ITC 1/53
Information Technology and Control
Vol. 53 / No. 1 / 2024
pp.243-261
DOI 10.5755/j01.itc.53.1.34933

An Efficient Deep Learning-based Intrusion Detection System for Internet of Things Networks with Hybrid Feature Reduction and Data Balancing Techniques

Received 2023/08/25 | Accepted after revision 2024/02/19

HOW TO CITE: Karamollaoğlu, H., Doğru, İ. A., Yücedağ, İ. (2024). An Efficient Deep Learning-based Intrusion Detection System for Internet of Things Networks with Hybrid Feature Reduction and Data Balancing Techniques. *Information Technology and Control*, 53(1), 243-261. https://doi.org/10.5755/j01.itc.53.1.34933

An Efficient Deep Learning-based Intrusion Detection System for Internet of Things Networks with Hybrid Feature Reduction and Data Balancing Techniques

Hamdullah Karamollaoğlu
Department of Computer Engineering, Faculty of Engineering, Düzce University, 81620, Düzce, Turkey

İbrahim Alper Doğru
Department of Computer Engineering, Faculty of Technology, Gazi University, 06570, Ankara, Turkey

İbrahim Yücedağ
Department of Computer Engineering, Faculty of Engineering, Düzce University, 81620, Düzce, Turkey

Corresponding author: hkaramollaoglu@gmail.com

With the increasing use of Internet of Things (IoT) technologies, cyber-attacks on IoT devices are also increasing day by day. Detecting attacks on IoT networks before they cause any damage is crucial for ensuring the security of the devices on these networks. In this study, a novel Intrusion Detection System (IDS) was developed for IoT networks. The IoTID20 and BoT-IoT datasets were utilized during the training phase and performance testing of the proposed IDS. A hybrid method combining the Principal Component Analysis (PCA) and the Bat Optimization (BAT) algorithm was proposed for dimensionality reduction on the datasets. The Synthetic Minority Over-Sampling Technique (SMOTE) was used to address the problem of data imbalance in the classes of the datasets. The Convolutional Neural Networks (CNN) model, a deep learning method, was employed for attack classification. The proposed IDS achieved an accuracy rate of 99.97% for the IoTID20 dataset and 99.98% for the BoT-IoT dataset in attack classification. Furthermore, detailed analyses were conducted to determine the effects of the dimensionality reduction and data balancing models on the classification performance of the proposed IDS.

KEYWORDS: Intrusion detection system, deep learning, IoT networks, feature reduction, data balancing.

1. Introduction

The Internet of Things (IoT) has witnessed substantial growth in recent years, enabling various physical devices and objects to connect to the internet and exchange data. This technology has found applications in diverse fields, including education, healthcare, energy, military, manufacturing, agriculture and transportation [15, 39]. It is projected that the global number of interconnected IoT devices will surpass 30 billion by 2025, as reported by statista.com [22]. However, this rapid proliferation of IoT devices has also led to an increase in cybersecurity threats. Attackers exploit vulnerabilities in IoT networks, aiming to compromise critical or sensitive data through activities such as data interception, modification, or corruption [56]. The resource limitations of IoT devices hinder the deployment of complex security mechanisms that necessitate substantial memory and computational power [23]. Consequently, IoT devices remain susceptible to various cyber-attacks.

Intrusion Detection Systems (IDS) are designed to identify potential attacks that could compromise the security of a network, generating alerts or notifying system administrators about incidents. In IoT networks, one of the primary objectives of IDS is to minimize the false alarm rate in attack detection. IDS can be categorized into two types based on their location: network-based IDS and host-based IDS. Network-based IDS analyze data traffic on the network to detect potential attacks, while host-based IDS examine data traffic on individual host computers [2, 17, 34]. IDS can also be classified based on their detection method: signature-based IDS and behavior-based IDS. Signature-based IDS compare network traffic against a database of known attack signatures, while behavior-based IDS develop a general behavior profile of the network and identify any deviations from this profile as potential attacks [7, 25]. Behavior-based IDS are particularly valuable as they can detect previously unknown attacks [54]. Machine learning-based and deep learning-based models are commonly employed in the design of IDS due to their high accuracy in detecting security threats [3]. However, the limited memory and processing capabilities of IoT devices pose challenges in the development and deployment of IDS [5]. Overcoming these challenges may involve applying dimensionality reduction and data balancing techniques to the datasets used for training and performance testing of IDS. By implementing such techniques, machine learning and deep learning models trained on these datasets can effectively detect intrusion attempts, thereby enhancing the overall performance of the IDS [52].

This study presents a novel approach to intrusion detection in IoT networks, offering several significant contributions. The proposed IDS model demonstrates its effectiveness in accurately identifying security threats with improved accuracy and performance. To address the limitations associated with IDS deployment on resource-constrained IoT devices, the following strategies are employed:

- In order to reduce the dimensionality of the data sets used in the study, a novel hybrid approach was applied by integrating the Bat Optimization (BAT) algorithm with the Principal Component Analysis (PCA) method. This integration aims to effectively reduce the computational complexity associated with the proposed model.
- The SMOTE method was employed to tackle class imbalance in the datasets, which helps prevent bias towards majority classes and ensures that the classification models used for intrusion detection remain effective across different attack scenarios.
- The Grid Search method was utilized to determine optimal hyperparameters for the Convolutional Neural Networks (CNN) model used in the classification stage of the proposed system. This optimization process improves the classification performance of the model, making it more robust in identifying various types of attacks in IoT networks.
- The early stopping technique was employed during the training phase of the CNN model to identify the optimal number of iterations needed for effective model training. This approach helps mitigate overfitting and facilitates efficient training, while taking into account the resource limitations of IoT devices. Additionally, the study incorporated the ReducLROnPlateau callback technique, which dynamically adjusts the learning rate as training advances, in conjunction with early stopping. This combined strategy aims to optimize the training process by contributing to improved convergence and performance improvement.

The study is structured as follows: the second section provides a literature review of existing IDS systems for IoT networks, highlighting the methodologies and datasets used. The third section outlines the materials and methods employed in the study. The fourth section presents the experimental results and findings. Finally, the fifth section evaluates the results, discusses future research directions, and concludes the study.

2. Literature Review

In this section, the literature on IDS for IoT networks is analyzed. The methods used for preprocessing and classification in IDSs and the datasets used in performance testing are examined. Additionally, details on the accuracy rates of IDSs in attack classification are presented.

In recent years, Intrusion Detection Systems (IDS) have garnered significant attention in the realm of securing Internet of Things (IoT) networks. Researchers have made notable contributions by proposing various methodologies and techniques to enhance the accuracy and effectiveness of IDS in detecting and classifying attacks. Within the domain of deep learning-based approaches, Biswas and Roy [8] developed an IDS utilizing the Gated Recurrent Unit (GRU) method, achieving an impressive classification accuracy of 99.76% on the BoT-IoT dataset. Similarly, Popoola et al. [41] employed Long Short-Term Memory (LSTM) for attack classification, attaining a commendable accuracy rate of 97.29% on the BoT-IoT dataset. Ullah et al. [60] explored the utilization of Deep Convolutional Neural Networks (DCNNs), yielding promising results with an accuracy rate of 98.12% on the IoTID20 dataset. Song et al. [55] employed an Autoencoder (AE) to develop an IDS, and its classification performance was measured using the NSL-KDD, IoTID20, and N-BaIoT datasets. The corresponding accuracy rates were 88.7%, 95.2%, and 99.8%, respectively.

Ensemble learning-based techniques have also garnered significant interest within the IDS research domain. Khraisat et al. [27] proposed a stacking technique, combining C5.0 Decision Tree and One-Class Support Vector Machines (One-Class SVMs), which demonstrated exceptional accuracy rates of 99.97% on the BoT-IoT dataset. Lian et al. [31] integrated the Decision Tree (DT) algorithm with Recursive Feature Elimination, resulting in notable enhancements in IDS performance.

Feature reduction techniques have proven to be vital in improving the efficiency of IDS. Alghanam et al. [4] devised an IDS utilizing the Isolation Forest (iForest) method, complemented by the Local Search Algorithm-Pigeon-Inspired Optimization (LS-PIO) hybrid method for feature reduction. Ramana et al. [44] employed the Information Gain (IG) method for feature selection, combined with the Reinforcement Learning-Deep Q-Network (RL-DQN) method for classification. Qaddoura et al. [43] integrated K-means clustering for feature reduction and utilized Support Vector Machine-Synthetic Minority Over-Sampling Technique (SVM-SMOTE) to address imbalanced data distribution.

