cross-entropy, and an L2 regularizer is included in the dense layer.

To handle the multiple categorization problem more efficiently, the output layer used the function of cross-entropy, and the loss layer utilized the softmax activation function.

$$loss(x, class) = -x[class] + log\left(\sum_{j=0}^{K-1} exp(x[\right. \quad (3)$$

Integrating the ideas of residual and dense connections defines the way an improved residual dense network (IRDN) for an NIDS operates. The vanishing gradient issue in deep networks is resolved by residual connections, which enable the network to learn residual functions by utilizing skip connections. Dense connections link every layer to every layer before it, making feature reuse and information flow easier. These coupled connections allow the network to instinctively acquire and utilize pertinent features from network traffic data in a residual dense network for NIDS categorization, improving the network's capacity to distinguish between malicious and legitimate activity.

3.5. Parameter Optimization using Lyrebird Optimization Algorithm

The rationale for the proposed lyrebird optimization algorithm (LOA) and its mathematical structure for application in optimization scenarios are presented in the ensuing section. By mimicking lyrebird behaviour in the wild, the LOA optimizes the parameters. Organisms imitate the characteristics of the fittest options during a mimicry phase, which involves a population with parameter arrangements. Then, the method presents random disturbances to maintain diversity while examining the parameter space. Finally, the fitness assessment guides the adaptation process to select superior options for the subsequent iteration. The LOA effectively explores and adjusts to the NIDS parameter space by utilizing mimicry and adaptation to identify ideal configurations for improved intrusion detection effectiveness.

3.5.1. inspiration of LOA

The superb lyrebird and Albert's lyrebirds are the two species native to Australia. These magnificent birds belong to the family Menuridae. They can remarkably mimic both manufactured and natural noises from their surroundings. Some of the most recognizable native birds of Australia have a distinctive plume of neutral-coloured tail feathers. Male superb lyrebirds measure 80–98 cm long, while females are 74–84 cm. In contrast, the maximum size of a female Albert's lyrebird is 84 cm, while the maximal size of a male is 90 cm. Although they are similar in some ways, Albert's lyrebird has less beautiful lyrate feathers than the superb lyrebird. Superb lyrebirds weigh around 0.97 kg, while Albert's lyrebirds weigh around 0.93 kg.

When the lyrebird detects possible danger, it carefully surveys its surroundings and then either flees or hides in a suitable location. The suggested LOA approach discussed below was created using computational simulations of this lyrebird technique in times of peril.

3.5.2. algorithm initialization

The suggested LOA method is based on an individual metaheuristic algorithm in which the population comprises lyrebirds. By leveraging the collective search capability of its members, the LOA can offer appropriate alternatives for problems with optimization in an iteration-based procedure. As a member of the LOA, every lyrebird chooses the value of the decision variables according to where they are in the problem-solving space. In the LOA, the algorithm's individuals may be represented mathematically as a matrix by Formula (33). Equation (34) randomly initializes LOA members' positions in the problem-solving space.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i,1} & \cdots & x_{i,d} & \cdots & x_{i,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,m} \end{bmatrix}_{N\times m} \quad (33)$$

$$x_{i,d} = lb_d + r \cdot (ub_d - lb_d) \quad (34)$$

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1} \quad (35)$$

X represents the LOA matrix of the individuals, Xi denotes a possible outcome, x_i is its d[th] dimensions, N indicates the number of lyrebirds, m represents the number of decision variables, r indicates the amount chosen at random within the interval [0,1], $l$band ub stands for the lower and upper bounds. The function of the existing objective can be assessed by considering that every member of the LOA represents a potential solution to the issue. Thus, the values for the function of the objective are available and correspond to the number of individuals.

Here, the assessed function objective is denoted byF_i, and the estimated objective function vector is represented by F. An appropriate criterion for determining the caliber of the potential solutions is estimated. Additionally, as the lyrebirds' positions within the problem-solving space change with every iteration, the optimal candidate solution must also be updated based on assessing the function's objective value.

3.5.3. mathematical modelling of LOA

The suggested LOA approach's architecture updates individual members' positions during every phase based on an algebraic representation of the lyrebird strategy's reaction to threats. The two phases of the population upgrade procedure are (i) hiding and (ii) escape, based on the lyrebird's decision in this scenario. Equation (4) simulates the lyrebird's decision-making procedure in the LOA design when selecting between hiding and escaping from danger. Hence, only one of the two initial phases updates every LOA member's location during an iteration.

$$update \text{ process for } X_i: \begin{cases} based \text{ on phase 1}, r_p \le 0.5 \\ \text{based on phase 2}, \quad \text{else} \end{cases} \quad (36)$$

Where r_p indicates the randomized number from the range [0,1].

Phase 1: Escaping Strategy (Exploration Phase)

Using an illustration of the bird's flight from the dangerous position to the safe areas, an individual member's location can be modified in the search area throughout this stage of the LOA. The lyrebird's capacity to explore new locations in the problem-solving space and make significant positional changes after moving to a safe place indicates the LOA's global search exploration capability. The positions of other population members with higher objective function values are considered secure areas for every participant in the LOA design.

$$S\ A_i = \{X_k, F_k < F_i \text{ and } k \in \{1,2,..,N\}\}, where\ i = 1,2,\ldots,N \quad (37)$$

The collection of secure regions for the i[th] lyrebird is SA_i in this instance and the row of the X matrix with a higher function's objective value than the i[th] LOA member is represented by X_k. The lyrebird is thought to randomly flee towards one of these safe havens in the LOA architecture. Formula (35) determines an alternate position for

every LOA member depending on the lyrebird movement modelling completed in this step. The new spot then substitutes the previous place of the relevant member by formula (38) if the outcome of the objective function is enhanced.

