Iranian Journal of Fuzzy Systems

Volume 22, Number 5, (2025), pp. 181-202

Original paper

University of Sistan and Baluchestan

# Optimized hybrid fuzzy NFSC approach for enhanced scalable and efficient anomaly detection in secure IoT- systems with low-latency

S. Umar [1], N. T. Gole [2], P. C. Reddy Alla [3] and V. S. Narayana Tinnaluri [4]

[1] *Department of Computer Science and Engineering, Malla Reddy (MR) Deemed to be university Hyderabad, Medchal-Malkajgiri, 500100, Telangana, India.*
[2] *Department of Computer Science and Engineering, Vidarbha Institute of Technology, Nagpur, 441209, Maharashtra, India.*
[3] *Department of Computer Engineering, Marwadi University, Rajkot. 360003, Gujarat, India.*
[4] *Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, 522302, Andhra Pradesh, India.*

syedumar@mrec.ac.in, drnileshtgole@gmail.com, allapoorna.reddy@marwadieducation.edu.in, drvsnarayanatinnaluri@gmail.com

**Abstract**

The rapid growth and widespread integration of the Internet of Things (IoT) across various domains have enabled systems to autonomously send, receive, and analyze data. Despite its significant benefits for improving quality of life and service efficiency, IoT still faces considerable challenges related to security and privacy. An anomaly-based Intrusion Detection System (IDS) can provide security for IoT networks against various cyber-attacks. This study proposes an anomaly-based IDS to prevent cyber-attacks in IoT environments. In this research, we propose a novel optimized ensemble deep learning-based approach for anomaly-based IDS classification. The proposed approach includes several innovative aspects: initially, raw data are pre-processed, and essential features are extracted using CoffeeNet. The approach employs the Binary Bamboo Forest Growth Optimization Algorithm (BBFGOA) to reduce data characteristics and enhance anomaly detection effectiveness. We use the Optimized Fuzzy NFSC approach an ensemble of the Sharpened Cosine Similarity-based Neural Network (SCS-Net) and a rule-based Deep Fuzzy System with Nonlinear Fuzzy Feature Transform (NFFS-DFS)–to categorize network flows as benign or malicious across multiple categories. The Partial Reinforcement Optimization Algorithm (PRO) is used to tune the hyperparameters in the Fuzzy NFSC approach. Additionally, we employ a Conditional Tabular Generative Adversarial Network (CTGAN) to address the class imbalance issue. The proposed anomaly detection approach is evaluated and analyzed using four benchmark datasets to validate its effectiveness. The evaluation's results demonstrate that the approach enhances detection performance, with a 99.43% F1-score, 99.61% accuracy, 99.45% precision, and 99.38% recall. The findings also show that employing an oversampling technique improves minority sample identification rates, which raises overall accuracy. The suggested approach works better than current techniques since it takes less time to train and detect.

*Keywords:* Internet of things (IoT), anomaly-based IDS, CoffeeNet, optimized fuzzy NFSC, partial reinforcement optimization algorithm (PROA), conditional tabular generative adversarial network (CTGAN)..

## 1 Introduction

The IoT is utilized in healthcare, transportation, and smart city applications involving connected physical devices with restricted computational capacity [5]. As the amount of devices grows, IoT networks produce massive amounts of statistics, making them an appealing target for attackers. The most significant difficulties facing IoT are limited resources and cybersecurity [16, 21]. Solving cyber security concerns in IoT networks is becoming the primary concern, which may be addressed by building and deploying effective IDS at edge nodes. In network security, anomaly detection is

Corresponding Author: S. Umar
Received: March 2025; Revised: September 2025; Accepted: October 2025.
https://doi.org/10.22111/ijfs.2025.51473.9092

commonly used for IDSs in IoT networks using a range of deep learning-based anomaly detection methods. Various risks or assaults frequently arise in IoT systems, so it is critical to recognize such assaults and devise a solution to neutralize them to safeguard the network. IoT has three layers: perception, network, and application [15, 18]. Data from IoT devices is collected in the perception layer and transferred to the connection layer, where it is vulnerable to assaults, tampering, and data injection [6]. Encryption can help to address these issues. The layer above the network transmits data via Bluetooth and WiFi, posing risks of DoS attacks and man-in-the-middle attacks [12]. To keep the network secure, security protocols must be implemented. Because of the large amount of data shared, the application layer offers application-specific services that necessitate robust privacy and authentication mechanisms [19]. The algorithm for deep learning has a high overall detection efficiency but performs poorly when the data is imbalanced. The proportion of normal and attack categories varies, leading to an unbalanced dataset. As an outcome, the structure is skewed towards the class with more samples. As a result, the dataset needs to be balanced before the model is trained. A dataset with numerous features necessitates complex DL approaches [14] with a massive amount of neurons and massive computation; therefore, valuable characteristics from the dataset must be chosen to decrease the computation cost and number of neurons [17].

The static representation of features used by current deep learning-based IDS models, including PCC-CNN [8] and IoTFECNN-BMECapSA [4], hinder their capacity to adapt to the varied and changing nature of IoT attack patterns. We tackle this by using CoffeeNet as a deep hierarchical feature extractor, which makes it possible to capture representations that are richer and more discriminative. Our Binary Bamboo Forest Growth Optimization Algorithm (BBFGOA) effectively identifies the most informative features while maintaining high detection accuracy, overcoming the high computational overhead and risk of selecting suboptimal subsets associated with previous feature selection techniques such as BMECapSA [4] and hybrid filter-wrapper strategies [7]. In contrast, our Conditional Tabular Generative Adversarial Network (CTGAN) generates high-fidelity minority-class instances, greatly increasing uncommon attack detection rates. Common imbalance-handling techniques like SMOTE [20] may yield unrealistic synthetic samples. Our Optimized Fuzzy NFSC, which combines SCS-Net and NFFS-DFS, improves flexibility and classification accuracy in heterogeneous IoT traffic. Conventional classifiers such as CNN [8], LSTM [2], and bagging ensembles [7] have trouble maintaining strong decision boundaries. Furthermore, we incorporate the Partial Reinforcement Optimization Algorithm (PRO) to attain an ideal accuracy-efficiency trade-off, in contrast to previous works [1, 8, 13] that do not have automated hyperparameter adjustment. As a result of these advancements, a new, effective, and optimized fuzzy ensemble method for high-performance anomaly detection and categorization in Internet of Things environments is presented.

The proposed IDS framework takes a design where optimization is essential to improving both classification and detection performance in light of these constraints and specifications. In addition to having strong classification models, accurate anomaly detection in IoT networks also requires choosing the most pertinent features and fine-tuning model parameters for best results. Our method ensures better detection accuracy, fewer false positives, and reliable identification of minority attack classes in real-time settings by redefining feature selection and hyperparameter tuning as optimization problems and incorporating them into a deep learning-based classification pipeline.

## 1.1 Motivation and our research contribution

The rapid growth of IoT devices has enhanced connectivity but also increased security risks. Traditional safeguards often fail to detect anomalies in complex IoT networks. This study aims to improve IoT security by developing advanced ML and DL-based anomaly detection methods. Our goal is to enhance detection accuracy, reduce false positives, and create resilient IoT networks for individuals and organizations. In order to do this, we use an optimized ensemble deep learning scheme that combines fuzzy classification, imbalance handling, optimization-based feature selection, and efficient feature extraction. This integration simultaneously addresses accuracy, minority class detection, and computational efficiency.

The research contribution of this paper are as follows:

- To extract essential features, the CoffeeNet approach is used; the approach achieves superior feature extraction capabilities. This enhancement directly contributes to improved anomaly detection accuracy by effectively capturing essential data characteristics.

- The Binary Bamboo Forest Growth Optimization Algorithm (BBFGOA) is employed to reduce data characteristics and enhance detection performance.

- We introduced a Novel Optimized Fuzzy NFSC approach, which ensembles a Sharpened cosine similarity-based neural network (SCS-Net) and a rule-based deep fuzzy system with nonlinear fuzzy feature transform (NFFS-DFS) for

classifying network flows as benign or malicious.

• To optimize the hyperparameters, the Partial Reinforcement Optimization (PRO) algorithm is used.

• Integrating an optimized ensemble deep learning framework, our approach incorporates advanced feature extraction and selection techniques. This integration significantly boosts anomaly detection accuracy and efficiency in IoT environments, surpassing traditional methodologies.

The following is the structure of the following sections of the paper. A brief explanation of the previous publications is provided in the second section. A thorough summary of the suggested strategy is provided in Part 3. A result and explanation of the suggested method are given in Part 4. A conclusion and future scope are presented at the end of Part 5.

