InfoTech Spectrum: Iraqi Journal of Data Science (IJDS)
ISSN: 3007-5467, Vol 1(No.1), 27-41 (2024)
DOI: https://doi.org/10.51173/ijds.v1i1.9
https://ijds.mtu.edu.iq

A Hybrid Technique Based on RF-PCA and ANN for Detecting DDoS Attacks IoT

Hayder Jalo[1], Mohsen Heydarian[1*]
[1] *Faculty of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran*
[*] *Corresponding author E-mail: m.heidarian@azaruniv.ac.ir*

ARTICLE INFORMATION

SUBMISSION HISTORY:

Received: 5 Feb 2024
Revised: 3 April 2024
Accepted: 15 May 2024
Published: 30 June 2024

KEYWORDS:

Network Intrusion Detection System (NIDS); ANN; Feature Selection; IoT; DDoS

ABSTRACT

The increasing reliance on smart products has increased vulnerabilities in Internet of Things (IoT) traffic, which poses significant security risks. These vulnerabilities allowed some hackers to exploit them, which led to system performance degradation. Attacks can lead to these vulnerabilities to various undesirable outcomes, including data leakage, economic losses, data breaches, operational disruptions, and damage to the company's reputation. To address these security challenges, network intrusion detection alarms play a crucial role in assessing system security. In recent years, the proliferation of intelligent and soft computing-based algorithmic and structural frameworks has been evident. However, previous studies have faced challenges related to comprehensiveness, zero-day attacks, realism, and data interpretation. In light of these concerns, this study proposes to design a neural network for proactive detection of attacks. Moreover, we propose to use a hybrid system called RF-PCA to facilitate dimensionality reduction and help classifiers. Notably, this is the first application of a BOT-IoT data set in such an approach. The study also includes a discussion of relevant IoT terms in the context of our work. The proposed method uses high-level data features to represent and draw conclusive conclusions. To evaluate its effectiveness, an experiment was conducted using Python as the programming environment, achieving a remarkable detection rate of 99.73%.

1. INTRODUCTION

With the rapid proliferation of the Internet of Things (IoT), communication between IoT devices no longer relies solely on human interaction or computer intervention. Businesses worldwide have embraced IoT to enhance their operations, and projections indicate that by 2030, approximately 50 billion items, ranging from smartphones to kitchen appliances, will be interconnected [1]. However, the widespread adoption of cutting-edge IoT technologies has also given rise to new security risks. Many IoT devices are used unattended, communicating over wireless networks, making them susceptible to unauthorized access on both physical and logical levels. To ensure the security of IoT networks and devices, it is essential to define proper security requirements during the early phases of IoT device design and deployment [2]. Implementing modern security measures is imperative to safeguard all stakeholders involved in IoT. Traditional security measures, such as encryption, authentication, access control, network security, and application security, were initially employed to address IoT security challenges. Nevertheless, these measures have proven inadequate in meeting the diverse requirements of IoT contexts. Deployed countermeasures against targeted security threats have shown some success, but they are often undermined by evolving attack methodologies. For example, the infamous Mirai botnet exploited IoT devices to orchestrate massive Distributed Denial of Service (DDoS) attacks, utilizing bogus source IP addresses to amplify their impact and evade existing defenses [3]. The constant

DOI: https://doi.org/10.51173/ijds.v1i1.9

evolution of attacks underscores the importance of exploring viable remedies for IoT security. Network attacks can be characterized as malicious actions taken within a computer network, aiming to cause harm, steal sensitive information, or disrupt data flow. Network Intrusion Detection Systems (NIDSs) analyze real-time network data packets and respond to potential threats. However, for NIDSs to be effective in IoT settings, they must cope with challenging conditions, including low energy consumption, limited processing capacity, rapid response times, and handling large data volumes. Thus, enhancing integrated NIDSs for IoT remains a critical and ongoing challenge, necessitating a comprehensive understanding of IoT system security flaws [4], [5] (As shown in Fig .1), which shows the location of NIDS within the network.

One prevalent strategy employed by hackers to disrupt an organization's operations and communication channels is DDoS attacks, which flood systems or websites with fake or bot users. Such attacks are classified into application layer attacks, protocol attacks, and volumetric attacks [6]. Although previous researchers have developed a range of features and datasets, their implementation is resource-intensive and time-consuming, leading to low detection precision and high false positive rates in identifying threats from incoming network data. This research addresses the limitations of DDoS detection through a hybrid feature selection technique, combining Random Forest and Principal Component Analysis (PCA) on the BOT-IoT dataset.