Community learning approaches have gained prominence in IDS research endeavors. Seth et al. [50] adopted a community learning approach by combining LightGBM and Histogram-Based Gradient Boosting (HBGB) methods, alongside Random Forest (RF) and Principal Component Analysis (PCA) for feature reduction.

Furthermore, researchers have proposed specialized algorithms specifically tailored for IDS. Saba et al. [49] presented an IDS based on the Enhanced Sequential Algorithm (ESA), specifically designed for IoT networks, with evaluations conducted on the Network Intrusion Detection (NID) and BoT-IoT datasets. Yang and Shami [65] employed the Optimized Adaptive and Sliding Window (OASW) method, along with Particle Swarm Optimization (PSO) and LightGBM, for effective attack detection, yielding remarkable accuracy rates on the NSL-KDD and IoTID20 datasets.

The literature indicates that IDSs have been proposed for detecting attacks on IoT networks, and machine learning and deep learning methods are commonly used in the attack classification stage. Various feature reduction and attack classification techniques have been proposed and evaluated on different datasets. However, the implementation of IDSs on IoT devices presents significant challenges due to the limited resources and computing power of these devices. As a result, the development of IDS algorithms that are both accurate and lightweight is a crucial area of research for securing IoT networks. In addition, IDSs may suffer from high false positive rates, resulting in unnecessary alarms and dis-

ruptions, when benign traffic is mistakenly classified as malicious. Another significant challenge is the lack of standardized datasets for evaluating IDS performance, and many studies use different datasets that may not be representative of real-world IoT network traffic. Moreover, new attack vectors and techniques are constantly being developed, and IDSs need to be updated continuously to detect these emerging threats.

3. Material and Method

In this section, comprehensive information regarding the datasets utilized in the study was presented. Additionally, a comprehensive analysis of the methodologies implemented in the architecture of the proposed IDS was provided, elaborating on their operational principles.

3.1. Datasets Used in the Study

The proposed IDS in this study was trained and tested on the IoTID20 and BoT-IoT datasets. The IoTID20 dataset was composed of data collected from various IoT devices, while the BoT-IoT dataset was generated in a controlled network environment. These datasets were chosen due to their distinct attack types and varied sizes, enabling a comprehensive evaluation of the IDS's performance across multiple scenarios. Using two distinct datasets also enhances the generalizability of the IDS's performance across diverse IoT networks.

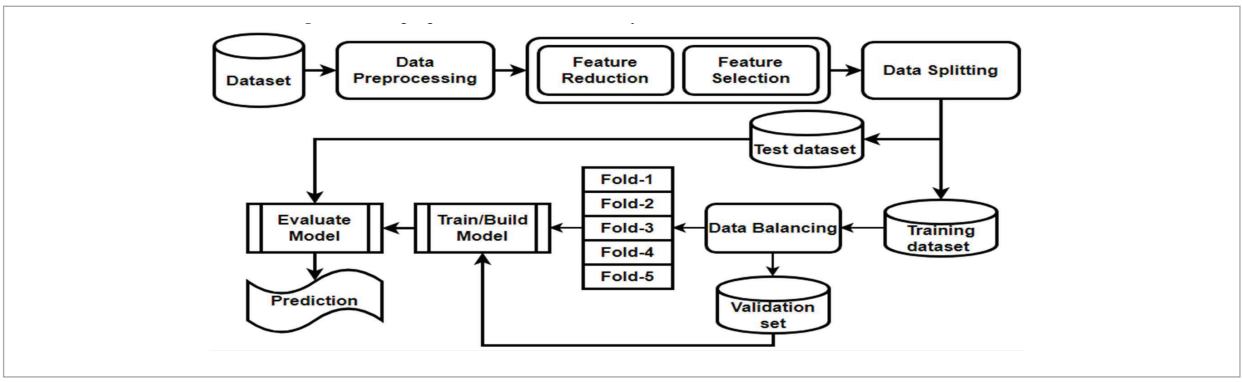
The IoTID20 dataset was generated using data collected from a network consisting of SK Telecom Nugu (Nu-100) smart assistant, EZVIZ wireless security camera, laptops, and smartphones. It contains 625,783 samples, of which 40,073 belong to the normal class, and 585,710 belong to the attack classes. The dataset comprises 84 attributes, except for the class label, which were obtained using the CICFlowMeter tool [29]. The IoTID20 dataset consists of five main classes: Normal (no attack), MITM (man-in-the-middle attack), DoS (denial of service attack), Scan (scanning attack), and Mirai (botnet attack), with 40,073, 35,377, 59,391, 75,265, and 415,677 samples, respectively [21, 59].

The BoT-IoT dataset, on the other hand, was generated by creating a realistic network environment at the UNSW Canberra Cybersecurity Center and collecting 3,668,522 samples from this network. Of these samples, 477 belong to the normal class, and 3,668,045 belong to the attack classes. The dataset contains 42 attributes, excluding the class label. The BoT-IoT dataset consists of five main classes: Normal, DoS, DDoS (distributed denial of service attack), Reconnaissance (reconnaissance attack), and Theft (information theft attack), with 477, 1,650,260, 1,926,624, 91,082, and 79 samples, respectively [9, 28].

3.2. The Proposed Framework

This section presents a detailed description of the IDS developed using the methodologies discussed in the previous section. The study analyzes the results of multi-class attack classification carried out on the IoT-ID20 and BoT-IoT datasets using the proposed model. The workflow of the proposed attack detection system is illustrated in Figure 1. As depicted in Figure 1, the proposed IDS comprises several stages, including data preprocessing, feature reduction/selection, data splitting, data balancing, and attack classification.

Figure 1
The flow diagram for the proposed intrusion detection system



3.2.1. Data Preprocessing

During the data preprocessing stage, the "Flow_ID", "Dst_IP", and "Src_IP" attributes in the IoTID20 dataset, as well as the "pkSeqID", "daddr", and "saddr" attributes in the BoT-IoT dataset were removed since they are identity-based attributes that do not affect the classification performance. Categorical data in the datasets were converted into numerical values using the "LabelEncoder" class of the "sklearn" library [40]. In order to enhance the classification performance of machine learning and deep learning models, it is necessary to scale the data within a certain range before presenting it to the models. The Z-score normalization [35] method was used for data scaling in this study, utilizing the "StandardScaler" class of the "sklearn" library. Furthermore, the Z-score normalization process was performed to normalize the distribution of the data, reducing their mutual influence.

3.2.2. Feature Reduction and Data Splitting

In this study, the PCA-BAT hybrid method was employed for feature reduction. The integration of PCA and BAT methods in this study yields significant advantages for feature reduction. PCA effectively manages noisy data, mitigates overfitting, and enhances classification efficiency by simplifying computational complexity. However, it has limitations in capturing complex relationships in the data. To address this, the BAT algorithm is introduced as a complementary approach. While PCA excels at capturing linear relationships in data, BAT is proficient at capturing non-linear relationships and complex interactions. PCA maintains diversity by reducing correlations among individuals in the feature space, preventing premature convergence of the Bat Algorithm to local optima, and enabling exploration of a wider solution space. Additionally, PCA's dimensionality reduction streamlines the Bat Algorithm's search space, enhancing efficiency and reducing the risk of getting trapped in local optima [14].

PCA is a technique used to transform a high-dimensional sample space, comprising multiple variables, into a lower-dimensional subspace. This subspace is constructed by generating linearly independent artificial variables known as principal components [26]. The primary objective of PCA is to reduce the dimensionality of the data while preserving the essential information contained in the original dataset. By reducing the number of variables, PCA simplifies the analysis process and enhances the visual representation of the data.

In Equation (1), matrix D represents a dataset consisting of n samples and m features. To prevent bias during the application of PCA, it is necessary to scale the samples to a certain range [47]. For this purpose, the study utilized the Standard Scaler method presented in Equation (2), which was obtained using Equations (3)-(4).

$$D = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = [x_1, x_2, \dots, x_n] \quad (1)$$

$$z_i = \frac{x_i - \mu}{\sigma}, i = 1, 2, 3, \dots, n \quad (2)$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \quad (3)$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} z_i \quad (4)$$

Here, x_i represents the i-th sample in the dataset, z_i represents the scaled data in the range of [-1, 1] after normalization, σ represents the standard deviation of the samples in the dataset calculated using Equation (3), and μ represents the mean of the scaled samples obtained using Equation (4). Scaling the data ensures that the dataset has a normal distribution with mean zero (0) and variance one (1). After normalization, the covariance matrix (R) is calculated using Equation (5) to determine the correlation between the samples in the dataset.

$$R = \frac{1}{n}\sum_{i=1}^{n} (z_i - \mu)(z_i - \mu)^T \quad (5)$$

$$Rv_i = \lambda_i v_i, i = 1,2, \dots, m \quad (6)$$

To obtain the principal components of the samples in the dataset, the eigenvectors and eigenvalues of the covariance matrix obtained using Equation (5) must be calculated. This process is carried out using Equation (6), where λ_i represents the i-th eigenvalue of the covariance matrix and v_i represents the corresponding eigenvector. The eigenvalues are sorted from largest to smallest and k eigenvectors corresponding to these eigenvalues are selected to perform dimensionality reduction on the original D matrix. Thus, a new W matrix is obtained from the original D matrix, consisting of k uncorrelated principal components and n samples, with negligible loss of information [18, 63].