## 2 Related works

*This portion presents an organized review of the literature on various methods used in DL and Ml for detecting cyber assaults in traditional and IoT networks.*

**Deep Learning based approaches:** Bhavsar et al. [8] developed an IDS for detecting network anomalies based on a deep learning model called PCC-CNN. The PCC-CNN model integrates the significant characteristics acquired through linear-based extractions and the CNN approach. It uses binary categorization for detecting anomalies and multiclass categorization for different types of assaults. Abusitta et al. [1] presented a DL-based IDS detection system for IoT that can develop and extract robust and significant characteristics not impacted by unstable environments. The suggested structure was built around a denoising autoencoder currently used as a building block in Deep Neural Networks. The denoising autoencoder enables the accurate extraction of characteristics that are resistant to heterogeneous and distracting conditions common in IoT systems. In this research, Asgharzadeh et al. [4] developed the IoTFECNN approach, an IoT extraction of features that utilizes a convolutional neural network with hybrid layers for more accurate IoT detection of anomalies. Furthermore, a binary multi-objective improved Capuchin Search Algorithm (CSA) known as BMECapSA was employed to select features effectively. Integrating IoTFECNN and BMECapSA approaches resulted in developing an ensemble method known as CNN-BMECapSA-RF. Idrissi et al. [13] suggested the Fed-ANIDS method, a NIDS that uses AD and FL to tackle the issues of confidentiality related to centralized structures. They utilize various AD systems to detect intrusions, such as simple autoencoders, variational autoencoders, and adversarial autoencoders, to determine an intrusion score based on the normal traffic reconstruction error. Alghamdi and Bellaiche [2] suggested a cascaded wormhole detection technique for IoT networks based on federated deep learning and a Dynamic Trust Factor. The DTF depends on two trust attributes, while the CNN and LSTM models were trained to employ the federated strategy, ensuring data privacy and security at the node level. In order to distinguish between categories of normal and abnormal use patterns in an IDS for an Internet of Things system, Bhatia et al. [7] provide an anomaly detection technique (MFEW_Bagging). First, a hybrid feature selection method is employed to search for the best feature subset. This method is based on a combination of several filter-based approaches and a wrapper algorithm. Moreover, bagging an ensemble learning technique is employed for the ultimate classification into groups of normal and pathological use patterns. By using ensemble feature selection, the bias of individual approaches to feature selection is eliminated, and the best collection of relevant and non-redundant features is found.

**Machine Learning based approaches:** Bacha et al. [6] presented an IDS to defend against cyber assaults in IoT settings. The suggested approach employs kernel PCA to reduce the dimension of data characteristics and enhance the detection of anomalies effectiveness. Xu et al. [22] developed data has been pre-processed and filtered by employing various algorithms. Using the Synthetic Minority Oversampling Technique (SMOTE) algorithm and mutual information improves the training dataset's quality. Automated ML was additionally employed to identify the method with the best hyper-parameters for classifying the data. Vishwakarma and Kesswani [20] presented a two-phase IDS for IoT. In the initial phase, they divide the data into four sections based on data type (nominal, integer, binary, and float). They subsequently categorize them, employing various iterations of the Naive Bayes algorithm. Following that, they utilize majority voting to determine the final categorization result. In the subsequent phase, pass on data that behaved normally or benignly in the first phase and categorize them employing an unsupervised elliptic envelope. Alwaisi et al. [3] suggests a machine learning detection method intended to classify and identify resource-constrained assaults in Internet of Things devices. With the use of EdgeML and CloudML for detection, we view IoT devices as essential

parts of the edge and cloud computing continuum. With an emphasis on memory and energy consumption in Internet of Things applications, this paper compares various machine learning techniques. For both training and detection, we examine different machine learning approaches, such as cloud-based, edge-based, and device-based approaches. The assessment includes the use of various machine learning algorithms on cloud and edge devices, as well as small IoT devices. In IoT settings, an IDS is suggested Bacha et al. [6] to combat a variety of cyberattacks. They use the kernel extreme learning machine to classify the collection of attacks according to their particular type for multiclass categorization and to identify whether the traffic flow is malicious or benign for binary detection.

### 2.1 Research gap

The literature research found that, while the presented approaches perform well, the detection accuracy for each category in multiclass classification is low due to class imbalance. The model's training accuracy is strongly biased towards classes with more samples, emphasizing the importance of addressing this imbalance and improving accuracy across all classes. To accomplish this, we propose employing the GAN method to produce synthetic data for minority classes and lowering the number of characteristics by deleting irrelevant and duplicate ones to shorten the initial training period. Specifically, we employed a filter-based strategy to delete one of two strongly correlated features, reducing the deep learning model's training time while increasing accuracy.

## 3 Proposed methodology

The primary goal behind our detection of anomalies approach is to create an efficient model capable of identifying anomalous IoT data by learning the latent structure of normal instances through encoding and decoding.

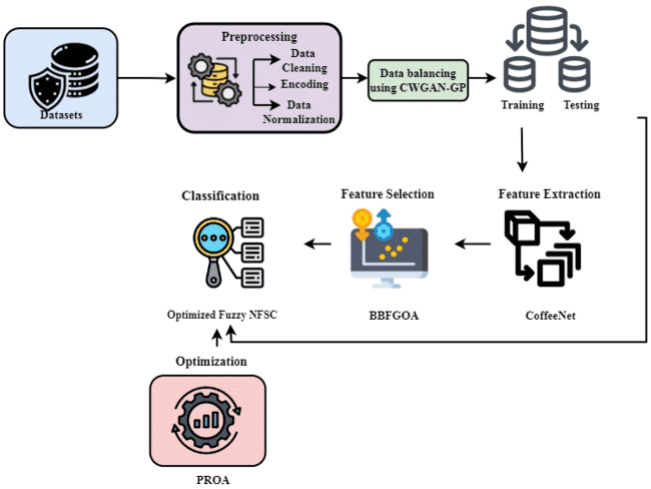


Figure 1: Framework of proposed methodology

The model should be able to distinguish anomalies from "normal" samples because anomalies lose information during encoding. Figure 1 depicts the architecture of the proposed model. The below subsections explain our proposed methodology in detail.

### 3.1 Data pre-processing

**Data cleaning:** Data cleaning includes detecting and correcting errors, insufficient structure, duplication, or missing values in a dataset. Errors may occur during data integration, and the lack of a standard cleaning methodology can result in inconsistency. Even when accuracy is achieved, inaccuracies in data may threaten the reliability of outcomes. A standardized data cleaning strategy is essential for maintaining accuracy and consistency.

**Categorical encoding:** Label encoding, a popular technique in deep learning, converts categorical data into numerical data. It involves providing an integer to every distinctive categorized variable so that algorithms can process data more efficiently. For instance, the set's absolute feature values are 'HTTPFlood', 'UDPFlood', and 'TCPFlood'.

This phase uses the encoded label to allocate numbers to the features as needed. One-hot encoding is a different approach for converting categorical data into numerical data; it represents categorized variables as binary vectors.

**Feature scaling:** Min-Max normalization is employed to normalize column values within a specified range of $[0, 1]$, allowing them to be displayed on a uniform scale.

$$y_{norm} = \frac{y - \min(y)}{\max(y) - \min(y)} \times (B - A) + A. \tag{1}$$

## 3.2 CTGAN for class imbalance

After pre-processing, the unbalanced data issue is resolved and network fitting is reduced using CTGAN-based data augmentation. It removes specific instances from groups that are majority and adds more to minority classes. It is a GAN-based architecture designed to synthesize tabular data. Addressing the problems with using GAN framework for tabulated data modeling is the main goal of CTGAN's features. The following formula represents the CTGAN loss function.

$$L = E_{g(z)\sim P_g}[S(g(z))] - E_{\sim P_r}[S(a)] + \lambda E_{y\sim P_y}\left[(\parallel \nabla_y S(y) \parallel -1)^2\right]. \tag{2}$$

In the provided formula, $g(z)$ indicates the output of the discriminator for a real data sample $x$, and $g$ represents the synthetic data produced by the generator using a noise vector. The distribution of actual data samples is represented by $P_g$, which also applies to the distribution of the created (false) data. When calculating the gradient penalty, the phrase explicitly refers $P_y$ to the distribution of interpolated data points between produced and real samples. $\nabla_y S(y)$ represents to the discriminator's output gradient in relation to the interpolated sample as well. The gradient penalty term's weight is controlled by the constant, a regularization parameter and it is defined as $\lambda$.

The tabular data $(T)$ provides the generator in CTGAN with randomized noise $(z)$ and conditioning information $(y)$. The conditioning information may indicate the range or kind of sampling in every row, among other characteristics of the final data. The generator then generates fictional samples, which are sent to the discriminator $(S)$ for evaluation. The discriminator gives a signal to the generator after deciding if the instance is genuine or fraudulent using backpropagation. The generator uses this hint to alter its weights and improves its capacity.

## 3.3 Feature extraction

The primary objective of extracting features is to transform raw data into a compact, informative representation that emphasizes crucial traits, allowing for more accurate and efficient anomaly detection. We used the CoffeeNet technique to extract features. CoffeeNet provides an advantage by efficiently capturing geographical and contextual information, enhancing detection performance while keeping a lightweight design that allows for speedier computation, making it perfect for real-time IoT applications.

### 3.3.1 CoffeeNet method for feature extraction

The proposed backbone framework for extracting features combines a CBAM attention block with a ResNet-50 CNN that achieves good accuracy by leveraging residual connections. ResNet-50 addresses the gradient vanishing issue common in deeper networks by introducing several residual blocks that allow for skipping some convolution layers without sacrificing accuracy. CoffeeNet architecture extracts features using 48 convolutional layers grouped into 5 stages. Residual blocks include $3 \times 3$ convolution layers, normalization, ReLU activation, and skip connections for identity mapping. These residual blocks assist in capturing and enhancing essential patterns through residual mapping.