This paper proposes an alternative method for categorizing BOT-IoT attacks, accounting for variations in attack behavior. The key contributions of this study are as follows:

1. An updated comprehensive study of significant findings from previous researchers.
2. An analysis of the BOT-IoT database, including its variants and properties.
3. The proposal of using a holistic system to extract essential features from BOT-IoT datasets using Random Forest and PCA techniques, preserving the data sets' original essence.
4. The classification of attacks associated with the dataset using the Artificial Neural Network (ANN) methodology, with a particular focus on DDoS attacks due to their severity. Two important classifiers, MLP and DNN, are employed to obtain clear results.

The subsequent sections of this paper are organized as follows: Section 2 discusses available NIDS approaches, and the most important terms influencing our work with their explanations including dataset analysis. Section 3 presents the loophole that we are trying to highlight, while Section 4 explains the experimental setup and the results followed by conclusions are written in Sections 5 and 6 of the paper.

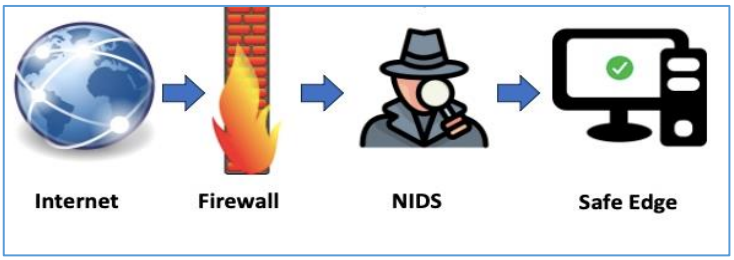


Figure 1. Network Intrusion detection system (NIDS)

2. Related Works

Various studies have contributed to the development of NIDS and the identification of cyberattacks. Some studies have focused on rule-based methods for detecting attacks [7], [8], [9]. These methods include monitoring the nodes that monitor network data, filtering out important information, and applying rules to the filtered data for detection. Comprehensive discussions on multiple attacks and defenses are covered in [10], while [8] presents a distinct approach that uses the activity patterns of neighboring sensor network nodes as inputs. Ferrag et al.[7] proposed a hierarchical model with three classifiers to accurately detect attacks. Manso et al. [6] Provide software-defined identifiers that integrate with Software-Defined Networking (SDN) to ensure network stability by detecting and preventing attacks at an early stage. Notably, their method involved the lowest processing time achieved with the lowest memory usage, even though this method can consume a lot of system resources.

Several studies have employed machine learning techniques to enhance NIDS capabilities. M. Sakr, and Cheema [11], [12] propose an effective anomaly-based NIDS for the cloud, utilizing particle swarm optimization (PSO) to improve SVM classification models. Hajimirzaei and Jafari [13] utilize the fuzzy

clustering method to partition the NSL-KDD intrusion dataset for training the multilayer perceptron network (MLP) with fine-tuning via the ABC method. Aamir et al. [8] suggest a feature engineering-based machine learning approach to detect DDoS attacks, showcasing the importance of feature reduction for improved model performance. [14] devise a honeypot-based botnet detection approach using machine learning, gathering data from an Internet of Things honeypot to identify novel malware families used in botnet attacks. [15], [16] Used a distinctive method in selecting data and reshaping its dimensions, using a hybrid system called RF-PCA, while [17] focused on using PCA as a dimension reducer with the ANN classifier. While previous research lacks a focus on IoT effects, our work addresses the sensitivity of IoT systems and aims to detect BOT-IoT attacks using feature engineering and machine learning. We employ a recent dataset designed for simulations, considering the imbalanced nature of BOT-IoT data. In his research, [18] explained the details of the BOT-IoT data set and what features are considered major, secondary, or neglected, and we will rely on them to delete some of the features that we find unimportant. Table 1. provides an overview of works that demonstrate the viability of the BOT-IoT dataset for real-time intrusion detection.