$$x_{i,j}^{P1} = x_{i,j} + r_{i,j} \cdot (SSA_{i,j} - I_{i,j} \cdot x_{i,j}) \quad (38)$$

$$X_i = \{X_i^{P1}, \quad F_i^{P1} \leq F_i \; X_i, else \quad (39)$$

The chosen secure area for the i[th] lyrebird is denoted by$SSA_{i,j}$, its j[th] dimension; the new spot for the ith lyrebird, $x_{i,j}^{P1}$, depends on the escaping method of the suggested LOA; F_i^{P1} is the function's objective value; randomized numbers from the range [0,1] are represented by $r_{i,j}$; and data that are selected at random as 1 or 2 are represented by $I_{i,j}$.

Phase 2: Exploitation Phase

During this stage of the LOA, a population member's spot is modified in the search space according to the lyrebird's modelled approach of hiding in its immediate safe area. Little variations in the lyrebird's position demonstrate that the LOA can be exploited in local search when it is used to accurately survey the surroundings and move in modest steps to find a suitable hiding place.

In the LOA design, a new spot is determined for every LOA member utilizing formula (40) depending on the simulation of the lyrebird's migration towards the nearby appropriate region for concealment. If the new spot enhances the outcome of the intended

$$x_{i,j}^{P1} = x_{i,j} + r_{i,j} \cdot (SSA_{i,j} - I_{i,j} \cdot x_{i,j}) \quad (40)$$

$$X_i = \begin{cases} X_i^{P1}, & F_i^{P1} \leq F_i \\ X_i, & else \end{cases} \quad (41)$$

function as per Formula (41), it eliminates the old spot of the appropriate member.

Where X_i^{p2} is its j[th] dimension, F_i^{p2} is the desired function value.

3.5.4. computational complexity

This section assesses the computation required of the suggested LOA technique, considering both time and space complexity. The initialization procedure, determining the goal function, and population update have the following effects on the time complexity of the LOA:

- The temporal complexity of the LOA's setup and activation phases is O(Nm), while the number of decision variables is denoted by m in the problem and N is the total number of lyrebirds.
- The objective function for every lyrebird is determined in every iteration. Consequently, the time complexity of computing the objective function is O(NT), while T is the highest number of LOA iterations. Every lyrebird is upgraded at random depending on whether it is in hiding or escaping phase throughout every repetition. As a result, the time complexity of the lyrebird upgrade process is O(NmT). Figure 2 displays an illustration of the LOA.

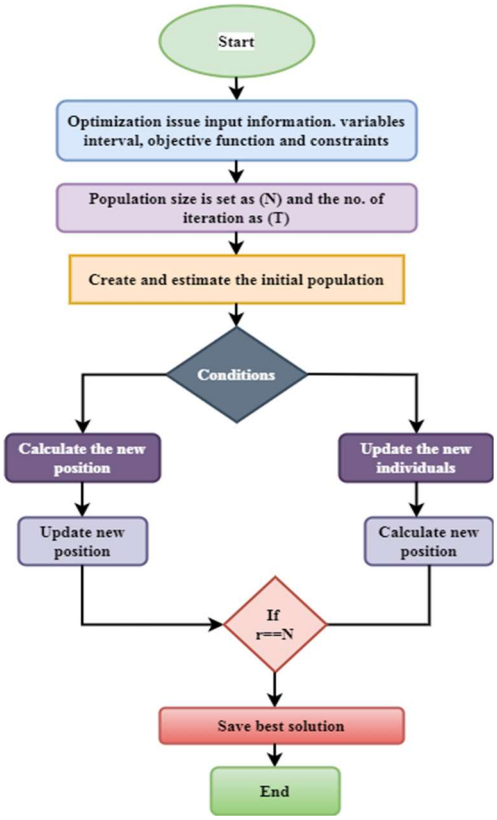


Figure 2. Flowchart.

4. RESULT AND DISCUSSIONS

In this part, the findings of the experiment and outcomes are discussed. This study validates our proposed method on three datasets and estimates its effectiveness. A pair of data instances are created, utilized as the training

dataset, and employed to construct the classifier. Using the testing dataset, a classifier is evaluated in the second stage. Two experiments were conducted to analyze the efficiency of the model. In the first investigation, more than one categorization is used, and in the second, it is contrasted with existing techniques. To evaluate other datasets for cutting-edge machine learning and deep learning techniques, we concurrently designed contrasted investigations on them.

4.1. Experimental Setup

The outcomes on the Python platform were achieved using a PC equipped with an i5 processor and 8 GB of RAM. The efficacy of deep

$$Specificity = \frac{TN}{FP+TN} \quad (44)$$

learning models' classification can be assessed using various techniques. The test setup is shown in Table 1.

Table 1. Setup.

Project	Environment
RAM	16 GB
System	Python
Processor	Intel i5 2.60 GHz
Anaconda	4.5.11
Python	3.9
Backdrop	TensorFlow

4.2. Dataset Descriptions

We examine the three publicly accessible intrusion detection datasets, the BoT-IoT, WSN-DS, and CICDDoS2019 datasets, which have been extensively used in previous investigations.

BOT-IOT dataset: The 11 common upgraded attacks, DDoS, stealing, reconnaissance, and denial of service are all covered by this new NIDS dataset. Over five days, Bot-IoT2019 generated a significant volume of traffic packets and attack kinds. The dataset included 3,119,345 occurrences and 15 characteristics, comprising five class labels (four assault and one regular label).