The CBAM attention block improves feature extraction by directing the model's attention to critical visual features while minimizing unnecessary information. This block adaptively refines features by first inferring the attention map in spatial and channel dimensions, then multiplying it with the intermediate keypoints map to highlight important locations. This improves the model's accuracy in difficult settings such as fluctuations in shadow, brightness, and intensity, while decreasing computing weight by employing 64 channels in replaced convolution layers. The proposed backbone design has various advantages, including enhanced feature representation via the CBAM attention block, which prioritizes critical aspects for more efficient feature extraction. ResNet-50's residual connections alleviate the gradient vanishing problem, allowing for more stable training of deeper networks. The CBAM mechanism emphasizes essential areas, improving accuracy in complex fluctuations such as shade and light. Furthermore, using 64 channels in

replaced convolution layers saves computing costs, while residual blocks allow efficient training by omitting unneeded convolution layers without compromising performance.

## 3.4 Feature selection

The goal of feature selection is to discover and keep just the most relevant characteristics in a dataset, thereby reducing dimensionality, minimizing computing complexity, and improving overall model performance by deleting redundant or irrelevant information. BBFGOA's unique optimization capabilities maintain only the most essential features, increasing detection efficiency and model robustness in IoT contexts.

Recent research has shown that optimization-driven approaches are successful in resolving challenging detection and classification problems. For instance, an adaptive centroid refinement strategy was developed by the Centroid update approach to K-means clustering (AECE) [9], which increases convergence speed and clustering accuracy. Similar to this, feature selection has been accomplished with success using the Binary Anarchic Society Optimization (BASO) algorithm [11], which outperforms conventional evolutionary techniques in terms of minimizing redundancy while maintaining classification accuracy. Furthermore, the synergy between optimization and neural models has been demonstrated by Neural Network-based Pattern Recognition in edge computing frameworks (RJIST) [10], which achieves superior learning and classification effectiveness in resource-constrained contexts. Our work offers a novel and competitive feature selection technique that is especially suited for intrusion detection issues in SDN and IoT-based systems by placing BBFGOA alongside these latest developments. A binary optimization problem can be used to formulate the feature selection task that BBFGOA addresses. Finding the ideal subset of features that minimizes the number of features chosen while optimizing the effectiveness of classification is the goal. The following cost (fitness) function is used to illustrate this:

$$\text{fitness}(x) = \alpha \cdot (1 - ACC(x)) + (1 - \alpha) \cdot \frac{|x|}{D}. \tag{3}$$

While $ACC(x)$ indicates the effectiveness of classification using the chosen feature subset, $D$ denotes the overall amount of features, $\alpha$ balances accuracy and feature reduction, and $|X|$ represents the amount of selected features. The binary vector that represents the decision variables is as follows:

$$x_i = \begin{cases} 1 & \text{if feature is selected} \\ 0 & \text{otherwise} \end{cases}$$

Constraints: Every solution must be within the search space. To guarantee a valid subset, at least one feature needs to be chosen. In order to improve the effectiveness and resilience of intrusion detection in IoT and SDN-based systems, BBFGOA aims to solve this optimization problem by striking the ideal balance among feature reduction and classification accuracy.

### 3.4.1 BBF strategy

Feature selection ($FS$) is a popular dimensionality reduction strategy for $DL$ that improves model computing efficiency. This section focuses on how to select the features using bamboo forest growth. Bamboo, one of the fastest-growing plants, grows by bamboo whips that spread horizontally, creating roots and shoots that develop into new lashes or bamboo poles, eventually forming a bamboo forest.

### 3.4.2 Extension of the Bamboo whip

This algorithm incorporates clustering into the algorithm, motivated by the connection between bamboo forests and bamboo whips. During optimization, members are grouped based on memes, with dynamic modifications among uniform and random grouping. Uniform grouping ranks individuals according to their fitness to establish meme-based clusters, whereas random grouping transfers members when group efficiency plateaus. The bamboo whip extension direction is affected by group cognition, whip memories, and center optimal. In formulas (4) and (5)-(7) contain the center location and extension direction formula.

$$d(k) = \frac{1}{n} \sum_{i=1}^{n} x(k_i). \tag{4}$$

$$\cos(\alpha) = \frac{x_t - x_G}{|x_t| \times |x_G|}. \tag{5}$$

$$\cos(\beta) = \frac{x_t \cdot x_p(k)}{\|x_t\| \cdot \|x_p(k)\|}. \tag{6}$$

$$\cos(\gamma) = \frac{x_t - x(k)}{|x_t| \times |x(k)|}. \tag{7}$$

Wherein $(k_i)$ indicates the position of the bamboo, then k indicates the whip of bamboo, $x_t$ represents the optimal shoot bamboo position.

### 3.4.3 Shoot growth

Trees, including bamboo, grow in response to various random causes, making the process unpredictable. These elements have varying impacts and are challenging to measure or characterize properly. Therefore, tree growth is frequently described as an arbitrary process. Stochastic process analysis to simulate bamboo shoot growth was employed, which has two stages: slow growth and quick growth.

$$D_t = D_G \times e^{\frac{B}{\omega \times e^{\omega}}}. \tag{8}$$

The bamboo shoot growth stage takes approximately 55 days and is separated into two parts: the first $1-25$ days display comparatively mild growth, while the $25-55$ day period shows explosive growth. Here, $D_G$ represents the total bamboo height in a given environment, which varies with conditions; $B$ is a randomized bamboo measuring factor; and $\omega$ is the shape parameter independent of environmental factors. The formula (9) depicts the calculation Formula based on the incremental computation of changes over time. In the multidimensional instance, the formula (9) produces a vector.

$$\Delta H = \frac{d(t) - d(t-1)}{D_G - c(k) + 1}. \tag{9}$$

$c(k)$ is the central position of the bamboo forest. The formulas (10) and (11) show whether particular bamboo shoots renew at this point. In the multidimensional instance, the formula (11) produces a vector.

$$d_{temp} = \begin{cases} d_t + D_D \times \Delta H & \text{rand} < 0.5 \\ d_t - D_D \times \Delta H & \text{else} \end{cases} \tag{10}$$

$$D_m = 1 - \left| \frac{d_t - c(k)}{D_G - c(k) + 1} \right|. \tag{11}$$

### 3.4.4 BBFGO strategy

The typical BFOA algorithm has constant choices and can update Formulas without restrictions; however, for selecting features, the space of search must be structured.

### 3.4.5 BBFGO with transfer functions

In this investigation, several transfer functions are employed. Transfer functions are divided into three classes: S-shaped, V-shaped, and Taper-shaped. The primary function of these transfer function is to calculate the likelihood of upgrading an element's value to 1 or 0. In FS, the outcome is either zero or one.

### 3.4.6 S-shaped transfer function

The formulas (12) and (13) determine the transfer vector $(S)$ in the shape of a $S$.

$$S_1(x_{m,j}^t) = \frac{1}{1 + e^{-x_{m,j}^t}}. \tag{12}$$

$$S_2(x_{m,j}^t) = \frac{1}{1 + e^{-10*(X_{m,j}^t - 0.5)}}. \tag{13}$$

Formula (14) yields $X_{m,j}^t$, which is further modified by the formula.

$$(x_{m,j}^t) = \begin{cases} 1 & \text{rand} < S_i(x_{m,j}^t) \\ 0 & \text{rand} \geq S_i(x_{m,j}^t) \end{cases} \tag{14}$$

#### 3.4.7 Relation to optimization problem

The specified feature selection optimization issue can be solved directly using the equations shown in Sections 3.4.2-3.4.6. In particular, by employing group cognition, memory, and cluster centers, the whip extension and center update equations (4)-(7) specify how the position of a candidate solution (feature subset) changes in the direction of the ideal subset. By simulating stochastic exploration, the shoot growth modeling equations (8)-(11) allow candidate solutions to diversify and avoid premature convergence. In order to connect the algorithm's search process with the binary nature of feature selection, the transfer functions in equations (12)-(14) finally translate the continuous location updates into binary decisions (0 or 1). This relationship can be summed up mathematically as follows:

$$x^{t+1} T\left(f\left(x^t, \vec{G}, \vec{M}, \vec{C}\right)\right). \tag{15}$$

While $\vec{G}$, $\vec{M}$ and $\vec{C}$ stand for group cognition, memory, and centers; $f(.)$ encodes the whip extension and shoot growth processes (Eqs. (4)-(11)); and $T(\cdot)$ indicates the transfer function (Eqs. (12)– (14)) that converts the solution into binary space. $x^t$ denotes the current feature subset. This approach makes sure that every BBFGOA update directly translates into the creation and improvement of candidate subsets of features until the ideal balance between dimensionality reduction and classification accuracy is reached.

### 3.5 Classification

The Optimized Fuzzy NFSC method aims to accurately categorize network flows as benign or malicious by combining neural network-based and fuzzy logic techniques. This strategy combines the benefits of both approaches, resulting in higher classification accuracy, better handling of uncertainties, and an increased ability to detect complex attack patterns in IoT contexts.

#### 3.5.1 SCS-Net for categorization

Four layers make up the SCS-Net architecture: a flattened, dense layer, a max-pooling layer, and a cosine similarity layer. The framework is depicted in Figure 2. The SCS-Net model is built using the training data. We used the SCS layer to create the features map rather than a convolution layer. SCS is a neural network feature-building method that offers an alternative to convolution. SCS (or comparable convolution) is a statistical technique for data feature extraction. Nevertheless, cosine similarity results from scaling the patch and kernel to one after generating the dot product. Formula 16 calculates the cosine of the angle formed by the kernel's vectors and the two-dimensional input.

$$\cos(\theta) = \frac{m \cdot n}{\|m\|\|n\|}. \tag{16}$$

$$SCD(M, N) = \left(\frac{m \cdot n}{(\|m\| + Q)(\|n\| + Q)}\right)^p. \tag{17}$$

$$SCS(M, N) = Sign(M, N)\left(\frac{m, n}{(\|m\| + Q)(\|n\| + Q)}\right)^p. \tag{18}$$

Input is represented as m, Kernel is indicated as n, Invariance of scale is defined as $Q$, and $P$ = Sharpening Exponents. SCS for example, extracts features from images using stride operations similar to convolution. SCS provides various benefits for anomaly categorization. It effectively handles unscaled inputs, which improves the model's adaptability.