Table 1. The accuracy of the classifiers for some works that dealt with the BOT-IoT dataset

No.	Article	Year	Dataset	Model	Classification	Acc %
1	Ferrag et al. [3]	2019	BOT-IoT	RNN	Multiclass	98.20
2	Ge et al. [19]	2019	BOT-IoT	DNN	Multiclass	98.09
3	Susilo et al. [20]	2020	BOT-IoT	CNN	Binary	91.00
4	Ge et al. [21]	2020	BOT-IoT	DNN	Multiclass	99.70
5	Ferrag et al. [7]	2020	BOT-IoT	RNN	Multiclass	98.37
6	Aldhaheri et al. [22]	2020	BOT-IoT	ANN, SNN	Multiclass	98.73
7	Ferrag et al. [7]	2020	BOT-IoT	RNN	Binary	98.31
8	Biswas et al. [9]	2021	BOT-IoT	M-GRU	Binary	99.76
9	Ullah et al. [23]	2022	BOT-IoT	LSTM	Multiclass	99.80
10	Ullah et al. [23]	2022	BOT-IoT	GRU	Multiclass	99.87
11	Bovenzi et al. [24]	2020	BOT-IoT	DNN	Multiclass	99.00
12	Lo et al. [25]	2022	BOT-IoT	E- GraphSAGE	Multiclass	99.99
13	Muhammad Shafiq et al. [26]	2020	BOT-IoT	C4.5, RF, Naïve Bayes, SVM	Multiclass	>96
14	Fatani et al. [27]	2021	KDDCup-99, NSL-KDD, BoT-IoT, CICIDS-2017	CNN	Multiclass, Binary	99.99
15	O Alkadi et al. [28]	2020	Bot-IoT, UNSW-BN15	BiLSTM RNN	Binary	98.91
16	Bhuvaneswari Amma et al. [29]	2020	Full Bot-IoT and Best 10 Bot-IoT	VCN	Multiclass	99.75
17	Cheema et al. [12]	2020	Bot-IoT	SVM	Binary	99.99
18	Lawal et al.[30]	2020	Full Bot-IoT	XGBoost	Multiclass, Binary	99.5
19	Guizani et al. [31]	2020	Bot-IoT	RNN-LSTM	Multiclass	96
20	Huong et al. [32]	2021	Bot-IoT	DNN	Binary	99.9
21	Ge et al. [21]	2021	Bot-IoT PCAP	FNN	Binary	99.99
22	Alyasiri et al. [33]	2021	IoT-MQTT, Bot-IoT	GE	Binary	99.98
23	Nadeem Sarwar et al. [34]	2023	Bot-IoT	ANN	Multiclass	98.00
24	Imane Kerrakchou et al. [35]	2023	Bot-IoT	ANN	Multiclass	99.42
25	E.I. Elsedimy et al. [36]	2023	Bot-IoT	ANN	Multiclass	97.62

In our working approach, we approached the following researches [19], [22], [24], [34], [35], [36], the point of difference was that we reduced features using RF-PCA, which produced a detection accuracy that rivaled and perhaps even exceeded these researches in the time factor sometimes. To appreciate the importance of our work and its implications, it was necessary to address 3 subsections dealing with the basic terms of our research.

2.1. Internet of Things (IoT)

The Internet of Things (IoT) encompasses an expansive network that interconnects, communicates, and remotely oversees a limitless array of automated devices. This futuristic landscape envisions everyday objects capable of storing, processing, and transmitting data, offering customers a pay-per-use model for items equipped with processing, storage, and communication capabilities. Given the potential scale of IoT, accommodating billions of interconnected devices, it necessitates substantial data storage and seamless connectivity. Consequently, IoT designs must prioritize scalability, dependability, quality of service, and interoperability to effectively cater to its dynamic demands [33].

2.2. Distributed Denial-of-Service attacks

DDoS is a malicious attack strategy where numerous computers collaborate to flood a server, service, or network with an overwhelming volume of traffic, causing it to become inaccessible or unresponsive [4]. The coordinated nature of DDoS attacks makes it difficult to trace the source effectively. Such attacks can escalate rapidly, bombarding the target with an exponentially increasing number of requests, and exacerbating the impact on the victim's resources. For online businesses and organizations, understanding the threats posed by DDoS attacks is paramount, and they should implement swift and effective mitigation methods to safeguard their services and operations. These attacks are perilous, as they seize control over the victim's resources, aiming to disrupt or cripple the normal functioning of the targeted service [37].