CICDDoS2019 dataset: The present investigation uses the CICDDoS2019 dataset, which has become widely used for detecting DDoS attacks and classification. The collection includes many recent, actual DDoS attack samples and benign instances.

WSN-DS: To distinguish between legitimate and malicious communication, WSN-DS was created in 2016 and uses sensors to track the number of nodes in wireless networks. The LEACH routing protocol is used to retrieve the records from this dataset, which are represented by 23 characteristics. Four other types of DoS

attacks exist: floods, grayhole, blackhole, and TDMA, in addition to regular records.

$$FAR = \frac{FP}{TN+FP} \quad (45)$$

4.3. Evaluation Metrics

To evaluate the efficacy of our approach, we calculate the false alarm rate, specificity, accuracy, and recall. Additionally, we evaluated the accuracy and F1-score of the framework. The justifications and references for these measurements are as follows:

4.3.1. Accuracy

The system's capacity to accurately determine whether a given behaviour is an attack or regular operation is known as accuracy. Equation (42) can be utilized to compute the accuracy value.

$$Accuracy = \frac{TP+TN}{TN+FN+TP+FP} \quad (42)$$

4.3.2. Sensitivity

Sensitivity shows how well the system distinguishes between all identified threats and the incoming activity representing a real attack. Equation (43) can be used to obtain the sensitivity value.

$$Sensitivity = \frac{TP}{FN+TP} \quad (43)$$

4.3.3. Specificity

Specificity, as opposed to sensitivity, indicates the system's capacity to distinguish between all observed normal data and the one incoming activity that is normal. Equation (44) is used to compute it..

4.3.4. False Alarm Rate

The number of attacks mistakenly estimated as typical activity is known as the false alarm rate. The more assault action anticipated to be typical, the higher the false alarm rate. Equation (45) is applied to achieve this.

Journal of Theoretical and Applied Information Technology
30th April 2024. Vol.102. No 8

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

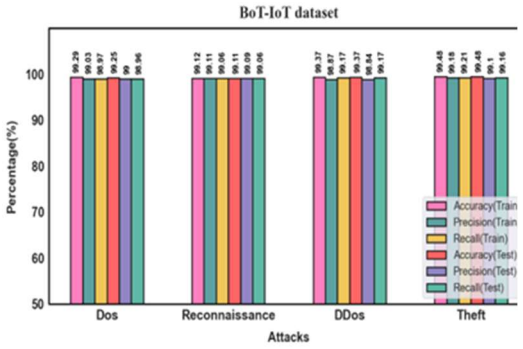
 # Experiment 1 (Assessment on BoT-IoT Dataset)

When assessing the effectiveness of NIDSs in IoT environments, the BoT-IoT dataset is a valuable tool. The complexity and difficulties involved in protecting IoT networks are captured in this dataset, which includes a broad range of genuine cases. Evaluating NIDSs' performance on datasets like BoT-IoT is crucial for determining how resilient and adaptable they are to new threats as academics and practitioners work to improve their capabilities. The class-wise evaluation of the dataset 1 is shown in Table 2.

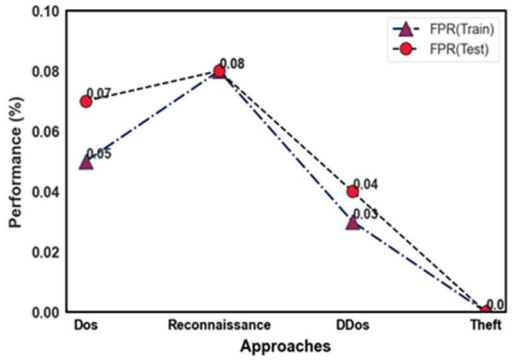
Table 2. *Training and testing class-wise evaluation using the BoT-IoT dataset.*

Attacks	Training Data				Testing Data			
	ACC (%)	Recall (%)	Precision (%)	FPR (%)	ACC (%)	Recall (%)	Precision (%)	FPR (%)
Dos	99.29	98.97	99.03	0.05	99.25	98.96	99	0.07
Reconnaissance	99.12	99.06	99.11	0.08	99.11	99.06	99.09	0.08
DDoS	99.37	99.17	98.87	0.03	99.37	99.17	98.84	0.04
Theft	99.48	99.21	99.18	0.00	99.48	99.16	99.10	0.00

The BoT-IoT dataset contains some attack types. The multiple categorization of the dataset 1 is illustrated in Figure 3. In the models suggested, with every attack class, the efficacy of numerous classification is higher and yields better outcomes. Every class attains a minimum accuracy rate of 99%. These are the ideal guidelines.



(a)



(b)

Figure 3. *Multi-categorization outcome of BoT-IoT dataset. **(a)** Estimation of accuracy, precision, and recall. **(b)** Comparison of FPR.*

Table 3 and Figure 4 compare existing approaches with the proposed approach using the BoT-IoT dataset. The existing approaches like RNN, DeepDCA, MLP, and CNN are employed to contrast with our proposed approach. In contrast with other methods, the presented approach yields a better result, which is 99.93% accuracy, 99.21% precision, 98.74% recall, and 99.47% of F1-score.