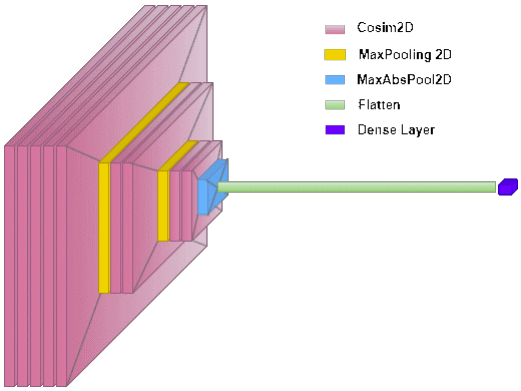


Figure 2: Overall framework of SCS-Net

The architecture is simple and parameter-efficient, enhancing accuracy with an optimal set of model parameters. Unlike other structures, SCS layers lack extra activation, normalization, or dropout layers, which reduces model complexity.

**Layers**

Pooling is the procedure of reducing the dimension of a dataset. This filter is improved throughout backpropagation till it finds an appropriate threshold for identifying the properties in the given data. Formula (19) depicts the maximum-pooling formula. Pooling is the procedure of reducing the dimension of a dataset. This filter is improved throughout backpropagation till it finds an appropriate threshold for identifying the properties in the given data. Formula (19) depicts the maximum-pooling formula.

$$\text{Abs Max Pooling} = \left|\left(\left(N + 2P - \ -\frac{F}{S}\right) + 1\right)\right|. \tag{19}$$

While $N$ indicates data size, $P$ is for padding, $F$ is for filter size, and $S$ is for stride. The flattened layer transforms data into a multidimensional column vector, which is then sent into the dense layer as input. The Dense layer comprises completely connected neurons that use Softmax activation to identify several classes and sigmoid for binary categorization.

### 3.5.2 NFFT-DFSC

This stage describes the NFFT-DFSC, an initial rule-based deep fuzzy system for data classification.

**System framework**

The structure consists of $d$ ($d \geq 1$) NFFT and a single MPRFD building block. These building components are connected by employing the Sigmoid and BN processes. Employing a sigmoid function interwoven with BN operations, the transferred data is compressed into another unit of hidden fuzzy space for features. The general fuzzy latent qualities implied in the input data can be acquired by continually transferring them across many NFFT construction pieces. After that, the MPRFD component gets these high-level fuzzy characteristics and makes decisions based on the numerous prototype presentations for every class. Whenever data from the input is fed into the method, it is initially transmitted and compressed constantly.

$$\hat{x}_{i,j}^{l} = \frac{1}{1 + \exp\left(-BN\left(\bar{x}_{i,j}^{l}\right)\right)}. \tag{20}$$

$$\bar{x}_{i,j}^{l} = \frac{\mu_{A_j^{l(\hat{x}_i^{l-1})}}}{\sum_{t=1}^{M_i} \mu_{A_j^{l(\hat{x}_i^{(l-1))}}}}. \tag{21}$$

$$\mu_{A_j^l}\left(\hat{x}_i^{l-1}\right) = \exp\left(-\sum_{q=1}^{M_{t-2}} \frac{\left(\hat{x}_{i,q}^{l-1} - s_{j,q}^{l}\right)^2}{2(\sigma_{j,q}^{l})^2}\right)$$
$$= \exp\left(-\frac{1}{2} \wedge_1 \left[(\hat{x}_i^{l-1} - s_{j,q}^{l})^2 \oslash (\sigma_j^l)^2\right]\right). \tag{22}$$

$$BN(\bar{x}_{i,j}^{l}) = \left[P_j^l \frac{\bar{x}_{i,j}^{l} - B_j^l}{\sqrt{(\delta_j^l)^2 + \varepsilon}} + \beta_j^l\right]. \tag{23}$$

The element wise squaring of a vector is represented by $(\cdot)^2$, the element-wise division of vectors is represented by $\oslash$, and the $\wedge_1$ is a dimensional vector of ones. Furthermore, when $l = 1$ in (20) and (21), and $\hat{x}_i^{l-1} = 0$. The BN operation is expressed by $BN(\cdot)$, and the median and standard deviation of each sample used to train the $l^{th}$ NFFT building block are denoted by $p_j^l$ and $\sigma_j^l$, accordingly. The parameters to be learned during training are $p_j^l$ and $\beta_j^l$. In order to prevent it from being split by zero, the constant $\varepsilon$ is often set at $1e-8$. In order to address the gradient vanishing issue and enhance the model's generalization, the BN approach which has its origins in the normalizing of the data distributions for every mini-batch throughout the DNN training process is incorporated into the suggested NFFT-DFSC.

In order to mitigate the gradient vanishing problem and enhance the model's adaptability, the proposed integrates a BN approach.

$$\bar{k}_{i,k} = \frac{k_{i,k}}{\sum_{r=1}^{k} k_{i,r}}. \tag{24}$$

$$\mu_{B_{k,m}}(\hat{x}_i^d) = \exp\left(-\sum_{h=1}^{M_d} \frac{\left(\hat{x}_{i,h}^{d} - \nu_{k,m,h}\right)^2}{2\psi_{k,m,h}^2}\right) \tag{25}$$
$$= \exp\left(-\frac{1}{2} \wedge_2 \left[\left(\hat{x}_i^d - \nu_{k,m}\right)^2 \oslash \psi_{k,m}^2\right]\right). \tag{26}$$

Moreover, $m = 1, 2, \ldots, Nk$. The $\wedge_2$ is a $M_d$-dimensional vector of ones.

The Partial Reinforcement Optimization Algorithm (PRO) was used to automatically optimize the fuzzy logic parameters of the Optimized Fuzzy NFSC, such as the neighborhood size, decision thresholds, and fuzzification coefficient. Standard guidelines from the literature were used to determine the initial values: the neighborhood size was set to 10, the decision threshold was set to 0.5, and the fuzzification coefficient was set to 2.0. After that, PRO iteratively improved these parameters by comparing candidate configurations to a number of performance metrics, such as detection rate, F1-score, and classification accuracy. This process ensures that the decision boundaries and fuzzy membership functions are adaptively calibrated to the features of every dataset, leading to reliable and accurate detection, especially for minority-class attacks.

## 3.6 Hyperparameter tuning using PRO algorithm

The PRO algorithm is used to optimize the hyperparameters of the ensemble NFSC technique, providing that the model performs optimally. PRO improves the accuracy, convergence speed, and generalization ability of the NFSC ensemble by efficiently tweaking critical hyperparameters, hence increasing its ability to identify network flows correctly in IoT contexts. This portion will go over the PRO algorithm in detail. Initially, the algorithm's preliminary steps will be stated, detailing how it was motivated by and modelled after the PRE theory.

### 3.6.1 PRO algorithm

To relate the concepts and ideas of PRE theory with the elements of the PRO approach, the PRE concept is presented as an optimizing algorithm with some basic assumptions: A pupil represents a people or species whose actions require training or improvement using PRE theory, which is depicted as a solution. The pupil's activity is employed as an option factor; therefore, a solution (learner) is characterized by a set of decision variables.

In the PRO algorithm, a population is made up of learners. Fitness function assessments: The fitness for every learner's behaviours $Xi$ is determined utilizing a defined by user objective function on the decision factors (behaviours).

The suggested algorithm employs a scoring mechanism so that an action with an improved rating is more likely to be reinforced in subsequent iterations. A schedule is an idea of the manner in which actions should be rewarded, as represented by a data structure at various intervals. Furthermore, the variable-interval rescheduling approach is regarded as an adaptive structure for stochastic analysis in the formulas (27)-eqrefR29. Every iteration, learner $i$ selects a subset of behaviours $\mu \subseteq \{1, 2, 3, \ldots, N\}$ with the most significant preferences in scheduling $i$. First, prioritize the schedule in descending order. Then, use Eq. (27) to select the first $\lambda$ total items as possible behaviours.

$$\tau \longleftarrow \frac{FEs}{MaxFEs}. \tag{27}$$

$$SR \longleftarrow e^{-(1-\tau)}. \tag{28}$$

$$\mu \subseteq \{1, 2, 3, \cdots, N\} | \forall j \in \mu,\ SCHEDULE^{j} \geq SCHEDULE^{*,\lambda}$$
$$\lambda \longleftarrow \{\|\mu\| \| \mu\| = [U(1, N \times SR]\}. \tag{29}$$

In this Formula, $\tau$ represents the factor of time, FEs indicate the amount of function assessments, and MaxFEs represent the maximum quantity of function assessments. SR represents the rate of selection, $\mu$ is a scheduling-based subset of behaviours, size of the chosen subset is indicated as $\lambda$, and $N$ indicates the overall amount of behaviours. $SCHEDULE*$ represents a prioritized schedule, while $SCHEDULE*$ is the $\lambda^{th}$ item.