2.3. BOT-IoT Dataset

The Bot-IoT dataset [6], [7], [10], [11], [13], [38], [39] as curated in a controlled testbed environment, comprising multiple virtual computers running various operating systems, network firewalls, network taps, Node-red, and Argus network security tools. To render this raw dataset usable for standard machine learning models, it undergoes preprocessing with network analysis tools such as Wireshark, Argus, or Zeek. Koroniotis et al. [40] suggested employing the 5% Subset of the Bot-IoT dataset, which represents the Full Set while containing 5% of its original instances, totaling 3.6 million instances. It is divided into 2,934,817 records for training and 733,705 records for testing. Within the 5% subset, there are 43 independent features and 3 dependent features, while we find that the dataset has proposed classifying the 10 most important features according to importance too, which consists of 19 columns, including 16 features. Please refer to Table 2. for further details.

Table 2. 5% and 10 best features subset of BOT-IoT

Category	Subcategory	No. of Instances
Normal	Normal	477
DoS/DDoS	TCP	1,593,180
	UDP	1,981,230
	HTTP	2,474
Reconnaissance	OS Fingerprinting	17,914
	Service Scanning	73,168
Information Theft	Keylogging	73
	Data Exfiltration	6

The Bot-IoT dataset encompasses three dependent categorical characteristics: category, subcategory, and attacks. All sets and subsets, except the Raw Set, include these three characteristics. The "Category" feature, represented by string values, consists of five possible categories: "Normal," "DoS," "DDoS," "Reconnaissance," and "Information Theft." These values indicate the type of attack being executed. Additionally, the "Subcategory" feature, also of string type, comprises eight options beyond the commonly found values such as normal, tcp, udp, http, os fingerprinting, service scanning, keylogging, and data exfiltration. The combination of "Category" and "Subcategory" values provides a more detailed characterization of the attacks.

3. Research Gap

While numerous studies have focused on botnet detection models, a research gap exists in the utilization of feature engineering techniques to address issues of duplication and multicollinearity in large datasets. Additionally, the modern problem of IoT botnets is often overlooked in research, as traditional datasets without IoT-traces are commonly employed. Moreover, the use of unbalanced real-time datasets in botnet detection model research poses further challenges. Many researchers prioritize achieving high accuracy on these imbalanced datasets using various machine learning methods but overlook the unequal distribution of data during the training phase. This can lead to misleading results, as accuracy is typically calculated based on the performance of the model on the training dataset rather than the test data.

To bridge this research gap, we propose a comprehensive approach involving feature engineering, machine learning, and resampling techniques to balance a real-time dataset. Specifically, we employ the BOT-IoT dataset, which is generated in an Internet of Things scenario and includes DDoS attack traffic logs, making it a relevant and up-to-date publicly available dataset. We apply RF-PCA, a method for feature scaling and reduction, to enhance the BOT-IoT dataset. Subsequently, we evaluate the updated dataset using two prominent Artificial Neural Network (ANN) algorithms, allowing us to compare and gain insights into the performance of various methodologies and determine the most effective approach among the broad range of available methods.

4. Research Methodology

Our methodology consists of four main steps interspersed with sub-steps; Fig. 2 shows a flowchart of the steps we will follow in our research method.

4.1. Data Pre-processing

This process consists of five important stages. After reading the dataset of the 10-Best features BOT-IoT dataset and classifying the test and training data, as shown in Fig. 3, the first stage comes, where the data must be cleaned by removing records with missing data. The second stage is to remove columns that do not contribute significantly to the importance of the data. Here, two columns are excluded from the targeted columns, which are subcategory and attack, to ensure clarity of results and effective focus on identifying DDoS attacks.

The BOT-IoT dataset includes different types of attributes, including numeric and non-numeric attributes, as shown in Fig. 3. Since ANN requires numeric arrays for both training and testing inputs, non-numeric attributes must be converted to numbers Digital, which also applies to target data and this is the third stage. Then we move to the fourth stage, we must separate the basic features from the target, and create an X and Y combination. Once the BOT-IoT dataset has undergone the necessary reformatting and initialization steps, we move to the balancing phase. Choosing the appropriate approach to address data imbalance greatly influenced our research. The dataset we used showed significant imbalance problems, as shown in Fig. 4.