Table 3. *Evaluation based on the dataset 1.*

Reference	Approach	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
[26]	RNN	99.91	-	-	-
[27]	DeepDCA	98.73	99.17	98.36	98.77
[28]	MLP	70.55%	43.82%	74.39%	55.15%
[29]	CNN	99.02	99.09	99.07	99.03
Our work	Proposed method	99.93	99.21	98.74	99.47

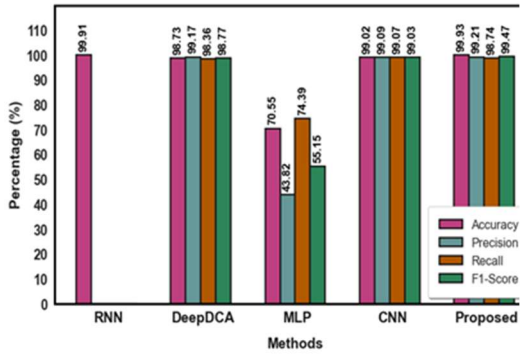


Figure 4. *Differentiation of existing approach with proposed utilizing BoT-IoT dataset.*

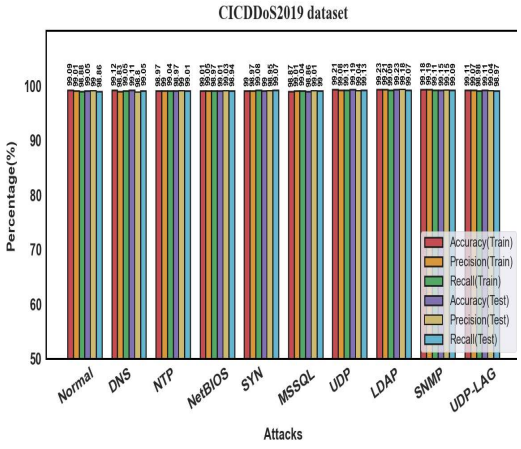
#Experiment 2 (Assessment on CICDDoS2019 Dataset)

Multiple experiments are conducted on dataset 2 to determine the proposed method's effectiveness. Table 4 displays the results of the multi-class identification process.

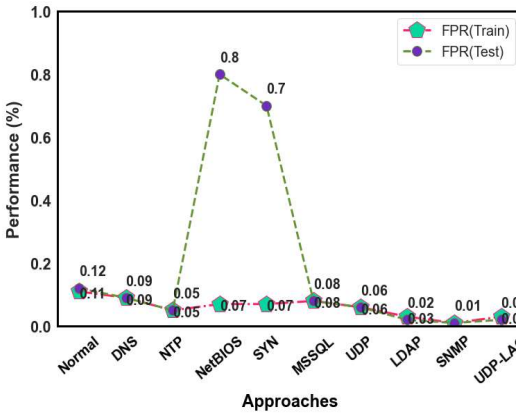
Table 4. *Training and testing class-wise evaluation using the CICDDoS2019 dataset.*

Attacks	Training Data				Testing Data			
	ACC (%)	Recall (%)	Precision (%)	FPR (%)	ACC (%)	Recall (%)	Precision (%)	FPR (%)
Normal	99.09	98.88	99.01	0.11	99.05	98.86	99	0.12
DNS	99.12	99.05	98.83	0.09	99.11	99.05	98.80	0.09
NTP	98.97	99.04	99	0.05	98.97	99.01	99	0.05
NetBIOS	99.01	98.97	99.05	0.07	99.01	98.94	99.03	0.8
SYN	99	99.08	98.97	0.07	99	99.07	98.95	0.7
MSSQL	98.87	99.04	99.01	0.08	98.86	99	99.01	0.08
UDP	99.21	99.13	99.08	0.06	99.19	99.13	99.04	0.06
LDAP	99.23	99.09	99.21	0.03	99.23	99.07	99.18	0.02
SNMP	99.18	99.11	99.19	0.01	99.15	99.09	99.15	0.01
UDP-LAG	99.11	98.98	99.07	0.03	99.11	98.97	99.04	0.02

The multi-categorization outcome of the CICDDoS2019 dataset is shown in Figure 5. The presented approach obtains higher performances in every assault category. Compared with other assaults with more than 99% accuracy, NTP and MSSQL show less accuracy.



(a)



(b)

Figure 5. *Multi-categorization outcome of CICDDoS2019 dataset. (**a**) Estimation of accuracy, precision, and recall. (**b**) Comparison of FPR.*

Differentiation of the existing approach with the proposed dataset 2 is shown in Figure 6 and Table 5. The proposed approach yields greater performance in the CICDDoS2019 dataset.

Table 5. Comparison to similar approaches using the CICDDoS2019 dataset.

Approaches	Precision (%)	Accuracy (%)	F1-Score (%)	Recall (%)
DRCNN [30]	98.89	99.12	99.32	99.06
MLP [31]	84.4	92.5	89	94.2
(AE)+MLP [32]	97.91	98.34	98.18	98.48
Bi-LSTM [33]	97.93	98.18	-	99.84
CNN [33]	93.3	95.4	92.8	92.4
Proposed	99.11	99.23	99.09	99.09

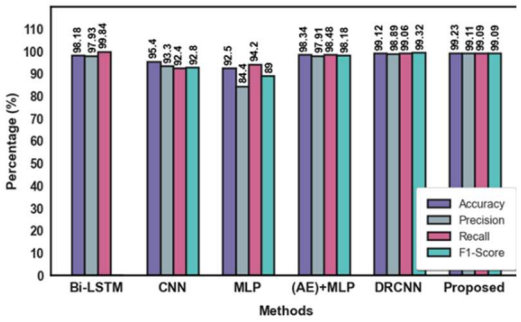


Figure 6. Differentiation of existing approach with proposed utilizing CICDDoS2019 dataset.