The threshold value for the total variance ratio that the principal components should explain was set to 0.99. This allowed for obtaining the principal components that can explain the highest possible total variance in the datasets with the least loss of information. The number of principal components calculated for IoTID20 and BoT-IoT datasets, the variance ratios explained by the principal components, and the total variance explained by the principal components for each dataset are presented in Figures 2-3, respectively. The results shown in Figures 2-3 indicate that even after applying the PCA method and reducing the number of features, the total variance values that explain the datasets remain at a high level. Specifically, the number of features was reduced from 84 to 34 for the IoTID20 dataset and from 42 to 24 for the BoT-IoT dataset. Upon applying PCA, the obtained total explained variance values were 0.99071 for the IoTID20 dataset and 0.99106 for the BoT-IoT dataset. These values indicate that a substantial portion of the original datasets' variability can be effectively captured by the retained principal components.

Figure 2

The variance ratios explained by the principal components for the IoTID20 dataset

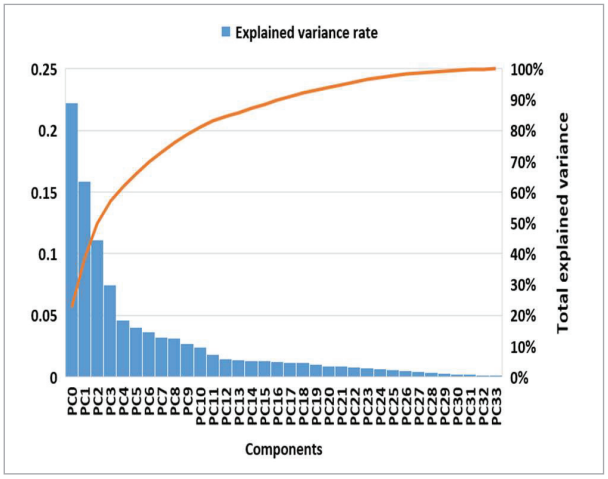
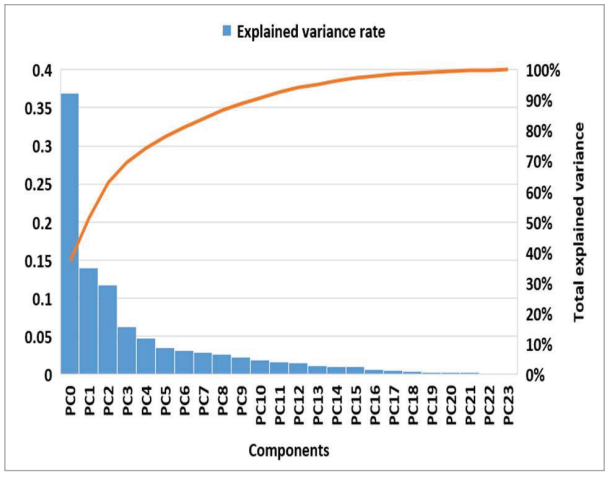


Figure 3

The variance ratios explained by the principal components for the BoT-IoT dataset



In the study, the BAT algorithm was used to select the most important new features among the features obtained by the PCA method. The BAT algorithm is a population-based metaheuristic method inspired by the echolocation behavior of bats in catching their prey. By iteratively adjusting the positions and frequencies of bats, it efficiently explores the search space and seeks optimal solutions. This characteristic renders BAT algorithm particularly well-suited for feature selection tasks [19, 38].

Bats emit high and short sound pulses around them. They use the echoes of these sound pulses that bounce back from objects to calculate the distance between objects, and assume that each bat flies in a random position (x_i) with a fixed frequency (f_i), variable wavelength (λ) and loudness (A) while searching for its prey. Bats can adjust the wavelength and the pulse emission rate (r) of the pulses they emit [1, 38]. The velocity and position of the i-th bat at time step t are calculated using Equations (7)-(9) [36].

$$x_i^t = x_i^{t-1} + v_i^t \tag{7}$$

$$v_i^t = v_i^{t-1} + \left(x_i^{t-1} - x_{gbest}^t\right) f_i \tag{8}$$

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{9}$$

where f_{min} and f_{max} are the minimum and maximum sound frequency, respectively. β is a random vector obtained from a uniform distribution in the range [0,1], and x_{gbest}^t is the global best solution in the t-th iteration. After obtaining the overall solution, a local random walk is calculated using Equation (10) to generate a new solution for each bat.

$$x^t = x_{gbest}^t + \varepsilon A^t, \tag{10}$$

where A^t represents the average loudness of all bats at time step t, and ε is a random number in the range

[-1,1]. The loudness A_i and the pulse emission rate r_i are updated using Equations (11)-(12) [36, 64]:

$$A_i^{t+1} = \alpha A_i^t \quad (11)$$

$$r_i^{t+1} = r_i^0[1 - exp(-\gamma t)], \quad (12)$$

where α is a constant in the range [0,1], and γ is a positive constant.

The proposed PCA-BAT hybrid method in the study follows Algorithm 1, which outlines the step-by-step procedure for integrating PCA and the BAT algorithm to achieve improved feature selection and representation.

Algorithm 1. The hybrid PCA-BAT model for feature reduction

Input: Dataset D consisting of n samples and m features, Number of principal components: k, Maximum number of iterations: max_iter, Termination criterion: max_iter

Define_function PCA (D): Apply the PCA optimization algorithm to the D matrix.

1: Normalize the dataset using the Standard Scaler method (Equations (2)-(4));

2: Calculate the covariance matrix (R) using Equation (5);

3: Obtain the eigenvectors and eigenvalues of the covariance matrix using Equation (6);

4: Sort the eigenvalues in descending order and select k eigenvectors corresponding to these eigenvalues;

5: Construct the projection matrix W consisting of the selected k eigenvectors;

Define_function BAT (W): Apply the BAT optimization algorithm to the W matrix.

6: Fitness function f(x), $x=(x_1,...,x_d)^T$, Initialize the bats' positions (x_i), frequencies (f_i), wavelengths (λ_i), and sound intensities (A_i);

7: Evaluate all the elements in the population by fitness function f(x);

8: Calculate the velocity (v_i) and position (x_i) of each bat using Equations (7)-(9);

9: If the random number (rand (0,1)) is greater than pulse emission rate (r_i), calculate the average loudness (A_t) and update the bats' positions using Equation (10);

10: If the random number (rand (0,1)) is lower than A_i and $f(x_i)<f(x_{gbest})$ Update the loudness (A_i) and pulse emission rate (r_i) using Equations (11) and (12);

11: Repeat Steps 8-10 until convergence (max_iter);

Output: Reduced dataset D'

As seen in Algorithm 1, The PCA method is first used to extract the principal components that capture the maximum possible variance in the dataset with minimal information loss. Then, the BAT optimization algorithm is applied to further reduce the dimensionality of the data by selecting the most relevant features based on the fitness function. This approach allows for efficient and effective feature reduction, which can lead to improved model performance and reduced computational complexity.

In the study, the Grid Search method [30] was used to determine the hyperparameter values used in the BAT algorithm. The "GridSearchCV" function of the "scikit-learn" library was used to perform this method. After the grid search process, the hyperparameter values obtained for the BAT algorithm are shown in Table 1.

Table 1
Hyperparameter values of the BAT method

Hyperparameter	Selected value
Lowest frequency	0
Highest frequency	2
Loudness	1.00
Sound wavelength	0.15
Alpha	0.95
Gamma	0.5
Number of bats used	30
Number of iterations	50

Applying the BAT algorithm for each dataset using the hyperparameter values in Table 1, the highest fitness rates and corresponding feature sets that led to these rates were obtained and presented in Table 2.

Table 2
The features obtained after the BAT algorithm

Dataset	Fitness rate	Features
IoTID20	0.96728720	PC0, PC3, PC8, PC9, PC10, PC20, PC22, PC25, PC28, PC29
BoT-IoT	0.97078535	PC3, PC8, PC11, PC13, PC15, PC16, PC20

As seen in Table 2, the highest fitness rate after 50 iterations was achieved using 10 features for the IoTID20 dataset and 7 features for the BoT-IoT dataset through the PCA-BAT algorithm. As a result, the number of features was reduced from 84 to 10 for the IoTID20 dataset and from 42 to 7 for the BoT-IoT dataset.