Stimulation: Operations are used to symbolize every effort to elicit a response from a learner's behaviors. Applying operations that alter the decision variables of a suggested solution models any attempt to elicit a response from a learner. It should be mentioned that any operation can be used to encourage (modify) the decision-making behaviors of students. According to Eqs. (30)-(33), the PRO algorithm uses the following actions to produce new solutions.

$$SF_i \longleftarrow \tau + \mu(0, \bar{\beta}),\ \text{where } \bar{\beta} \longleftarrow \sum_{j \in \mu} \left( \frac{SCHEDULE_{i,j}}{\max(SCHEDULE_i)} \right). \tag{30}$$

$$S_i^{\mu} \longleftarrow \begin{cases} (x_{best}^{\mu} - x_i^{\mu}) & \text{if rand} < 0.5 \\ (x_i^{\mu} - x_j^{\mu}) & \text{otherwise} \end{cases} \tag{31}$$

$$x_{i,new}^{\mu} \longleftarrow x_i^{\mu} + SF_i \times S_i^{\mu}. \tag{32}$$

While, $SFi$ represents the factor of stimulation and $\bar{\beta}$ is defined as the score of normalization.

**Reinforcement:** We utilize the technique to upgrade the schedule.

$$SCHEDULE_i^{\mu} \longleftarrow SCHEDULE_i^{\mu} \times RR. \tag{33}$$

Whereas $RR$ represents the reinforcement rate, and $Schedule_i^{\mu}$ indicates the priority.

In contrast, negative reinforcement is used when there is no reaction. In this case, a learner's objective function drops following the stimulation period, reducing the score for specified behaviour (Eq. 34). The components related to decisions with the highest scores will be chosen for stimulating and reinforcing in the following iteration.

$$SCHEDULE_i^{\mu} \longleftarrow SCHEDULE_i^{\mu} - (SCHEDULE_i^{\mu} \times RR). \tag{34}$$

Rescheduling: In this case, the PRO utilizes the schedule's average variance (Std) to determine when the pupil has to be scheduled. The formula (35) and (36) are used to implement this strategy.

$$SCHEDULE_i \longleftarrow \begin{cases} U(0,1) & \text{if } std(SCHEDULE_i) = 0 \\ \text{Do nothing} & \text{otherwise} \end{cases} \tag{35}$$

$$x_i \longleftarrow \begin{cases} U(L_B, U_B) & \text{if } std(SCHEDULE_i) = 0 \\ \text{Do nothing} & \text{otherwise} \end{cases} \tag{36}$$

The deviation from the mean of the $i^{th}$ learner's schedule is denoted by $std(SCHEDULE_i)$, while $L_B$ and $U_B$ represent the lower and upper bounds. Additionally, $(L_B, U_B)$ are random variables with a uniform distribution among $(0, 1)$. The proposed pseudocode is given below.

### 3.7 Linking theory to application:

From the perspective of theory, the suggested ensemble deep learning method is based on well-established anomaly detection and machine learning concepts. The Sharpened Cosine Similarity-based Neural Network (SCS-Net) and the rule-based Deep Fuzzy System with Nonlinear Fuzzy Feature Transform (NFFS-DFS) are two examples of complementary models that can be combined to improve generalization and robustness, decrease overfitting, and detect complex anomalies in IoT network traffic. The fuzzy system's ability to capture imprecision and uncertainty in network data is consistent with fuzzy logic theory, which holds that these methods enhance decision-making in ambiguous or noisy settings. According to theory, feature selection with CoffeeNet and the Binary Bamboo Forest Growth Optimization Algorithm (BBFGOA) enhances learning effectiveness and model interpretability by reducing dimensionality and removing redundant features. In accordance with optimization theory, hyperparameter adjustment by the Partial Reinforcement Optimization Algorithm (PRO) ensures convergence to optimal solutions. Lastly, theoretical insights that balanced datasets enhance classifier performance, especially for minority classes, encourage the use of a Conditional Tabular Generative Adversarial Network (CTGAN) to address class imbalance. When combined, these theoretically driven design decisions offer a solid basis for the proposed IDS's dependability and efficacy.

**Pseudo code:**

Algorithm Proposed_IDS_Framework
Input: Raw IoT/SDN dataset D
Output: Classified attack categories
**1: # Data Preprocessing**
2: Clean missing and inconsistent values in D
3: Encode categorical attributes
4: Normalize features $\rightarrow$ $D$_preprocessed
**5: # Data Balancing**
6: Apply CWGAN-GP to handle class imbalance
7: Obtain balanced dataset $D$_balanced
8: Split $D$_balanced into Training set ($D$_train) and Testing set ($D$_test)
**9: # Feature Extraction**
10: Extract high-level features using CoffeeNet
11: F $\longleftarrow$ Extracted feature set
**12: # Feature Selection via BBFGOA**
13: Initialize population $P$ of feature subsets from $F$
14: while termination condition not met do
15: Evaluate fitness of each subset:
16: Fitness = $\alpha^*$(1 - Accuracy) + $\beta^*$(|Subset|/|$F$|)
17: Update population using BBFGOA operators
18: end while
19: Select optimal feature subset $F^*$ with best fitness
**20: # Classification**
21: Train Optimized Fuzzy NFSC classifier using $F^*$ and $D_{\text{train}}$
22: Use PROA for hyperparameter optimization
23: Predict labels $L \longleftarrow$ Classifier($D$_test, $F^*$)
**24: # Evaluation**
25: Compute Accuracy, Precision, Recall, $F$1-score
26: Return $L$

## 4 Result and discussions

The experimental results obtained from using the theoretically based methodology outlined in Sections 2 and 3 on four benchmark IoT intrusion detection datasets are shown in this part. In order to ensure that the reported metrics accurately reflect the impact of these design decisions, the feature extraction stage (CoffeeNet), feature selection (BBFGOA), class imbalance handling (CTGAN), and classification (Optimized Fuzzy NFSC with PRO tuning) were all implemented precisely as outlined in the previous sections. Accuracy, precision, recall, F1-score, and detection rate

are used to examine the data, and performance patterns are explored in relation to the previously covered theoretical concepts and optimization techniques. Table 1 provides a summary of the parameter setting for the categorization step.

Table 1: Parameter setup and their values

| **Parameter** | **Values** |
|---|---|
| Epoch | 200 |
| Batch Size | 64 |
| Learning rate | 0.001 |
| Dropout Rate | 0.2 - 0.5 |
| Optimizer | PRO |
| $L1$ Regularization | 0.01 |

Only the classification hyperparameters of the Optimized Fuzzy NFSC are tuned by the proposed optimization method (PRO). The search ranges for Epochs, Batch Size, Learning Rate, Dropout Rate, and L1 Regularization are examples of user-defined parameters (Table 1). During PRO's stochastic investigation of potential hyperparameter values within these ranges, random parameters are added. The candidate configurations are algorithm-controlled parameters that are iteratively modified using the multi-objective fitness function that combines detection rate, accuracy, and F1-score.

## 4.1 Experimental setting

It runs the latest version of Windows on an Intel Corporation i5 2.60GHz processor with 32 GB of RAM. Python, KERAS, and TensorFlow are used in the assessments conducted on the Anaconda 3 platform. Table 2 illustrates the experimental setup.

Table 2: Environmental setup

| **Project** | **Environment** |
|---|---|
| Operating System | Windows 10 |
| Memory | 32 GB |
| Framework | Keras |
| CPU | i3-7100U |
| Software | Python version 3.11 |

## 4.2 Dataset description

**BoT-IoT:** It has developed and been recognized for multi-class applications. The label's characteristics included an attack flow, a category, and a subcategory. BoT-IoT has 46 attributes, in total, consist of the target variable.

**IoT-23:** It consists of real-time traffic collected by the vast AIC laboratory in partnership with the Czech University of Technology in Prague. IoT-23 includes twenty malicious software grabs from different IoT devices and three catches of benign abnormalities.

**IoTID20 Dataset:** The IoTID20 dataset includes both normal (benign) traffic and several IoT assaults. IoTID20 is a data collection compiled from smart home IoT ecosystems. Nmap tools were used to mimic assaults such as DoS, scanning, and man in the middle. Mirai botnet assaults were created on a laptop and then modified to represent their origin on IoT.

**N-BaIoT dataset:** The N-BaIoT collection provides traffic statistics from nine IoT devices. These statistics include both normal traffic and other forms of cyber-attacks. Hazardous cyber-attacks are classified into ten types, two of which are botnets.

## 4.3 Evaluation metrics

Any effective model must include an evaluation component. The following are some of the factors that are considered to assess the model's effectiveness.

**Accuracy (Acc):** The framework indicates which algorithm best spots patterns among variables.

$$ACC = \frac{Tn + Tp}{Tn + Fn + Tp + Fp}. \tag{37}$$

**Precision (Prec):** It estimates the amount of correct optimistic predictions made by a method compared to accurate optimistic forecasts. The Formula goes as follows.

$$Prec = \frac{Tp}{Fp + Tp}. \tag{38}$$

**Recall (Rec):** It indicates the strategy's optimistic forecasts compared to the TPR in the data.

$$Rec = \frac{Tp}{Fn + Tp}. \tag{39}$$

**F1-Score:** It represents the weighted average of recall and precision.

$$F1 - S = \frac{2 * prec * Rec}{prec + Rec}. \tag{40}$$

**ROC-AUC Curve:** It describes how well the framework can distinguish among the categories. The greater the AUC, the more accurate the model. The curve shows the TPR versus the FPR.