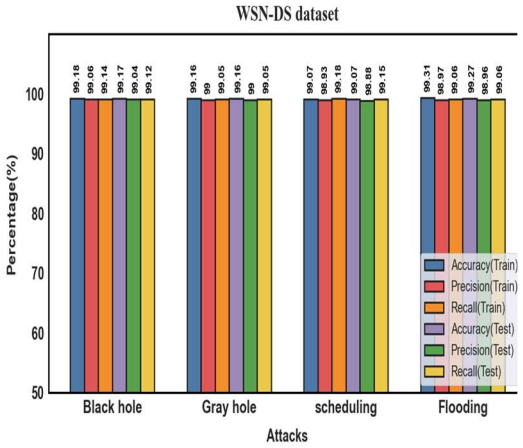
 # Experiment 3 (Assessment on WSN-DS Dataset)

In this phase, we use the WSN-DS dataset to assess the efficacy of the suggested method. The class-wise performance of the WSN-DS dataset is shown in Table 6.

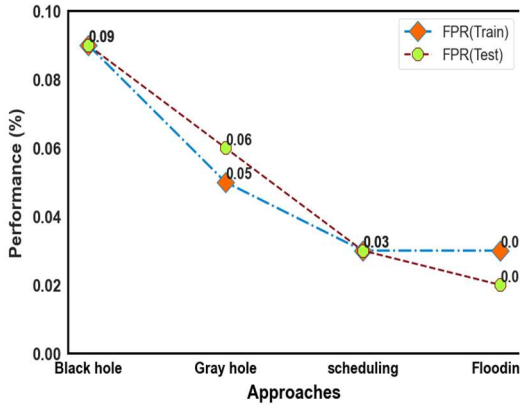
Table 6. Training and testing class-wise evaluation using the WSN-DS dataset.

Attacks	Training Data				Testing Data			
	ACC (%)	Recall (%)	Precision (%)	FPR (%)	ACC (%)	Recall (%)	Precision (%)	FPR (%)
Blackhole	99.18	99.14	99.06	0.09	99.17	99.12	99.04	0.09
Grayhole	99.16	99.05	99	0.05	99.16	99.05	99	0.06
Scheduling	99.07	99.18	98.93	0.03	99.07	99.15	98.88	0.03
Flooding	99.31	99.06	98.97	0.03	99.27	99.06	98.96	0.02

Multiple categorizations of assault are shown in Figure 7. While differentiating from existing approaches, the presented approach has superior performances over all assaults.



(a)



(b)

Figure 7. Multi-categorization outcome of WSN-DS dataset. (a) Estimation of accuracy, precision, and recall. (b) Comparison of FPR.

Table 7 and Figure 8 differentiate existing approaches employing the WSN-DS dataset. Contrasted with prior methods, the presented approach yields a greater performance in F1-score, precision, recall, and accuracy.

Table 7. Evaluation based on the WSN-DS dataset.

Reference	Approach	Accuracy (%)	Detection Rate (%)
[34]	LR	97	77.7
	NB	83.1	76.5
	DT	99.1	95.1
[35]	CNN-LSTM	99.58	97.77
Our work	Proposed method	99.64	98.87

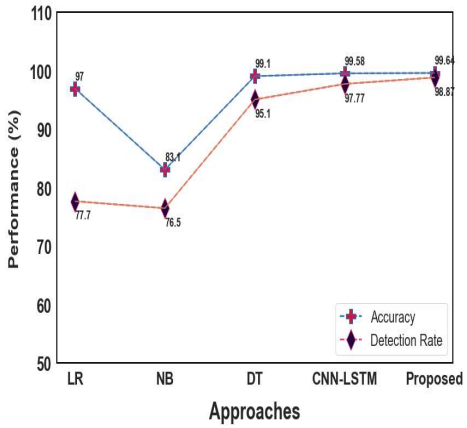
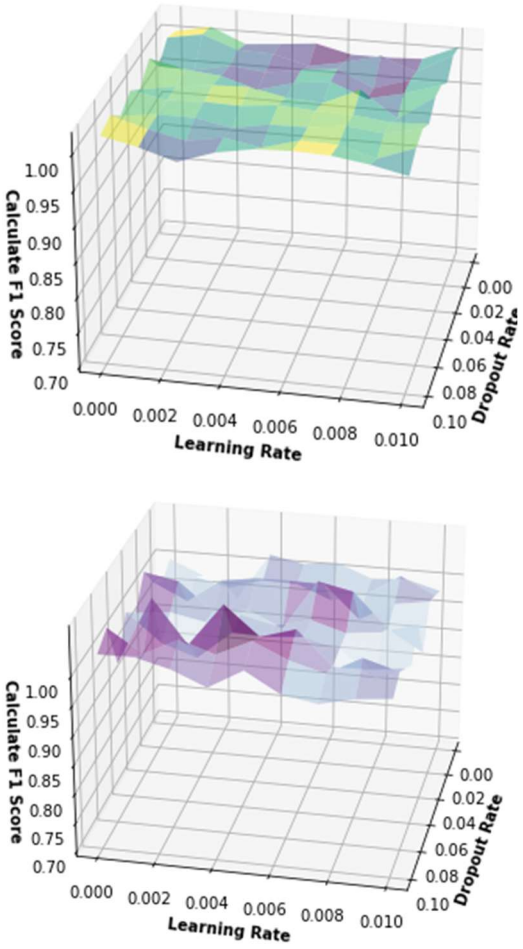


Figure 8. Differentiation of existing approach utilizing WSN-DS dataset.

#Experiment 4 (Hyperparameter Configuration)

The F1-score was used to determine the suggested technique's hyperparameters, including learning and dropout rates. Various hyperparameter tuning strategies can be used to determine the best settings for hyperparameters. Nonetheless, grid search is sufficient to identify the ideal hyperparameter values because the suggested NIDS only considers two parameters. Finding the ideal values is made simple by the upward convex shape of the grid search results, as seen in Figure 9.