In this study, the "train_test_split" function of the "sklearn" library was used to randomly divide the samples in the datasets into 75% training and 25% testing sets. The number of samples in the resulting training and testing sets can be seen in Tables 3-4, respectively.

Table 3

The number of samples in the training and test datasets for the IoTID20 dataset

Class	Training dataset	Test dataset	Total	Rate (%)
DoS	44,284	15,107	59,391	9.491
MITM	26,670	8,707	35,377	5.653
Mirai	311,993	103,684	415,677	66.425
Normal	30,136	9,937	40,073	6.404
Scan	56,254	19,011	75,265	12.027

Table 4

The number of samples in the training and test datasets for the BoT-IoT dataset

Class	Training dataset	Test dataset	Total	Rate (%)
DDoS	1,444,775	481,849	1,926,624	52.518
DoS	1,237,720	412,540	1,650,260	44.984
Normal	358	119	477	0.013
Reconna-issance	68,476	22,606	91,082	2.483
Theft	62	17	79	0.002

Upon examination of Tables 3-4, it is evident that the data within the classes of both the training and testing sets are distributed unevenly. This imbalanced distribution could potentially affect the classification performance of the IDS model used in the study, especially for minority classes.

3.2.3. Handling Imbalanced Data

The imbalanced distribution of samples in the classes of the datasets causes the majority classes to be more represented than the minority classes. This can cause bias in the classification model towards the more represented class, leading to a decrease in the classification performance of the model, particularly for the minority classes. The imbalance ratios of the datasets considered in the study were calculated using Equation (13) which expresses the ratio of the number of samples belonging to the majority class to the number of samples belonging to the minority class.

$$Imbalance\ ratio = \frac{N\_majority}{N\_minority}, \tag{13}$$

where $N\_majority$ represents the number of samples in the majority class and $N\_minority$ represents the number of samples in the minority class [11]. As a result of the calculations using Equation (13), the imbalance ratio was obtained as 11.74 for the IoTID20 dataset and 24387.64 for the BoT-IoT dataset.

Considering the calculation results, it can be observed that there is a significant imbalance between the classes, especially in the BoT-IoT dataset. To address this data imbalance issue, The Synthetic Minority Over-Sampling Technique (SMOTE) method was used in the study. SMOTE is a technique that generates new synthetic samples through linear interpolation between each sample (x_i) in the minority class and its k-nearest minority class neighbors. This method enables the creation of a new balanced dataset with equal numbers of samples from each class by oversampling the minority class in datasets with imbalanced class distributions. This prepares the ground for a classification model that better recognizes the classes in the dataset by preventing the bias that can be exhibited towards the majority class [57, 67]. In SMOTE, the k-nearest minority neighbors of x_i are determined by calculating the Euclidean distance between x_i and each sample in the minority class. Random samples are selected from among the k-nearest minority neighbors of x_i until the number of samples in each class is equal. New synthetic samples are obtained through the calculations performed using Equation (14).

$$x_{sentetik} = x_i + (\bar{x} - x_i) \cdot r. \tag{14}$$

Here, $x_{sentetik}$ represents the new synthetic sample generated, x_i represents the feature vector of the i-th sample in the minority class, $\bar{x}$ represents a randomly selected sample from the k-nearest minority class neighbors of x, and r represents a random number between 0 and 1 [12, 16].

Table 5
The number of samples in the IoTID20 training dataset after applying the SMOTE method

Class	Before SMOTE	After SMOTE
DoS	44,284	311,993
MITM	26,670	311,993
Mirai	311,993	311,993
Normal	30,136	311,993
Scan	56,254	311,993
Total number of samples	469,337	1,559,965

Table 6
The number of samples in the BoT-IoT training dataset after applying the SMOTE method

Class	Before SMOTE	After SMOTE
DDoS	1,444,775	1,444,775
DoS	1,237,720	1,444,775
Normal	358	1,444,775
Reconnaissance	68,476	1,444,775
Theft	62	1,444,775
Total number of samples	2,751,391	7,223,875

In this context, the SMOTE method was applied to the IoTID20 and BoT-IoT training sets to generate synthetic new samples, which helped to balance the number of samples between classes. The number of new samples in the training sets after applying the SMOTE method is presented in Tables 5-6. As indicated in Tables 5-6, the SMOTE technique was employed to oversample the minority classes in the datasets, generating new synthetic samples and equalizing the sample sizes across all classes. In this way, the problem of class imbalance in the datasets was alleviated.

3.2.4. Attack Classification

In the attack classification phase of the study, the CNN method was employed. CNNs are deep learning methods that have found widespread application in various domains. The CNN approach offers several important advantages, including robust fault tolerance, parallel processing capabilities, local connection, weight sharing during convolution operations, and autonomous learning abilities. Furthermore, it excels at accurately extracting features from data while undergoing rapid training. These attributes make CNNs well-suited for addressing the challenges related to network intrusion detection in IoT environments. By utilizing CNNs, intrusion detection systems can effectively identify abnormal behavior and promptly detect emerging threats in real-time. These advanced systems provide enhanced accuracy, scalability, and adaptability, making them a promising solution for strengthening network security and mitigating the risks associated with evolving cyber threats [20, 32, 61].

Typically, a CNN architecture comprises convolutional layers, activation layers, pooling layers, flattening layers, and fully connected layers [6, 33]. In the convolutional layer, a series of numerical filters called kernels are convolved with the input data to extract relevant features, leading to the creation of a feature map. The mathematical formulation for discrete-time and one-dimensional convolution operation is given by Equation (15).

$$y(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) + b. \quad (15)$$

Here, the input data is represented by x, and the kernel by w. The bias value is denoted by b, while time is represented by t. The convolution operation is denoted by the symbol $*$, and the resulting output is represented by $y(t)$ [45, 62].

Activation functions are applied to the output of the convolutional layer in the activation layer. Sigmoid, *tanh* (hyperbolic tangent), and rectified linear unit (ReLU) are commonly used activation functions. However, ReLU is preferred due to its ability to prevent the gradient from vanishing, thus making the network easier to train [24, 37]. ReLU is defined mathematically using Equation (16).

$$ReLU(\beta) = maximum(0, \beta), \quad (16)$$

where β is the input to the activation function, and the output is the maximum of 0 and the input β.

The pooling layer is used to reduce the dimensionality of the feature maps obtained from the convolutional layer. Maximum pooling is typically preferred over average pooling in CNNs [67]. The pooling layer reduces the input feature map's dimensionality by considering a pooling window, performing a maximum operation, and replacing the values with the maximum value. This preserves the most important features of the input data, leading to faster learning and preventing overfitting. The mathematical formulation for one-dimensional maximum pooling is given by Equation (17).

$$Output[i] = max(Input[s * i : s * i + f]), \quad (17)$$

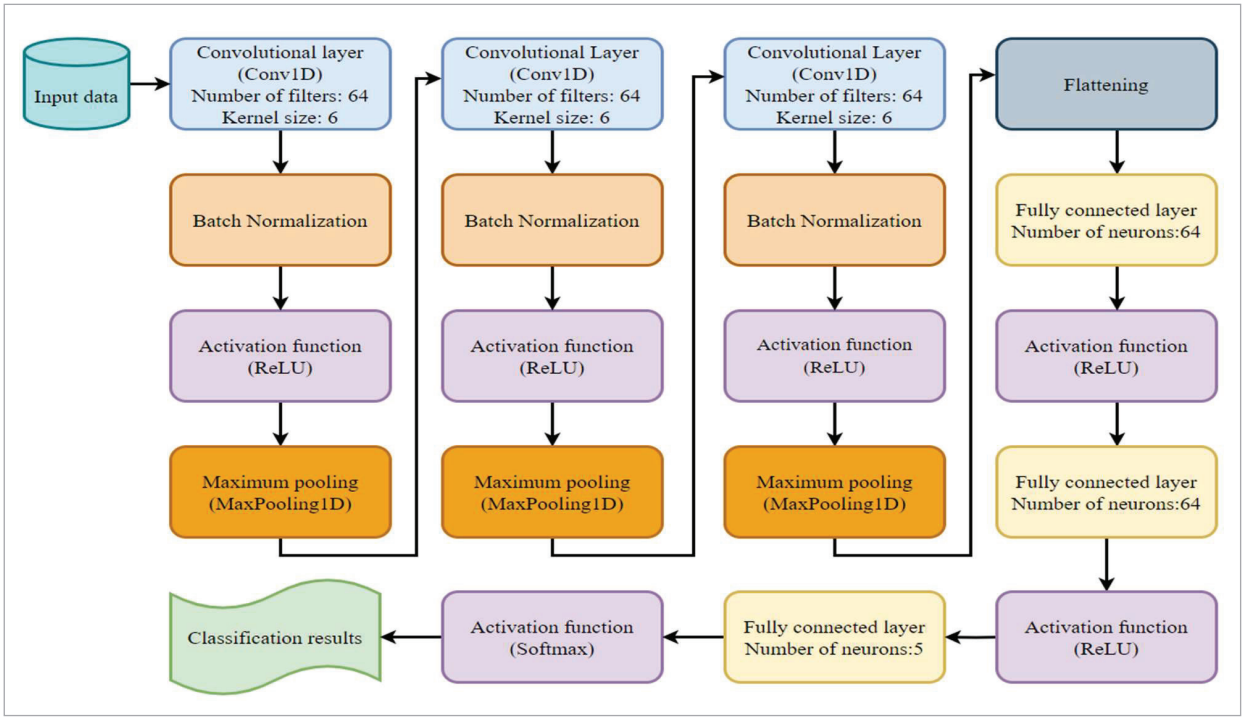
where i is the index of the output feature map, f is the pooling window size, and s is the pooling stride [58, 66].