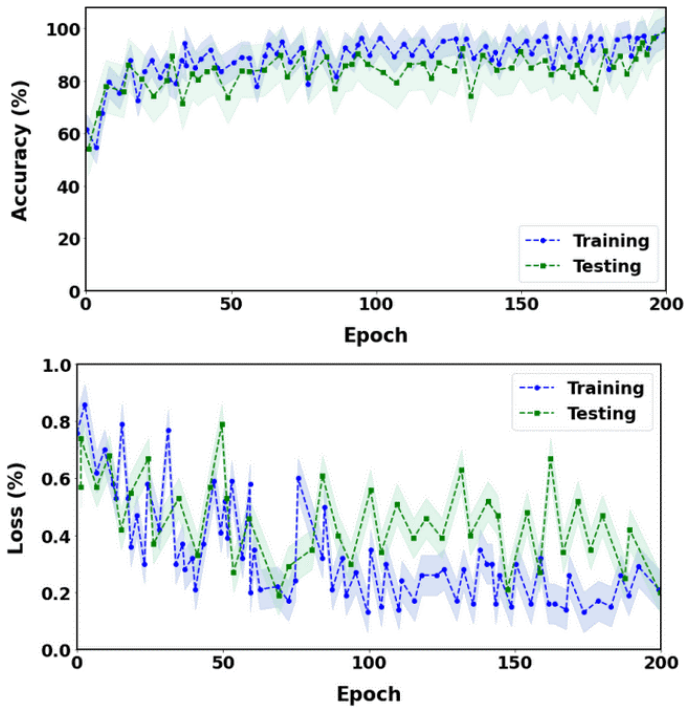


Figure 3: Evaluation of training and testing accuracy and their losses

Figure 3 displays the training phase and testing evaluation of our suggested dataset. More than 200 iterations were taught. A learning rate of 0.01 was established.

## # Experimental analysis on dataset 1 (BoT-IoT)

This section presents an experimental analysis using the BoT-IoT dataset, which simulates various attack scenarios in IoT environments.

Table 3: Differentiation using the BoT-IoT dataset

| **Method** | **Acc** | **Prec** | **Rec** | **F1-S** |
|---|---|---|---|---|
| Improved-IDS | 99.33% | 97.36% | - | 99.33% |
| RNN | 99.74% | 99.99% | 99.74% | - |
| LSTM | 99.74% | 99.99% | 99.75% | - |
| FNN | 96.19% | 98.9% | 99.8% | 99.4% |
| RNN and BiLSTM | 99.55% | 99.99% | 99.02% | 99.49% |
| Stacking | 99.92% | 99.86% | 99.78% | 99.82% |
| Proposed | 99.94% | 99.88% | 99.90% | 99.89% |

Table 3 present the metrics of various methods assessed employing the BoT-IoT dataset. The Proposed Method achieved the accuracy at 99.94%.

Table 4: Contrasting imbalance approach using BoT-IoT dataset

| | | **GAN** | | |
|---|---|---|---|---|
| **Category** | **Acc** | **Prec** | **Rec** | **F1-S** |
| Normal | 98.98% | 99.47% | 99.57% | 99.86% |
| Theft | 99.14% | 99.77% | 98.33% | 99.11% |
| Scan | 99.25% | 99.66% | 94.76% | 98.78% |
| DOS | 98.71% | 99.53% | 94.36% | 99.84% |
| | | **DCGAN** | | |
| Normal | 98.99% | 99.05% | 98.49% | 98.89% |
| Scan | 99.23% | 98.46% | 99.27% | 99.12% |
| DOS | 98.41% | 98.85% | 98.13% | 98.28% |
| Theft | 99.47% | 99.12% | 99.01% | 99.71% |
| | | **Proposed** | | |
| Normal | 99.23% | 99.82% | 99.36% | 99.49% |
| DOS | 99.08% | 99.57% | 99.21% | 99.36% |
| Scan | 98.99% | 99.60% | 99.14% | 99.28% |
| Theft | 99% | 99.21% | 99.02% | 99.11% |

Table 4 presents an analysis of data imbalances. In comparison to current methods, the suggested methodology achieves better results.

Table 5: Attack-wise training and testing differentiation

| **Attacks** | **Training Time** | **Testing Time** |
|---|---|---|
| Normal | 0.012 | 0.0009 |
| DOS | 0.025 | 0.0014 |
| Scan | 0.0143 | 0.0082 |
| Theft | 0.009 | 0.0006 |

Table 5 displays the training and testing times for various assaults.

"Normal" took 0.012s (training) and 0.0009s (testing); "DOS" took 0.025s and 0.0014s; "Scan" took 0.0143s and 0.0082s; and "Theft" took 0.009s and 0.0006s. DOS required the most training time while stealing required the least testing time.

### # Experimental analysis on dataset 2 (IoTid 20 dataset)

In this section, we conduct an experimental analysis using the IoTid 20 dataset, which encompasses diverse attack types specifically designed for IoT networks.

We aim to determine the effectiveness of the proposed methods in identifying and mitigating threats within IoT environments while examining their training and testing times to ensure practical applicability in real-world scenarios.

Table 6: Performance differentiation using IoTid 20 dataset

| Approach | Prec | Acc | F1-S | Rec |
|---|---|---|---|---|
| LSTM | 95.43% | 95.84% | 93.55% | 92.01% |
| GRU | 95.76% | 96.81% | 95.19% | 94.68% |
| DNN | 93.40% | 95.47% | 93.67% | 94.47% |
| DBN | 94.30% | 95.89% | 94.69% | 95.49% |
| AE | 95.15% | 96.44% | 94.56% | 94.40% |
| MLP | 89.33% | 92.38% | 85.29% | 84.36% |
| DT | 97.44% | 97.70% | 97.41% | 97.37% |
| LR | 77.28% | 83.14% | 73.11% | 72.97% |
| NB | 66.28% | 67.72% | 64.79% | 73.81% |
| SVM | 84.16% | 85.57% | 78.83% | 78.45% |
| KNN | 97.46% | 97.93% | 97.22% | 96.99% |
| DCNN | 97.13% | 98.12% | 97.46% | 97.83% |
| Proposed | 99.18% | 99.43% | 99.20% | 99.23% |

Table 6 outline the performance metrics of various models evaluated on the IoTid 20 dataset.

The Proposed Model stands out with the accuracy of 99.43%, alongside impressive precision (99.18%), recall (99.23%), and F1-score (99.20%), indicating its superior capability in detecting anomalies.

Table 7: Attack wise time comparison

| Attacks | Training (S) | Testing (S) |
|---|---|---|
| Dos | 0.045 | 0.0031 |
| MiralAck-Flooding | 0.0126 | 0.0092 |
| Miral Brute Force | 0.0132 | 0.0088 |
| Miral HTTP Floading | 0.062 | 0.0019 |
| Miral UDP Flooding | 0.031 | 0.0009 |
| MITM | 0.022 | 0.0013 |
| Scan Host Port | 0.0144 | 0.0073 |
| Scan port os | 0.042 | 0.0022 |

Table 7 contrasts training and testing timeframes for various attacks.

"DoS" had the longest training time (0.045 seconds), whereas "Mirai UDP Flooding" had the smallest testing time (0.0009 seconds). "Mirai HTTP Flooding" took longer to train at 0.062 seconds, while "Mirai ACK-Flooding" tested in 0.0092 seconds. Other attacks, such as "MITM," "Scan Host Port," and "Scan Port OS," required intermediate training and testing times, with "Scan Host Port" taking 0.0073 seconds to test.

### # Experimental analysis on dataset 3 (IoT 23 dataset)

This section focuses on the IoT 23 dataset, which provides a rich set of attack types pertinent to IoT networks. We analyze the performance of various detection models specifically tailored to this dataset, highlighting their strengths in identifying attack vectors. The Figure 4 presents the performance metrics of various methods evaluated on the IoT23 dataset, focusing on accuracy, precision, and recall. The Proposed Method achieved the accuracy at 99.94%.