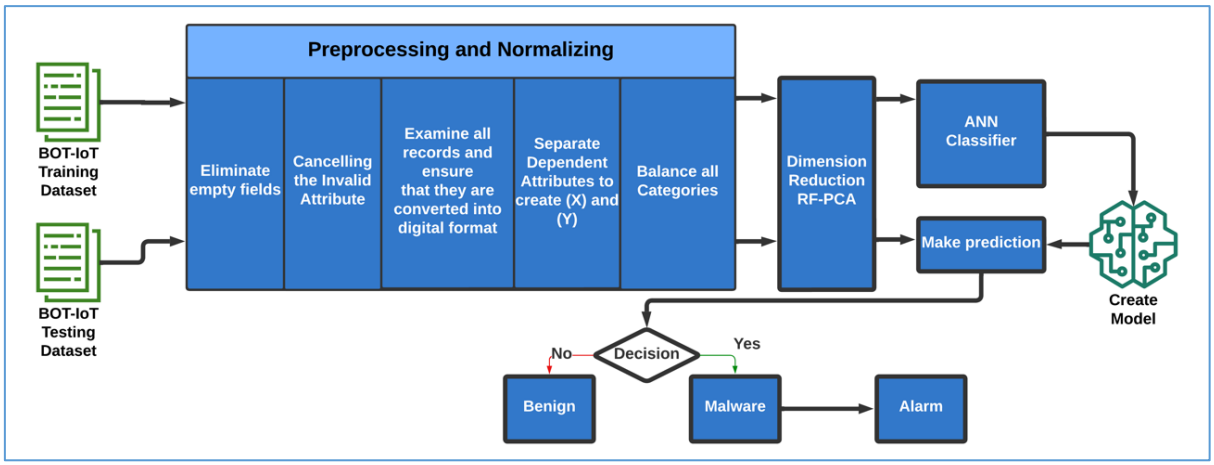


Figure 2. Flowchart of the steps we will follow in our research method

To mitigate this imbalance, we propose to use two important functions: RandomOverSampler, which allows us to augment underrepresented classes by iterating them multiple times, and RandomUnderSampler, used to reduce the iterations of all function values and achieve a balanced state while retaining the basic properties. From the data function.

```
 #   Column            Dtype
---  ------            -----
 0   pkSeqID           int64
 1   proto             object
 2   saddr             object
 3   sport             object
 4   daddr             object
 5   dport             object
 6   seq               int64
 7   stddev            float64
 8   N_IN_Conn_P_SrcIP int64
 9   min               float64
 10  state_number      int64
 11  mean              float64
 12  N_IN_Conn_P_DstIP int64
 13  drate             float64
 14  srate             float64
 15  max               float64
 16  attack            int64
 17  category          object
 18  subcategory       object
dtypes: float64(6), int64(6), object(7)
memory usage: 425.4+ MB
```

```
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   pkSeqID           733705 non-null  int64
 1   proto             733705 non-null  object
 2   saddr             733705 non-null  object
 3   sport             733705 non-null  object
 4   daddr             733705 non-null  object
 5   dport             733705 non-null  object
 6   seq               733705 non-null  int64
 7   stddev            733705 non-null  float64
 8   N_IN_Conn_P_SrcIP 733705 non-null  int64
 9   min               733705 non-null  float64
 10  state_number      733705 non-null  int64
 11  mean              733705 non-null  float64
 12  N_IN_Conn_P_DstIP 733705 non-null  int64
 13  drate             733705 non-null  float64
 14  srate             733705 non-null  float64
 15  max               733705 non-null  float64
 16  attack            733705 non-null  int64
 17  category          733705 non-null  object
 18  subcategory       733705 non-null  object
dtypes: float64(6), int64(6), object(7)
memory usage: 106.4+ MB
```

Figure 3. Train and test dataset information

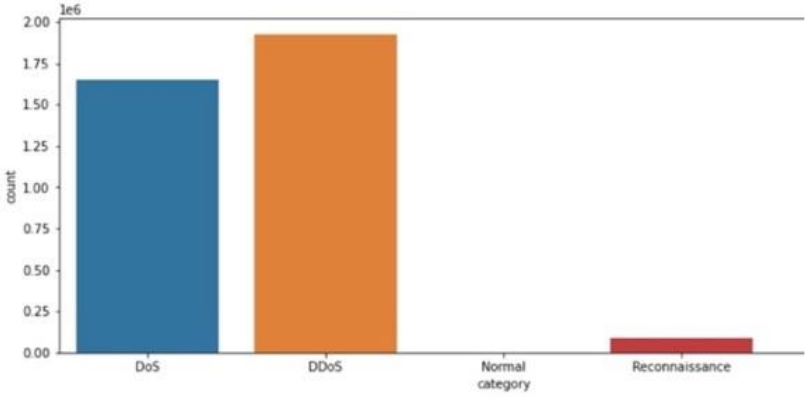


Figure 4. The distribution of attacks in the BOT-IoT dataset

4.2. Feature Selection

In this phase, the resulting dataset undergoes processing through a hybrid system comprising two crucial feature selectors: Random Forest and Principal Component Analysis (PCA). Let us delve into each of these methods separately:

4.2.1. Random Forest (RF)

Introduced by Breiman in 2001, the Random Forest method involves randomly selecting K training samples and constructing a decision tree for each sample. The decision tree nodes are classified to ensure maximal growth. Notably, if the feature dimension of a sample is denoted as M, a constant 'm' much smaller than M is chosen. This study employs a randomized selection of 'm' feature subsets from the 'M' available features to train K decision trees. Combining the predictions from these decision trees yields a more accurate estimation of the prediction target.