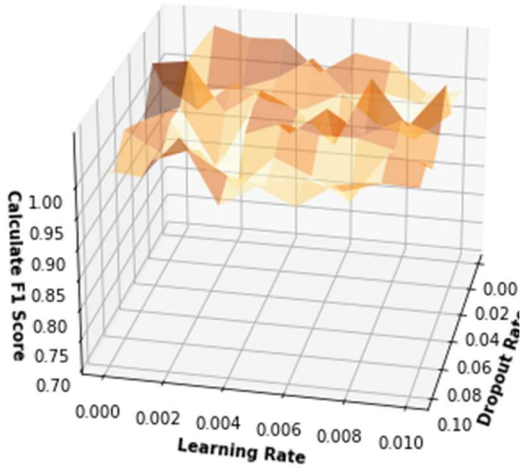


Figure 9. Hyperparameter configuration of proposed datasets

Regarding the ideal value set depicted in the images, the dropout was set for our suggested dataset at 0.0032 and 0.001. For every dataset, the learning rate was set to 0.01.

 # Experiment 5 (Assessment of Training and Testing)

Figure 10 depicts the categorization loss and accuracy percentage of the IDS plotted with the number of iterations. As the graphic illustrates, this paper's approach produces a practical convergent effect. Within the entire dataset, there were distinct phases for training and testing. In this inquiry, 20% of the data is used for testing, and 80% is used for training.

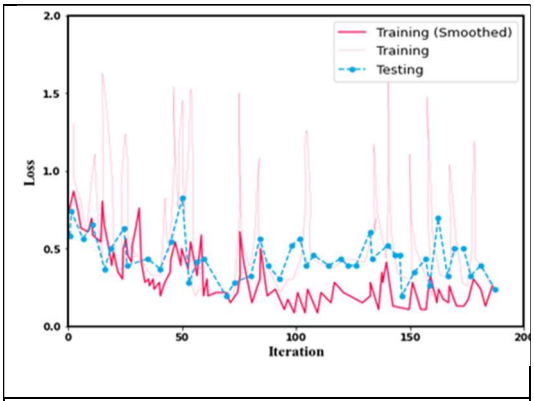


Figure 10. Evaluation of training and testing

Table 8 represents the differentiation of the presented approach from prior research. The performance of the proposed approach is better than that of other approaches.

Table 8. Differentiation of related works.

References	Methods	Pros	Cons	Accuracy
Zhou et al. [21]	GNN	Reduces the time for training of classifiers and increases categorization precision	Overfitting issue occurs	-
Liu et al. [22]	LightGBM	Enhanced detection rate	Its functionality is dependent on input factors	99.91%
Al et al. [23]	LSTM-CNN	Better categorization accuracy	High computational cost	99.17%
Jiang et al. [24]	BiLSTM-CNN	Enhanced categorization efficiency	Overfitting issue	83.58%
Kunang et al. [25]	DNN	Resolution of high-dimensional issues	Ineffective with high-dimensional sets of data	83.33%
Our work	Proposed approach	Enhanced overall performances	It has a significant number of parameters	99.93%

Table 9 shows the differentiation of the proposed feature selection over existing ones. The proposed approach was better at feature selection than other approaches.

Table 9. Differentiation of feature selection algorithms.

Algorithm	Accuracy	TPR	FPR
GA	94	98.1	1.28
SSO	92.4	98.5	1.74
Cuttlefish	91.9	98	1.79
Sigmoid-PIO	94.7	97.4	0.97
Cosine-PIO	96	98.2	0.76
Proposed	99.1	99	0.24

Various existing class imbalance approaches, like DSSTE, AESMOTE, ADASYN, WGAN, and SMOTE, are employed to contrast with the presented model. The presented model yielded better results than the other approaches, as shown in Table 10.

Table 10. Differentiation of class imbalance approaches.

Approach	Accuracy	F1-Score
DSSTE	82.84	81.66
AESMOTE	82.09	82.43
ADASYN	78.97	-
WGAN	80.80	-
SMOTE	79.10	75.76
Proposed	99.00	89.74

The deep-learning-based existing approaches are contrasted with those presented, as shown in Table 11. The proposed method yields superior performances.

Table 11. Differentiation of prior approaches.

Methods	Precision (%)	F1-Score (%)	Recall (%)	Accuracy (%)
HCRNNIDS	93.47	97.98	93.47	94.58
RNN-ABC	95.12	97.29	92.57	96.89
CNN	97.58	92.7	93.5	93.8
LSTM	91.09	93.3	92.55	94.73
Conv-LSTM	98.14	99.67	95.78	97.03
Proposed	99.17	99.11	99.21	99.93

Statistical Significance Test

We employ the widely used Wilcoxon statistical significance test to demonstrate the statistical significance of the efficiency enhancement achieved by the proposed approach. The MG-GAN strategy and the original data are used to assess the test. Additionally, we conduct a test of statistical significance to demonstrate how our suggested approach significantly enhances the findings of previous studies in this area. A nonparametric statistical test called the Wilcoxon signed-ranks test is used to rank the differences between the outcomes of two methods when choosing features using the dataset. It evaluates the positions for positive and negative differences while ignoring the signs. Let di represent the difference in the feature selection approaches' effectiveness ratings on the ith classification model. The differences are then sorted based on their absolute values. In the event of a tie, average ranks are determined. Considering R^+ as the total ranks in which the subsequent method performed better than the first and R^- as the total ranks for the reverse situation,

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \quad (46)$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \quad (47)$$

Ranks of $d_i = 0$ are distributed equally among the totals. One of them is disregarded if there are an odd number of them. Let the Wilcoxon test τ be considered as average, then $\tau = (R^+, R^-)$ for the level of confidence $\alpha = 0.05$.