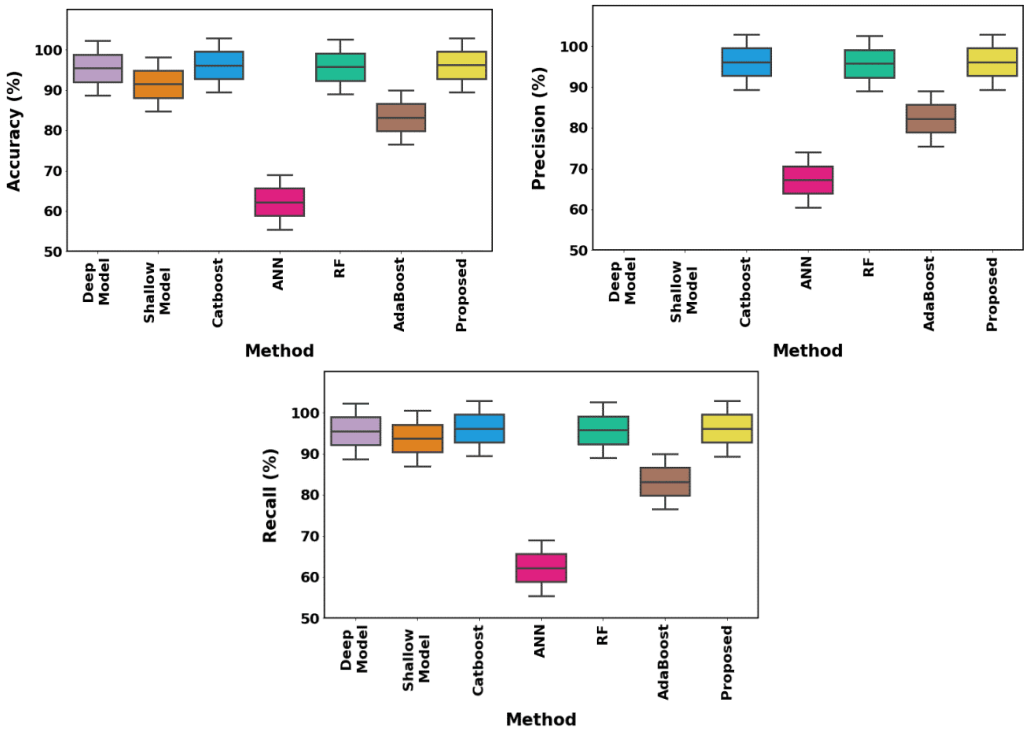


Figure 4: effectiveness of using dataset IoT23

Table 8: Attack wise time comparison

| **Attacks** | **Training (s)** | **Testing (S)** |
|---|---|---|
| C 0.0085 | C | 0.0111 |
| DDOS | 0.025 | 0.0011 |
| File Down | 0.031 | 0.0022 |
| Heartbeat | 0.0132 | 0.0074 |
| Mirai | 0.018 | 0.0009 |
| Okiru | 0.023 | 0.015 |
| Torii | 0.0127 | 0.0023 |
| Horizontal | 0.016 | 0.0012 |
| Port Scan | 0.0201 | 0.0106 |

The training and testing times of attack categories are shown in Table 8.

## # Experimental analysis on dataset 4 (N-BaIoT dataset)

In this section, we explore the N-BaIoT dataset, designed to address specific challenges in IoT security. We consider the training and testing times of these models to assess their operational efficiency in real-world IoT environments.

Table 9: Performance differentiation using the N-BaIoT dataset

| **Approaches** | **Acc (%)** | **Prec (%)** | **Rec (%)** | **F-S (%)** |
|---|---|---|---|---|
| Bat algorithm | 90 | 85 | 90 | 86.6 |
| Ensemble Naïve Bayes | 99 | - | - | - |
| KELM | 99.4 | 98.8 | 99.25 | 99.05 |
| Proposed | 99.58 | 99.31 | 99.26 | 99.28 |

The Table 9 summarizes the performance metrics of various approaches evaluated on the N-BaIoT dataset. This

indicates its effectiveness in detecting anomalies within IoT environments. The KELM method also performed well, with a slightly higher accuracy of 99.4%, alongside high precision and recall values, reflecting its reliability. In contrast, the Ensemble Naïve Bayes approach achieved a high accuracy of 99% but did not provide precision or recall metrics.

Table 10: Training and testing time differentiation of N-BaIoT dataset

| Attack | | Training time (seconds) | Testing time (seconds) |
|---|---|---|---|
| Gafgyt | Combo | 0.015 | 0.0084 |
| | Junk | 0.011 | 0.0079 |
| | Scan | 0.0132 | 0.0081 |
| | Tcp | 0.0111 | 0.0075 |
| | Udp | 0.0105 | 0.0088 |
| Mirai | Ack | 0.0114 | 0.0087 |
| | Scan | 0.1110 | 0.0083 |
| | Udpplain | 0.0102 | 0.0081 |

Table 10 presents the training and testing times for various attack types in the N-BaIoT dataset, measured in seconds. For Gafgyt attacks, training times range from 0.0132 to 0.015 seconds, with Combo taking the longest at 0.015 seconds and Udp the shortest at 0.0105 seconds. Testing times for Gafgyt range from 0.0075 to 0.0088 seconds, with TCP having the lowest at 0.0075 seconds. In Mirai attacks, training times range from 0.0102 to 0.0114 seconds, while testing times are slightly higher, with Ack at 0.0087 seconds. Overall, both Gafgyt and Mirai attacks demonstrate efficient training and testing times, highlighting the models' effectiveness in detecting these attacks.

### # Experimental analysis of optimization approach

The F1-score was used to determine the hyperparameters in the suggested technique, including learning rates and dropout rates. To determine the best settings for hyperparameters, a variety of hyperparameter tuning strategies can be used. 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.

Table 11: Performance differentiation of proposed with existing optimization approaches

| Optimizer | Acc | Prec | Rec | F1-S |
|---|---|---|---|---|
| SGD | 97.89% | 96.76% | 97.06% | 96.90% |
| RMSprop | 76.30% | 74.57% | 71.95% | 65.27% |
| Adam | 98.01% | 97.61% | 96.95% | 97.25% |
| Nadam | 98.38% | 97.73% | 97.83% | 97.77% |
| AdaMax | 98.06% | 97.26% | 97.21% | 97.23% |
| Proposed | 99.25% | 99.14% | 99.09% | 99.12% |

The efficiency difference between the suggested method and current optimization techniques is shown in Table 11. This illustrates its exceptional capacity to identify and categorize cases efficiently. Nadam demonstrated strong precision and recall values, as well as the next-highest accuracy of all of the current approaches, at 98.38%. With an accuracy rate of 98.01%, the Adam optimizer also demonstrated strong performance.

### # Experimental analysis of overall performances

In this section, we analyse the efficacy of the proposed approach. K-fold validation of the proposed approach is shown in Table 12.

Table 12: Analysis of 10-fold cross-validation over proposed with prior approaches

| Methods | FOLD (%) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th | 9th | 10th |
| Ensemble DNN | 97.39 | 97.19 | 96.99 | 97.19 | 96.79 | 97.6 | 96.19 | 96.19 | 97.39 | 96.39 |
| Hybrid LSTM | 98.67 | 98.65 | 98.59 | 98.51 | 97.89 | 97.76 | 97.55 | 97.43 | 96.97 | 96.77 |
| Proposed | 99.72 | 99.43 | 99.21 | 99.02 | 99 | 98.99 | 98.87 | 98.99 | 98.43 | 98.97 |

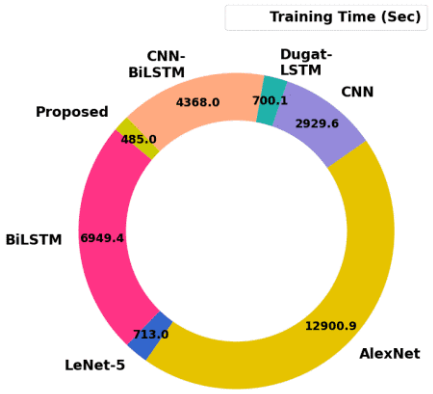


Figure 5: Training time comparison over existing with the proposed approach

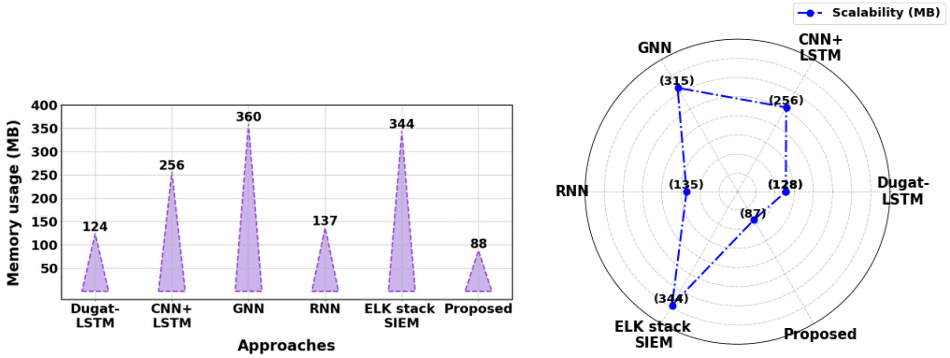


Figure 6: Memory usage and scalabilty analysis of the proposed appro

While contrasting with existing approaches, the proposed approach gains superior efficiency.

Figure 5 shows the training time comparison over existing approaches; while contrasting with existing approaches, the proposed approach takes less time to train. Figure 6 illustrates the proposed approach's Memory usage and scalability analysis; it shows our proposed approach performs superior.

Table 13: Computational cost analysis

| **Approach** | **Training Time (s)** | **Inference Time (ms)** | **Memory Usage (MB)** | **Computational Complexity** |
|---|---|---|---|---|
| CNN-based IDS (ResNet-50) | 320 | 4.1 | 280 | $O(n \cdot d^2)$ |
| LSTM-RNN IDS | 410 | 6.8 | 350 | $O(n \cdot t)$ |
| Autoencoder-based Anomaly Detection | 280 | 3.5 | 210 | $O(n \cdot d)$ |
| Graph Neural Network (GNN)-based Detection | 460 | 7.2 | 400 | O( |
| Lightweight Random Forest IDS | 120 | 2.0 | 150 | $O(n \cdot \log d)$ |
| Proposed Approach | 210 | 2.7 | 190 | $O(n \cdot d \cdot k)$ |

Table 13 shows the computational cost analysis. While comparing with existing approaches proposed obtain less computational complexity.