In the flattening layer, multidimensional data from the previous layers are converted into one-dimensional data by rearranging the data into a sequence of one-dimensional elements. This step prepares the data for the fully connected layer, which operates on one-dimensional data [33].

The fully connected layer, also referred to as the dense layer, plays a fundamental role in numerous deep learning architectures. It establishes connections between all the neurons in the current layer and those in the preceding layer, enabling the extraction of intricate features from high-dimensional data. Within this layer, the input data undergoes a multiplication operation with weights acquired by the model, followed by the addition of a bias value, prior to being fed through an activation function. The outcome of this layer is employed to obtain the ultimate classification result [24, 37].

The general structure and process steps of the CNN method used in the study are presented in Figure 4. In the proposed CNN model, as seen in Figure 4, there are three convolutional layers. After each convolutional layer, data normalization is performed using batch normalization to improve the network's performance and processing speed during training, as well as to prevent the problems of gradient explosion and vanishing gradient. Following the batch normalization process, an activation layer and a pooling layer are used in the proposed model. ReLU is used as the

Figure 4
The general structure of the proposed CNN model



activation function, and maximum pooling is used for the pooling process. After the data is flattened by the flatten layer, it is made suitable for the fully connected layer. Three fully connected layers are used, and after each fully connected layer, an activation layer is used. ReLU is used as the activation function in the first two activation layers, while Softmax activation function is used in the last activation layer with the number of neurons equal to the number of classes in the dataset. The hyperparameter configuration of the CNN method is determined using Grid Search method and the experimental studies in the literature. The hyperparameters used in the layers of CNN in the study and their corresponding values are presented in Table 7.

In addition to the hyperparameters detailed in Table 7, the "early stop" function of the "Keras" library [13, 42] was used in this study. The training process was initialized with 500 epochs and the "early stop" mechanism was implemented using a predefined threshold of 20. This approach aims to minimize the risk of overfitting by terminating the training process when the model reaches the epoch value of a certain performance threshold. Within the training dataset, a special subset containing 20% of the samples is reserved as validation data. At any point in the training phase, the training process was terminated if no reduction in "validation loss" was observed during the previous 20 epochs.

In order to comprehensively evaluate the recommended classification model, k-fold cross-validation method [46] was employed on the training dataset. Taking into account the resource constraints often encountered in IoT networks the default k=5 value provided by the scikit-learn library was preferred in this study instead of a higher k value to avoid unnecessary computational complexity and cost [51, 53]. The training data was divided into 5 distinct folds, each of which served for an individual experiment.

Table 7
Hyperparameters used in the CNN model

Hyperparameter	Selected value
Kernel size (Convolution layer)	6
Activation function (Convolution layer)	ReLU
Number of filters (Convolution layer)	64
Stride (Convolution layer)	1
Momentum	0.9
Epsilon	0.0001
Number of neurons (Fully connected layer)	64 (first two layers), 5 (last layer)
Activation function (Fully connected layer)	ReLU (first two layers), Softmax (last layer)
Optimization method	Adam (learning rate: 0.001)
Loss function	Categorical cross entropy
Batch size	64
Number of epochs	500

4. Performance Evaluation and Experimental Results

The proposed IDS in this study was developed using a personal computer equipped with an AMD Ryzen Pro 3.70 GHz processor, 32 GB RAM, and a 64-bit Windows operating system, using the Python programming language through the Anaconda platform [48]. The performance of the proposed model was evaluated using several metrics, including accuracy, precision, recall, and F1-score, which were calculated using Equations (18)-(21), respectively.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+F} \quad (18)$$

$$Precision = \frac{TP}{TP+FP} \quad (19)$$

$$Recall = \frac{TP}{TP+F} \quad (20)$$

$$F1-score = 2 * \frac{Precision*Recall}{Precision+Recall}. \quad (21)$$

Here, TP (True Positive) represents the number of samples that are correctly classified as positive by the model and are actually positive, FP (False Positive) represents the number of samples that are incorrectly classified as positive by the model but are actually

negative, FN (False Negative) represents the number of samples that are incorrectly classified as negative by the model but are actually positive, and TN (True Negative) represents the number of samples that are correctly classified as negative by the model and are actually negative. Figures 5-6 display confusion matrices, while Tables 8-9 present performance metrics of the proposed IDS for the IoTID20 and BoT-IoT datasets, respectively.

In addition, the study analyzed the effects of feature reduction and data balancing methods used in the proposed IDS on classification performance by applying two different scenarios on the IoTID20 and BoT-IoT datasets. For Scenario 1, the attack classification was tested by applying only the feature reduction process without data balancing on the relevant datasets. For Scenario 2, the attack classification was tested without applying data balancing and feature reduction processes on the datasets.

Figure 5

Confusion matrix of the proposed IDS for the IoTID20 dataset

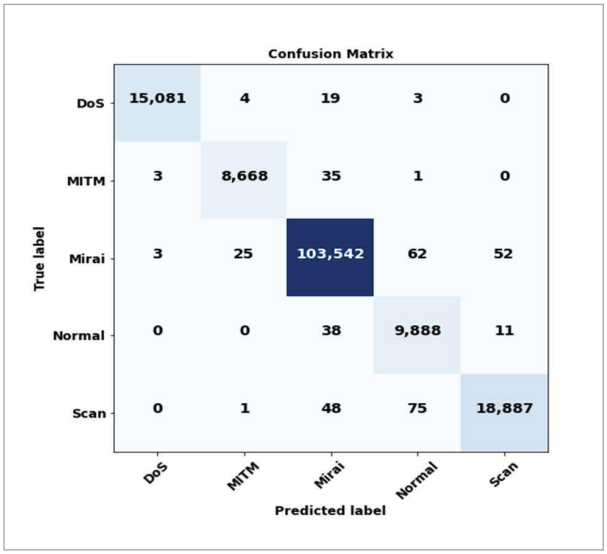


Table 8

Classification performance metric values of the proposed model for the IoTID20 dataset

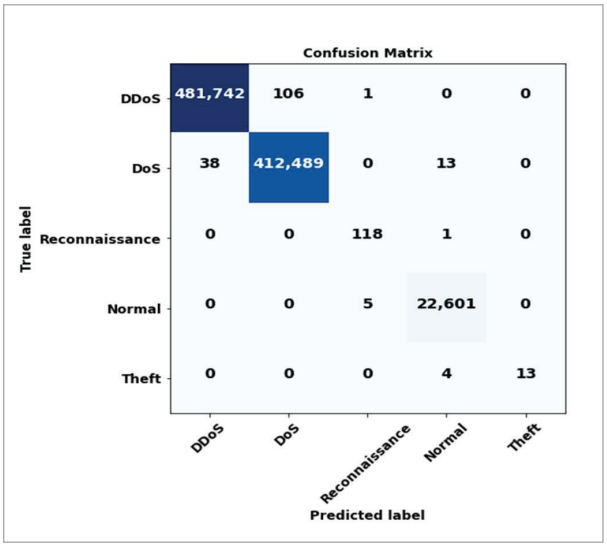
Category	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Number of samples	Training time (s)
Model	99.967	99.790	99.696	99.743	156,354	13,818
DoS	99.980	99.960	99.828	99.894	14,848	
MITM	99.956	99.655	99.552	99.604	8,844	
Mirai	99.820	99.865	99.863	99.864	103,828	
Normal	99.879	98.594	99.507	99.048	10,018	
Scan	99.880	99.668	99.348	99.507	18,816	