4.4. Discussion

The main goal of the project is to develop an efficient system for intrusion detection that can discriminate between legitimate and malicious traffic. The complexity of cybersecurity issues has increased due to the daily discovery of new assaults, and standard detection systems for intrusions have a high rate of false alarms that lead security analysts to overlook malicious attempts and leave the system open to attacks of any kind. The data utilized for training intrusion systems are deemed outdated and comprise redundant information, leading to inadequate training and an inefficient process for training and evaluating systems. Recently, experts have started working on deep-learning-based systems for intrusion detection. According to a recent study, deep learning performs better than traditional learning techniques when it comes to classifying received traffic in enormous datasets and continuously attacked environments and identifying fraudulent traffic.

To address the dead neuron problem in the residual block, the activation ReLU function is applied following the convolution process in each layer, and functions in the LeakyReLU are used in the tensor following normalization. The RDN design, which combines the flexibility of attention mechanisms with the strength of residual dense blocks, is enhanced by incorporating these attention layers. The information flow in the RDN is facilitated by the residual connections, and attention layers offer a way to modify the significance of features dynamically. Utilizing the LOA in the optimization layer, the learning rate was adaptively adjusted to 0.0001 at the beginning and the loss function was minimized. While differentiating from other approaches, the presented approach yields greater performances.

4.5. Limitations

Even though the model suggested in this paper increases detection accuracy, it still has certain shortcomings: first, it has a significant number of parameters; second, it improves the accuracy of detection for a limited number of specimens, but the impact of the enhancement is minimal. To increase minority sample identification accuracy, enhance the model's overall categorization effect, and lower the running time cost, we will investigate light weighting of the model in more detail in the future.

5. Conclusions and Future Scope

Generic multiple categorization impacts and insufficient feature extraction are two common problems with traditional intrusion detection algorithms. This work presents a novel approach to attack detection that efficiently classifies and detects attacks to address these issues. The framework uses the improved binary dandelion algorithm (IBDA) and the modified generator GAN (MG-GAN) algorithm to tackle the problems of dataset disparities and redundant feature extraction. Next, the novel improved residual dense network is used to categorize the attacks. The lyrebird optimization algorithm (LOA) is employed to modify the hyperparameters. According to the BoT-IoT, CICDDoS2019, and WSN-DS datasets, the suggested model's accuracy is 99.93%, 99.23%, and 99.64%, respectively.

As a result, superior performance in detection was attained. The prior NIDS demonstrated DR of up to 93.49% and 98.31% based on the F1-score; in contrast, the proposed NIDS attained the greatest detection performance of 98.87% and 99.64% in the effectiveness analysis. Furthermore, compared to existing approaches, the study shows that the framework can effectively extract characteristics with a low FPR and high accuracy in detection from multi-dimensional, massive network data. This was demonstrated by a number of studies, including feature selection analysis, ensemble methods versus single-model evaluations, testing and training time distinction, and the efficacy for the assessment of the dataset.

Additionally, the structure significantly enhanced the impact of detection for a restricted class set, offering promising real-time applications for IDSs. This study can be extended in the future by creating an IDS framework with cutting-edge explainable artificial intelligence (EAI) models. By categorizing mobile traffic, attacks can be found using this technique. Multi-modal deep learning will be employed to enhance the overall efficacy of IDSs.

Author Contributions: Conceptualization, methodology, software Implementation, writing—review and editing, A.G.; supervision, review and editing, S.K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data will be provided upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

[1] Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep learning approach for intelligent intrusion detection system. *IEEE Access* **2019**, *7*, 41525–41550.

[2] Devan, P.; Khare, N. An efficient XGBoost–DNN-based classification model for network intrusion detection system. *Neural Comput. Appl.* **2020**, *32*, 12499–12514.

[3] Pawlicki, M.; Choraś, M.; Kozik, R. Defending network intrusion detection systems against adversarial evasion attacks. *Future Gener. Comput. Syst.* **2020**, *110*, 148–154.

[4] Mulyanto, M.; Faisal, M.; Prakosa, S.W.; Leu, J.S. Effectiveness of focal loss for minority classification in network intrusion detection systems. *Symmetry* **2020**, *13*, 4.

[5] Bertoli, G.D.C.; Júnior, L.A.P.; Saotome, O.; Dos Santos, A.L.; Verri, F.A.N.; Marcondes, C.A.C.; Barbieri, S.; Rodrigues, M.S.; De Oliveira, J.M.P. An end-to-end framework for machine learning-based network intrusion detection system. *IEEE Access* **2021**, *9*, 106790–106805.

[6] Magán-Carrión, R.; Urda, D.; Díaz-Cano, I.; Dorronsoro, B. Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches. *Appl. Sci.* **2020**, *10*, 1775.

[7] Ashiku, L., & Dagli, C. (2021). Network intrusion detection system using deep learning. Procedia Computer Science, 185, 239-247.

[8] Choi, H.; Kim, M.; Lee, G.; Kim, W. Unsupervised learning approach for network intrusion detection system using autoencoders. *J. Supercomput.* **2019**, *75*, 5597–5621.

[9] Almiani, M.; AbuGhazleh, A.; Al-Rahayfeh, A.; Atiewi, S.; Razaque, A. Deep recurrent neural network for IoT intrusion detection system. *Simul. Model. Pract. Theory* **2020**, *101*, 102031.