Table 14: Computational cost analysis

| References | Method used | Advantage | Disadvantage | Accuracy(%) |
|---|---|---|---|---|
| Bhavsar et al. [9] | PCC-CNN | Effective at classifying anomalies and attacks. | May struggle with large-scale datasets. | 95.63% |
| Abusitta et al. [1] | Denoising autoencoder | Robust to unstable environments. | Computationally expensive. | 91.56% |
| Asgharzadeh et al. [4] | BMECapSA-RF | Improves accuracy in IoT anomaly detection. | Increased complexity. | 89.77% |
| Idrissi et al. [12] | Fed-ANIDS | Ensures data privacy. | Requires high computational resources. | |
| Alghamdi and Bellaiche [2] | Federated deep learning with CNN and LSTM | Enhances privacy and security. | Relies on accurate trust attribute definition. | 97.89% |
| Bacha et al. [8] | Kernel PCA and KELM | Reduces data dimensions effectively. | High computation time for large datasets. | 97.71% |
| Xu et al. [6] | SMOTE, mutual information, automated ML | Improves dataset quality. | Risk of overfitting. | 99.12% |
| Vishwakarma and Kesswani [21] | Two-phase IDS with Naive Bayes | Effectively handles different data types. | Time-consuming approach. | 99.42% |
| Proposed approach | | High detection accuracy, reduced training time, efficient on imbalanced data. | May require fine-tuning for deployment in highly dynamic networks. | 99.61% |

Table 14 shows the differentiation of proposed with existing related work.

# 5 Conclusion and future scope

In conclusion, anomaly-based IDS play a critical role in safeguarding IoT networks from diverse and evolving cyber-attacks. Although IoT technology offers numerous advantages, it continues to face persistent challenges related to data security, privacy, and real-time threat detection. Many existing approaches fall short due to limitations in optimizing feature extraction and selection, which adversely impacts the accuracy, speed, and adaptability of detection systems. To address these gaps, this research presents a novel optimized ensemble deep learning-based IDS designed specifically for anomaly detection in IoT environments. The proposed model leverages enhanced CenterNet for effective feature extraction and incorporates the Binary Bamboo Forest Growth Optimization Algorithm (BBFGOA) to identify and select the most relevant features, significantly reducing dimensionality and improving detection accuracy. Furthermore, the Optimized Fuzzy NFSC an ensemble of SCS-Net and NFFS-DFS provides a powerful classification mechanism capable of categorizing network flows and attack types with high precision. The novelty of the proposed method lies in its intelligent integration of feature extraction, selection, and classification techniques that collectively enhance system performance. This comprehensive framework enables precise differentiation between benign and malicious traffic, thereby strengthening the IDS's ability to detect a wide spectrum of anomalies with minimal false positives. The results demonstrate high detection accuracy, efficient resource utilization, and improved responsiveness, making the system suitable for dynamic and resource-constrained IoT environments.

For future work, deploying the model across edge computing nodes is recommended to further reduce latency and distribute computational overhead, thereby enhancing scalability in large-scale IoT ecosystems. Additionally, future studies may explore adaptive learning mechanisms that allow the IDS to evolve in response to newly emerging threats, and lightweight cryptographic integration for securing IDS communication without adding computational strain. This will lead to a more resilient, scalable, and autonomous intrusion detection system capable of maintaining robust IoT network security in real-time.

## Declaration

**Conflict of interests:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

We declare that this manuscript is original, has not been published before and is not currently being considered for publication elsewhere.

**Funding:** No funding was received to assist with the preparation of this manuscript.

**Availability of data and material:** The datasets code are available at

`https://figshare.com/s/942b8eb89ae85c03db10`

**Authors' contributions:** The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

**Ethics approval:** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

[1] A. Abusitta, G. H. S. de Carvalho, O. A. Wahab, T. Halabi, B. C. M. Fung, S. Al Mamoori, *Deep learning-enabled anomaly detection for IoT systems*, Internet of Things, **21** (2023), 100656. https://doi.org/10.1016/j.iot.2022.100656

[2] R. Alghamdi, M. Bellaiche, *A cascaded federated deep learning based framework for detecting wormhole attacks in IoT networks*, Computers and Security, **125** (2023), 103014. https://doi.org/10.1016/j.cose.2022.103014

[3] Z. Alwaisi, T. Kumar, E. Harjula, S. Soderi, *Securing constrained IoT systems: A lightweight machine learning approach for anomaly detection and prevention*, Internet of Things, **28** (2024), 101398. https://doi.org/10.1016/j.iot.2024.101398

[4] H. Asgharzadeh, A. Ghaffari, M. Masdari, F. S. Gharehchopogh, *Anomaly-based intrusion detection system in the Internet of Things using a convolutional neural network and multi-objective enhanced Capuchin Search Algorithm*, Journal of Parallel and Distributed Computing, **175** (2023), 1-21. https://doi.org/10.1016/j.jpdc.2022.12.009

[5] T. Assy, Y. Mostafa, A. Abd El-khaleq, M. Mashaly, *Anomaly-based intrusion detection system using one-dimensional convolutional neural network*, Procedia Computer Science, **220** (2023), 78-85. https://doi.org/10.1016/j.procs.2023.03.013

[6] S. Bacha, A. Aljuhani, K. B. Abdellafou, O. Taouali, N. Liouane, M. Alazab, *Anomaly-based intrusion detection system in IoT using kernel extreme learning machine*, Journal of Ambient Intelligence and Humanized Computing, **15**(1) (2024), 231-242. https://doi.org/10.1007/s12652-022-03887-w

[7] M. P. S. Bhatia, S. R. Sangwan, *Soft computing for anomaly detection and prediction to mitigate IoT-based real-time abuse*, Personal and Ubiquitous Computing, **28**(1) (2024), 123-133. https://doi.org/10.1007/s00779-021-01567-8

[8] M. Bhavsar, K. Roy, J. Kelly, O. Olusola, *Anomaly-based intrusion detection system for IoT application*, Discover Internet of things, **3**(1) (2023), 5. https://doi.org/10.1007/s43926-023-00034-5

[9] I. D. Borlea, R. E. Precup, F. Dragan, A. B. Borlea, *Centroid update approach to K-means clustering*, Advances in Electrical and Computer Engineering, **17**(4) (2017). https://doi.org/10.4316/AECE.2017.04001

[10] P. Deng, Y. Huang, *Edge-featured multi-hop attention graph neural network for intrusion detection system*, Computers and Security, **148** (2025), 104132. https://doi.org/10.1016/j.cose.2024.104132

[11] A. Govindaram, *Flbc-ids: A federated learning and blockchain-based intrusion detection system for secure iot environments*, Multimedia Tools and Applications, **84**(17) (2025), 17229-17251. https://doi.org/10.1007/s11042-024-19777-6

[12] N. Hamdi, *Federated learning-based intrusion detection system for internet of things*, International Journal of Information Security, **22**(6) (2023), 1937-1948. https://doi.org/10.1007/s10207-023-00727-6

[13] M. J. Idrissi, H. Alami, A. El Mahdaouy, A. El Mekki, S. Oualil, Z. Yartaoui, I. Berrada, *Fed-anids: Federated learning for anomaly-based network intrusion detection systems*, Expert Systems with Applications, **234** (2023), 121000. https://doi.org/10.1016/j.eswa.2023.121000

[14] Z. Li, P. Wang, Z. Wang, D. C. Zhan, *Flowgananomaly: Flow-based anomaly network intrusion detection with adversarial learning*, Chinese Journal of Electronics, **33**(1) (2024), 58-71. https://doi.org/10.23919/cje.2022.00.173

[15] M. H. L. Louk, B. A. Tama, *Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system*, Expert Systems with Applications, **213** (2023), 119030. https://doi.org/10.1016/j.eswa.2022.119030

[16] L. Narengbam, S. Dey, *Harris hawk optimization trained artificial neural network for anomaly based intrusion detection system*, Concurrency and Computation: Practice and Experience, **35**(23) (2023), e7771. https://doi.org/10.1002/cpe.7771

[17] T. Saba, A. Rehman, T. Sadad, H. Kolivand, S. A. Bahaj, *Anomaly-based intrusion detection system for IoT networks through deep learning model*, Computers and Electrical Engineering, **99** (2022), 107810. https://doi.org/10.1016/j.compeleceng.2022.107810

[18] L. Sharma, C. Sharma, S. Lal Roy, *Anomaly based network intrusion detection for IoT attacks using deep learning technique*, Computers and Electrical Engineering, **107** (2023), 108626. https://doi.org/10.1016/j.compeleceng.2023.108626

[19] M. Soltani, B. Ousat, M. J. Siavoshani, A. H. Jahangir, *An adaptable deep learning-based intrusion detection system to zero-day attacks*, Journal of Information Security and Applications, **76** (2023), 103516. https://doi.org/10.1016/j.jisa.2023.103516

[20] M. Vishwakarma, N. Kesswani, *A new two-phase intrusion detection system with Naïve Bayes machine learning for data classification and elliptic envelop method for anomaly detection*, Decision Analytics Journal, **7** (2023), 100233. https://doi.org/10.1016/j.dajour.2023.100233

[21] J. Xiao, L. Yang, F. Zhong, H. Chen, X. Li, *Robust anomaly-based intrusion detection system for in-vehicle network by graph neural network framework*, Applied Intelligence, **53**(3) (2023), 3183-3206. https://doi.org/10.1007/s10489-022-03412-8

[22] H. Xu, Z. Sun, Y. Cao, H. Bilal, *A data-driven approach for intrusion and anomaly detection using automated machine learning for the internet of things*, Soft Computing, **27**(19) (2023), 14469-14481. https://doi.org/10.1007/s00500-023-09037-4