Table 9

Classification performance metric values of the proposed model for the BoT-IoT dataset

Category	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Number of samples	Training time (s)
Model	99.984	99.982	99.982	99.982	917,131	20,466
DDoS	99.984	99.992	99.978	99.985	481,849	
DoS	99.983	99.974	99.988	99.981	412,540	
Reconnaissance	99.999	95.161	99.160	97.119	119	
Normal	99.997	99.920	99.978	99.949	22,606	
Theft	99.999	99.999	76.471	86.667	17	

Figure 6
Confusion matrix of the proposed IDS for the BoT-IoT dataset



The results obtained by applying Scenario 1 and Scenario 2 on the IoTID20 dataset are presented in Tables 10-11, respectively. Similarly, the results obtained by applying Scenario 1 and Scenario 2 on the BoT-IoT dataset are presented in Tables 12-13, respectively. When comparing the performance results in Tables 10-13 for Scenarios 1-2 with the performance results in Tables 8-9 for the proposed IDS in the study, it can be seen that performing classification without data balancing significantly reduces precision, recall, and F1-score values, especially for minority classes such as the Theft. Additionally, it is observed that feature reduction significantly reduces training time, while data balancing increases training time due to oversampling.

In this study, a comprehensive comparison was made with different state-of-the-art machine learning methods including DT, AdaBoost, Logistic Regression (LR), Gaussian Naive Bayes (Gaussian NB),

Table 10
Performance metrics obtained as a result of the implementation of Scenario 1 for the IoTID20 dataset

Category	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Number of samples	Training time (s)
Model	99.720	99.569	99.570	99.569	156,354	7,154
DoS	99.973	99.914	99.808	99.861	14,848	
MITM	99.889	99.023	98.989	99.006	8,844	
Mirai	99.643	99.677	99.785	99.731	103,828	
Normal	99.822	98.768	98.420	98.594	10,018	
Scan	99.813	99.383	99.079	99.231	18,816	

Table 11
Performance metrics obtained as a result of the implementation of Scenario 2 for the IoTID20 dataset

Category	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Number of samples	Training time (s)
Model	98.146	97.480	97.254	97.290	156,354	16,378
DoS	99.962	99.993	99.616	99.804	14,848	
MITM	99.240	99.828	86.494	92.684	8,844	
Mirai	97.563	98.894	97.413	98.148	103,828	
Normal	98.601	83.118	97.856	89.887	10,018	
Scan	99.172	94.241	99.248	96.680	18,816	

Table 12
Performance metrics obtained as a result of the implementation of Scenario 1 for the BoT-IoT dataset

Category	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Number of samples	Training time (s)
Model	99.994	83.227	75.204	78.396	917,131	5,386
DDoS	99.997	99.999	99.996	99.997	481,849	
DoS	99.982	99.961	99.999	99.980	412,540	
Reconnaissance	99.994	76.786	72.269	74.459	119	
Normal	99.989	99.713	99.845	99.779	22,606	
Theft	99.999	88.889	47.059	61.538	17	

Table 13
Performance metrics obtained as a result of the implementation of Scenario 2 for the BoT-IoT dataset

Category	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Number of samples	Training time (s)
Model	99.991	92.405	67.306	76.774	917,131	27,334
DDoS	99.971	99.965	99.980	99.973	481,849	
DoS	99.966	99.973	99.952	99.962	412,540	
Reconnaissance	99.994	88.571	61.386	72.515	119	
Normal	99.989	99.989	99.567	99.777	22,606	
Theft	99.999	99.999	41.667	58.824	17	

Figure 7
The performance comparison for IoTID20 dataset

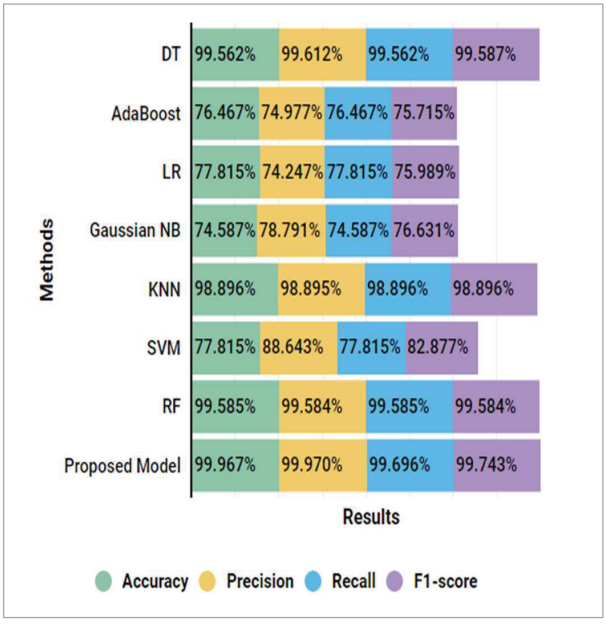


Figure 8
The performance comparison for BoT-IoT dataset

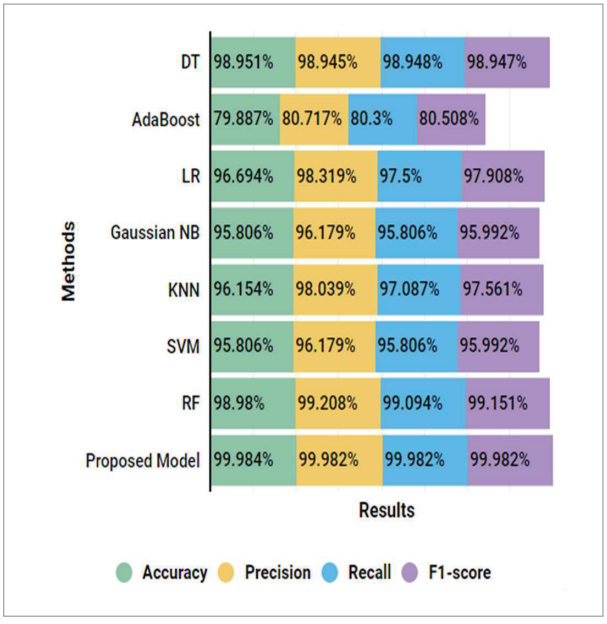


Table 14
Accuracy comparison for the IoTID20 dataset

Study	The method(s) used	Accuracy (%)
Qaddoura et al. [43]	SLFN	93.51
Song at al. [55]	AE	95.20
Ullah et al. [60]	DCNN	98.12
Ramana et al. [44]	IG, RL-DQN	99.40
Yang and Shami [65]	OASW, PSO, LightGBM	99.92
Proposed model	PCA, BAT, SMOTE, CNN	99.97

Table 15
Accuracy comparison for the BoT-IoT dataset

Study	The method(s) used	Accuracy (%)
Saba et al. [49]	CNN	95.55
Popoola et al. [41]	LAE, LSTM	97.29
Alghanam et al. [4]	LS-PIO, iForest	97.37
Biswas and Roy [8]	GRU	99.76
Khraisat et al. [27]	IG, C5.0 DT, One-Class SVM	99.97
Proposed method	PCA, BAT, SMOTE, CNN	99.98

K-Nearest Neighbors (KNN), SVM, and RF [10] to evaluate the performance of the CNN method used for the attack classification of the proposed IDS. This evaluation was carried out on IoTID20 and BoT-IoT datasets, which were subjected to feature reduction with PCA-BAT hybrid technique and data stabilization with SMOTE method. The results obtained from this evaluation are presented in Figures 7-8. The comparative analysis results shown in Figures 7-8 reveal that the CNN method used in the proposed IDS outperforms other state-of-the-art techniques. Moreover, Tables 14-15 provide a comprehensive evaluation of the accuracy performance of the proposed IDS in comparison to the IDSs utilized in previous studies on the IoTID20 and BoT-IoT datasets for attack classification. Upon analyzing Tables 14-15, it is seen that the IDS model proposed in this study outperforms previous IDS studies on IoTID20 and BoT-IoT datasets in terms of classification accuracy.

5. Conclusions

This study presents a novel IDS model for IoT network which was developed and evaluated using the BoT-IoT and IoTID20 datasets, which consist of various attack types, including MITM, Mirai, Scan, DoS, DDoS, Reconnaissance, and Theft. The findings suggest that implementing the proposed CNN method in combination with PCA-BAT feature reduction and SMOTE data balancing techniques can significantly enhance the detection performance of IoT network attacks. Specifically, for the IoT-IoT dataset, the proposed IDS achieved an accuracy of 99.967%, a precision of 99.790%, a recall of 99.696%, and an F1-score of 99.743%. Likewise, for the BoT-IoT dataset, it achieved an accuracy of 99.984%, a precision of 99.982%, a recall of 99.982%, and an F1-score of 99.982. These results demonstrate the effectiveness of the proposed model for detecting intrusion in IoT networks, while addressing the challenge of limited hardware resources.