[10] Gao, Y.; Wu, H.; Song, B.; Jin, Y.; Luo, X.; Zeng, X. A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc networks. *IEEE Access* **2019**, *7*, 154560–154571.

[11] Musafer, H.; Abuzneid, A.; Faezipour, M.; Mahmood, A. An enhanced design of sparse autoencoder for latent features extraction based on trigonometric simplexes for network intrusion detection systems. *Electronics* **2020**, *9*, 259.

[12] Liu, G.; Zhang, J. CNID: Research of network intrusion detection based on convolutional neural network. *Discret. Dyn. Nat. Soc.* **2020**, *2020*, 4705982.

[13] Almomani, O. A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms. *Symmetry* **2020**, *12*, 1046.

[14] Mendonça, R.V.; Teodoro, A.A.; Rosa, R.L.; Saadi, M.; Melgarejo, D.C.; Nardelli, P.H.; Rodríguez, D.Z. Intrusion detection system based on fast hierarchical deep convolutional neural network. *IEEE Access* **2021**, *9*, 61024–61034.

[15] Henry, A.; Gautam, S.; Khanna, S.; Rabie, K.; Shongwe, T.; Bhattacharya, P.; Sharma, B.; Chowdhury, S. Composition of hybrid deep learning model and feature optimization for intrusion detection system. *Sensors* **2023**, *23*, 890.

[16] Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. Network intrusion detection for IoT security based on learning techniques. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2671–2701.

[17] Nguyen, M.T.; Kim, K. Genetic convolutional neural network for intrusion detection systems. *Future Gener. Comput. Syst.* **2020**, *113*, 418–427.

[18] Qiu, H.; Dong, T.; Zhang, T.; Lu, J.; Memmi, G.; Qiu, M. Adversarial attacks against network intrusion detection in IoT systems. *IEEE Internet Things J.* **2020**, *8*, 10327–10335.

[19] Jia, Y.; Wang, M.; Wang, Y. Network intrusion detection algorithm based on deep neural network. *IET Inf. Secur.* **2019**, *13*, 48–53.

[20] Shahraki, A.; Abbasi, M.; Haugen, Ø. Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103770.

[21] Zhou, X.; Liang, W.; Li, W.; Yan, K.; Shimizu, S.; Kevin, I.; Wang, K. Hierarchical adversarial attacks against

graph-neural-network-based IoT network intrusion detection system. *IEEE Internet Things J.* **2021**, *9*, 9310–9319.

[22] Liu, J.; Gao, Y.; Hu, F. A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. *Comput. Secur.* **2021**, *106*, 102289.

[23] Al, S.; Dener, M. STL-HDL: A new hybrid network intrusion detection system for imbalanced dataset on big data environment. *Comput. Secur.* **2021**, *110*, 102435.

[24] Jiang, K.; Wang, W.; Wang, A.; Wu, H. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access* **2020**, *8*, 32464–32476.

[25] Kunang, Y.N.; Nurmaini, S.; Stiawan, D.; Suprapto, B.Y. Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. *J. Inf. Secur. Appl.* **2021**, *58*, 102804.

[26] Ferrag, M.A.; Maglaras, L. DeepCoin: A novel deep learning and blockchain-based energy exchange framework for smart grids. *IEEE Trans. Eng. Manag.* **2020**, *67*, 1285–1297.

[27] Aldhaheri, S.; Alghazzawi, D.; Cheng, L.; Alzahrani, B.; Al-Barakati, A. Deepdca: Novel network-based detection of iot attacks using artificial immune system. *Appl. Sci.* **2020**, *10*, 1909.

[28] Derhab, A.; Aldweesh, A.; Emam, A.Z.; Khan, F.A. Intrusion detection system for the Internet of Things based on temporal convolution neural network and efficient feature engineering. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 6689134.

[29] Alosaimi, S.; Almutairi, S.M. An Intrusion Detection System Using BoT-IoT. *Appl. Sci.* **2023**, *13*, 5427.

[30] Kumar, G.S.C.; Kumar, R.K.; Kumar, K.P.V.; Sai, N.R.; Brahmaiah, M. Deep residual convolutional neural Network: An efficient technique for intrusion detection system. *Expert Syst. Appl.* **2024**, *238*, 121912.

[31] Shewale, Y.; Kumar, S.; Banait, S. Machine Learning Based Intrusion Detection in IoT Network Using MLP and LSTM. *Int. J. Intell. Syst. Appl. Eng.* **2023**, *11*, 210–223.

[32] Wei, Y.; Jang-Jaccard, J.; Sabrina, F.; Singh, A.; Xu, W.; Camtepe, S. Ae-mlp: A hybrid deep learning approach for ddos detection and classification. *IEEE Access* **2021**, *9*, 146810–146821.

[33] Aswad, F.M.; Ahmed, A.M.S.; Alhammadi, N.A.M.; Khalaf, B.A.; Mostafa, S.A. Deep learning in distributed denial-of-service attacks detection method for Internet of Things networks. *J. Intell. Syst.* **2023**, *32*, 20220155.

[34] Edeh, D.I. Network Intrusion Detection System Using Deep Learning Technique. Master's Thesis, Department of Computing, University of Turku, Turku, Finland, 2021.

[35] Halbouni, A.; Gunawan, T.S.; Habaebi, M.H.; Halbouni, M.; Kartiwi, M.; Ahmad, R. CNN-LSTM: Hybrid deep neural network for network intrusion detection system. *IEEE Access* **2022**, *10*, 99837–99849.