In addition, the study evaluated the performance of the proposed model by comparing it with state-of-the-art machine learning techniques, including DT, AdaBoost, LR, Gaussian NB, KNN, SVM, and RF, as well as previous studies focusing on IDS using the BoT-IoT and IoTID20 datasets. The results demonstrated that the proposed model outperformed these machine learning methods and previous studies in terms of critical evaluation metrics, including accuracy, precision, recall, and F1-score.

References

1. Agarwal, T., Kumar, V. A Systematic Review on Bat Algorithm: Theoretical Foundation, Variants, and Applications. Archives of Computational Methods in Engineering, 2022, 29, 2707-2736. https://doi.org/10.1007/s11831-021-09673-9
2. Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Alazab, M., Bhattacharya, S., Maddikunta, P. K. R., Gadekallu, T. R. Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions. Computer Communications,

2022, 195, 346-361. https://doi.org/10.1016/j.comcom.2022.09.012

3. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., Ahmad, F. Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches. Transactions on Emerging Telecommunications Technologies, 2021, 32(1), e4150. https://doi.org/10.1002/ett.4150
4. Alghanam, O. A., Almobaideen, W., Saadeh, M., Adwan, O. An Improved PIO Feature Selection Algorithm for IoT Network Intrusion Detection System Based on Ensemble Learning. Expert Systems with Applications, 2023, 213, 118745. https://doi.org/10.1016/j.eswa.2022.118745
5. Alkadi, S., Al-Ahmadi, S., Ben Ismail, M. M. Toward Improved Machine Learning-Based Intrusion Detection for Internet of Things Traffic. Computers, 2023, 12(8), 148. https://doi.org/10.3390/computers12080148
6. Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Al-Amidie, M., Farhan, L. Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions. Journal of Big Data, 2021, 8, 1-74. https://doi.org/10.1186/s40537-021-00444-8
7. Bhavsar, M., Roy, K., Kelly, J., Olusola, O. Anomaly-based Intrusion Detection System for IoT Application. Discover Internet of Things, 2023, 3(1), 5. https://doi.org/10.1007/s43926-023-00034-5
8. Biswas, R., Roy, S. Botnet Traffic Identification Using Neural Networks. Multimedia Tools and Applications, 2021, 80(16), 24147-24171. https://doi.org/10.1007/s11042 -021-10765-8
9. BoT-IoT Dataset. https://ieee-dataport.org/documents/ bot-iot-dataset. Accessed on October 10, 2023.
10. Buczak, A. L., Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications surveys and tutorials, 2015, 18(2), 1153-1176. https://doi.org/10.1109/COMST.2015.2494502
11. Buda, M., Maki, A., Mazurowski, M. A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks. Neural Networks, 2018, 106, 249-259. https://doi.org/10.1016/j.neunet.2018.07.011
12. Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer W. P. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 2022, 16, 321-357. https://doi.org/10.1613/jair.953
13. Chollet, F. Deep Learning with Python. Simon and Schuster, New York, 2021.
14. Cui, Z., Li, F., Zhang, W. Bat Algorithm with Principal Component Analysis. International Journal of Machine Learning and Cybernetics, 2019, 10, 603-622. https://doi.org/10.1007/s13042-018-0888-4
15. Dina, A. S., Siddique, A. B., Manivannan, D. A Deep Learning Approach for Intrusion Detection in Internet of Things Using Focal Loss Function. Internet of Things, 2023, 22, 100699. https://doi.org/10.1016/j.iot.2023.100699
16. Elreedy, D., Atiya, A. F. A Comprehensive Analysis of Synthetic Minority Oversampling Technique (SMOTE) for Handling Class Imbalance. Information Sciences, 2019, 505, 32-64. https://doi.org/10.1016/j.ins.2019.07.070
17. Gamage, S., Samarabandu, J. Deep Learning Methods in Network Intrusion Detection: A Survey and an Objective Comparison. Journal of Network and Computer Applications, 2020, 169, 102767. https://doi.org/10.1016/j.jnca.2020.102767
18. Gao, J., Chai, S., Zhang, B., Xia, Y. Research on Network Intrusion Detection Based on Incremental Extreme Learning Machine and Adaptive Principal Component Analysis. Energies, 2019, 12(7), 1223. https://doi.org/10.3390/en12071223
19. Ghanem, W. A. H., Ghaleb, S. A. A., Jantan, A., Nasser, A. B., Saleh, S. A. M., Ngah, A., Abiodun, O. I. Cyber Intrusion Detection System Based on a Multiobjective Binary Bat Algorithm for Feature Selection and Enhanced Bat Algorithm for Parameter Optimization in Neural Networks. IEEE Access, 2022, 10, 76318-76339. https://doi.org/10.1109/ACCESS.2022.3192472
20. Habib, G., Qureshi, S. Optimization and Acceleration of Convolutional Neural Networks: A Survey. Journal of King Saud University-Computer and Information Sciences, 2022, 34(7), 4244-4268. https://doi.org/10.1016/j.jksuci.2020.10.004
21. Network Intrusion Dataset. https://ieee-dataport.org/open-access/iot-network-intrusion-dataset. Accessed on October 10, 2023.
22. IoT Number of Connected Devices Worldwide. https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide. Accessed on October 10, 2023.
23. Jullian, O., Otero, B., Rodrigue, E., Gutierrez, N., Antona, H., Canal, R. Deep-Learning Based Detection for Cy-

ber-Attacks in IoT Networks: A Distributed Attack Detection Framework. Journal of Network and Systems Management, 2023, 31(2), 33. https://doi.org/10.1007/s10922-023-09722-7

24. Kabakus, A. T. DroidMalwareDetector: A Novel Android Malware Detection Framework Based on Convolutional Neural Network. Expert Systems with Applications, 2022, 206, 117833. https://doi.org/10.1016/j.eswa.2022.117833
25. Khanna, A., Rani, P., Garg, P., Singh, P. K., Khamparia A. An Enhanced Crow Search Inspired Feature Selection Technique for Intrusion Detection Based Wireless Network System. Wireless Personal Communications, 2022, 127(3), 2021-2038. https://doi.org/10.1007/s11277-021-08766-9
26. Kherif, F., Latypova, A. Principal Component Analysis. In: Machine Learning, Academic Press, 2020, 209-225. https://doi.org/10.1016/B978-0-12-815739-8.00012-2
27. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., Alazab A. A Novel Ensemble of Hybrid Intrusion Detection System for Detecting Internet of Things Attacks. Electronics, 2019, 8(11), 1210. https://doi.org/10.3390/electronics8111210
28. Koroniotis, N., Nour, M., Elena S., Benjamin, T. Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset. Future Generation Computer Systems, 2019, 100, 779-796. https://doi.org/10.1016/j.future.2019.05.041
29. Lashkari, A. H., Draper-Gil, G., Mamun, M. S. I., Ghorbani, A. A. Characterization of Tor Traffic Using Time Based Features. International Conference on Information Systems Security and Privacy, 2017, 253-262. https://doi.org/10. 5220/0006105602530262
30. Lerman, P. M. Fitting Segmented Regression Models by Grid Search. Journal of the Royal Statistical Society Series C: Applied Statistics, 1980, 29(1), 77-84. https://doi.org/10.2307/2346413
31. Lian, W., Nie, G., Jia, B., Shi, D., Fan, Q., Liang, Y. An Intrusion Detection Method Based on Decision Tree-Recursive Feature Elimination in Ensemble Learning. Mathematical Problems in Engineering, 2020, 1-15. https://doi.org/10.1155/2020/2835023
32. Liu, J., Shi, Q., Han, R., Yang, J. A Hybrid GA-PSO-CNN Model for Ultra-Short-Term Wind Power Forecasting. Energies, 2021, 14(20), 6500. https://doi.org/10.3390/en14206500
33. Mahadik, S., Pawar, P. M., Muthalagu, R. Efficient Intelligent Intrusion Detection System for Heterogeneous Internet of Things (HetIoT). Journal of Network and Systems Management, 2023, 31(1), 2. https://doi.org/10.1007/s10922-022-09697-x
34. Martins, I., Resende, J. S., Sousa, P. R., Silva, S., Antunes, L., Gama, J. Host-Based IDS: A Review and Open Issues of an Anomaly Detection System in IoT. Future Generation Computer Systems, 2022, 133, 95-113. https://doi.org/10.1016/j.future.2022.03.001
35. Milligan, G. W., Cooper, M. C. A Study of Standardization of Variables in Cluster Analysis. Journal of Classification, 1988, 5(2), 181-204. https://doi.org/10.1007/BF01897163
36. Mirjalili, S., Mirjalili, S. M., Yang, X. S. Binary Bat Algorithm. Neural Computing and Applications, 2014, 25(3), 663-681. https://doi.org/10.1007/s00521-013-1525-5
37. Nair, V., Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. 27th International Conference on Machine Learning, 2010, 807-814.
38. Natesan, P., Rajalaxmi, R. R., Gowrison, G., Balasubramanie, P. Hadoop Based Parallel Binary Bat Algorithm for Network Intrusion Detection. International Journal of Parallel Programming, 2017, 45(5), 1194-1213. https://doi.org/10.1007/s10766-016-0456-z
39. Nguyen, T. N., Ngo, Q. D., Nguyen, H. T., Nguyen, G. L. An Advanced Computing Approach for IoT-Botnet Detection in Industrial Internet of Things. IEEE Transactions on Industrial Informatics, 2022, 18(11), 8298-8306. https://doi.org/10.1109/TII.2022.3152814
40. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Duchesnay, E. Scikit-Learn: Machine Learning in Python. The Journal of Machine Learning Research, 2011, 12, 2825-2830. https://doi.org/10.48550/arXiv.1201.0490
41. Popoola, S. I., Adebisi, B., Hammoudeh, M., Gui, G., Gacanin, H. Hybrid Deep Learning for Botnet Attack Detection in the Internet-of-Things Networks. IEEE Internet of Things Journal, 2020, 8(6), 4944-4956. https://doi.org/10.1109/JIOT.2020.3034156
42. Prechelt, L. Automatic Early Stopping Using Cross Validation: Quantifying the Criteria. Neural Networks, 1988, 11(4), 761-767. https://doi.org/10.1016/S0893-6080(98)00010-0
43. Qaddoura, R., Al-Zoubi, A. M., Almomani, I., Faris, H. A Multi-Stage Classification Approach for IoT Intru-

sion Detection Based on Clustering with Oversampling. Applied Sciences, 2021, 11(7), 3022. https://doi.org/10.3390/app11073022

44. Ramana, T. V., Thirunavukkarasan, M., Mohammed, A. S., Devarajan, G. G., Nagarajan, S. M. Ambient Intelligence Approach: Internet of Things Based Decision Performance Analysis for Intrusion Detection. Computer Communications, 2022, 195, 315-322. https://doi.org/10.1016/j.comcom.2022.09.007
45. Riyaz, B., Ganapathy, S. A Deep Learning Approach for Effective Intrusion Detection in Wireless Networks Using CNN. Soft Computing, 2020, 24, 17265-17278. https://doi.org/10.1007/s00500-020-05017-0
46. Rodriguez, J. D., Perez, A., Lozano, J. A. Sensitivity Analysis of K-Fold Cross Validation in Prediction Error Estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009, 32(3), 569-575. https://doi.org/10.1109/TPAMI.2009.187
47. Roy, S., Li, J., Choi, B. J., Bai, Y. A Lightweight Supervised Intrusion Detection Mechanism for IoT Networks. Future Generation Computer Systems, 2022, 127, 276-285. https://doi.org/10.1016/j.future.2021.09.027
48. Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., Church, K. Introduction to Anaconda and Python: Installation and Setup. Python Res. Psychol, 2016, 16(5), 3-11. https://doi.org/10.20982/tqmp.16.5.S003
49. Saba, T., Rehman, A., Sadad, T., Kolivand, H., Bahaj, S. A. Anomaly-Based Intrusion Detection System for IoT Networks Through Deep Learning Model. Computers and Electrical Engineering, 2022, 99, 107810. https://doi.org/10.1016/j.compeleceng.2022.107810
50. Seth, S., Chahal, K. K., Singh, G. A Novel Ensemble Framework for an Intelligent Intrusion Detection System. IEEE Access, 2021, 9, 138451-138467. https://doi.org/10.1109/ACCESS.2021.3116219
51. Sevinç, E. An Empowered AdaBoost Algorithm Implementation: A COVID-19 Dataset Study. Computers and Industrial Engineering, 2022, 165, 107912. https://doi.org/10.1016/j.cie.2021.107912
52. Sharma, B., Sharma, L., Lal, C., Roy, S. Anomaly Based Network Intrusion Detection for IoT Attacks Using Deep Learning Technique. Computers and Electrical Engineering, 2023, 107, 108626. https://doi.org/10.1016/j.compeleceng.2023.108626
53. Sklearn K-Folds Cross-Validator. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html. Accessed on October 10, 2023.
54. Sommestad, T., Holm, H., Steinvall, D. Variables Influencing the Effectiveness of Signature-Based Network Intrusion Detection Systems. Information Security Journal: A Global Perspective, 2022, 31(6), 711-728. https://doi.org/10.1080/19393555.2021.1975853
55. Song, Y., Hyun, S., Cheong, Y. G. Analysis of Autoencoders for Network Intrusion Detection. Sensors, 2021, 21(13), 4294. https://doi.org/10.3390/s21134294
56. Talukder, M. A., Hasan, K. F., Islam, M. M., Uddin, M. A., Akhter, A., Yousuf, M. A., Alharbi, F., Moni, M. A. A Dependable Hybrid Machine Learning Model for Network Intrusion Detection. Journal of Information Security and Applications, 2023, 72, 103405. https://doi.org/10.1016/j.jisa.2022.103405
57. Tan, X., Su, S., Huang, Z., Guo, X., Zuo, Z., Sun, X., Li, L. Wireless Sensor Networks Intrusion Detection Based on SMOTE and the Random Forest Algorithm. Sensors, 2019, 19(1), 203. https://doi.org/10.3390/s19010203
58. Tolias, G., Sicre, R., Jégou, H. Particular Object Retrieval with Integral Max-Pooling of CNN Activations. arXiv Preprint arXiv:1511.05879, 2015. https://doi.org/10.48550/arXiv.1511.05879
59. Ullah, I., Mahmoud, Q. H. A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks. Canadian Conference on Artificial Intelligence, 2020, 508-520. https://doi.org/10.1007/978-3-030-47358-7_52
60. Ullah, S., Ahmad, J., Khan, M. A., Alkhammash, E. H., Hadjouni M., Ghadi, Y. Y., Saeed, F., Pitropakis, N. A New Intrusion Detection System for the Internet of Things via Deep Convolutional Neural Network and Feature Engineering. Sensors, 2022, 22(10), 3607. https://doi.org/10.3390/s22103607
61. Wu, Y., Nie, L., Wang, S., Ning, Z., Li, S. Intelligent Intrusion Detection for Internet of Things Security: A Deep Convolutional Generative Adversarial Network-Enabled Approach. IEEE Internet of Things Journal, 2023, 10, 3094-3106. https://doi.org/10.1109/JIOT.2021.3112159
62. Wu, J. M. T., Li, Z., Herencsar, N., Vo, B., Lin, J. C. W. A Graph-Based CNN-LSTM Stock Price Prediction Algorithm with Leading Indicators. Multimedia Systems, 2021, 1-20. https://doi.org/10.1007/s00530-021-00758-w
63. Xiao, Y., Xing, C., Zhang, T., Zhao, Z. An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks. IEEE Access, 2019, 7, 42210-42219. https://doi.org/10.1109/ACCESS.2019.2904620

64. Yang, B., Lu, Y., Zhu, K., Yang, G., Liu, J., Yin, H. Feature Selection Based on Modified Bat Algorithm. IEICE Transactions on Information and Systems, 2017, 100(8), 1860-1869. https://doi.org/10.1587/transinf.2016EDP7471

65. Yang, L., Shami, A. A Lightweight Concept Drift Detection and Adaptation Framework for IoT Data Streams. IEEE Internet of Things Magazine, 2021, 4(2), 96-101. https://doi.org/10.1109/IOTM.0001.2100012

66. Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., ..., Almotairi, S. A Comparison of Pooling Methods for Convolutional Neural Networks. Applied Sciences, 2022, 12(17), 8643. https://doi.org/10.3390/app12178643

67. Zhang, H., Huang, L., Wu, C. Q., Li, Z. An Effective Convolutional Neural Network Based on SMOTE and Gaussian Mixture Model for Intrusion Detection in Imbalanced Dataset. Computer Networks, 2020, 177, 107315. https://doi.org/10.1016/j.comnet.2020.107315