Specific proScedures on sample classification outcomes allow for the determination of each feature's significance, reflecting its impact on the prediction outcome. The random forest approach assesses a feature's importance by averaging its relevance across all internal decision trees. The importance of a feature is inversely proportional to its weight in the final forecast, thereby prioritizing impactful features.

The out of bag error is commonly used in RF to rank the significance of features:

$$I_m = \frac{1}{M}\sum(\text{errb2} - \text{errb1}) \qquad (1)$$

Where M is the number of trees in RF, errb2 is the out of bag error when data is subjected to noise interference, and errb1 is the out of bag error when data is collected without interference.

4.2.2. Principal Component Analysis (PCA)

Building on the results obtained from RF, we apply PCA to uncover underlying patterns within the data and assess the similarity or dissimilarity of each attribute in comparison to others. PCA provides an effective approach to comprehending the dataset's characteristics. Both the raw data and its average are used in this process. By computing the covariance matrix, we determine the resulting eigenvalues and eigenvectors, which highlight the principal components in the BOT-IoT dataset that exhibit the strongest correlations. To select the most critical information while excluding less significant aspects, the eigenvalues are ranked from highest to lowest. The algorithm comprises eight essential steps to achieve these objectives:

Step 1: Standardizing data by subtracting the mean and scaling to unit variance is essential.

Step 2: Calculate the covariance matrix of the standardized feature matrix, which displays the feature relationships.

Step 3: Covariance matrix analysis to find eigenvectors and eigenvalues.

Step 4: Sort eigenvectors by eigenvalues in descending order. The number of principal components depends on how much contrast we want to preserve, usually 90% or more.

Step 5: Create a projection matrix from the given eigenvectors.

Step 6: Determine the principal components derived from the top-k transformation to reduce the dimensionality of the dataset.

Step 7: Using the selected principal components, the low-dimensional representation can be inverted to reconstruct the original data.

Step 8: Analyze the transformed dataset to see how the key components capture the structure and patterns of the data.

4.3. ANN Classifier

Neural networks (ANN) are one of the sciences of artificial intelligence and are specifically used in the processes of prediction and classification, and they closely mimic the work of a human nerve cell. Many methods fall under it, such as MLP, DNN, RNN, and LSTM, and it will be adopted in this research as a classifier, focusing on the MLP and DNN methods.

4.4. Test and Evaluation

The evaluation system plays a crucial role in various fields, including machine learning, research, business, and education. Its importance lies in its ability to assess the effectiveness, performance, quality, and impact of systems, processes, models, strategies, or initiatives. The results consist of several terms, the most famous of which we mention:

C_00 represents the count of true negatives.
C_01 represents the count of false positives.
C_10 represents the count of false negatives.
C_11 represents the count of true positives. As shown in Fig. 5.

The higher the accuracy value, the higher the quality of the work:

Accuracy= (TN+TP)/(TN+FP+TP+FN) (2)

Accuracy may not provide an accurate assessment when dealing with imbalanced datasets, where the distribution of negative and positive classes is uneven, and where high accuracy can lead to misinterpretation of results [108], [109]. Consequently, we must explore alternative metrics for classification.

Our focus now shifts to evaluating precision (also known as positive predictive value). The precision formula is:

Precition= TP/(FP+TP) (3)

Next, we will introduce another significant metric known as "recall." Recall is alternatively referred to as "sensitivity" or the "true positive rate," and it's definition is:

Recall= TP/(FN+TP) (4)

In an ideal classifier, we aim for perfect precision and recall, signifying zero false positives (FP) and false negatives (FN). Therefore, it's desirable to have a metric that balances both precision and recall. The F1-score is such a metric, effectively considering both precision and recall in its calculation, defined as in Equation:

F1 Score= 2*(Precition*Recall)/(Precition+Recall) (5)

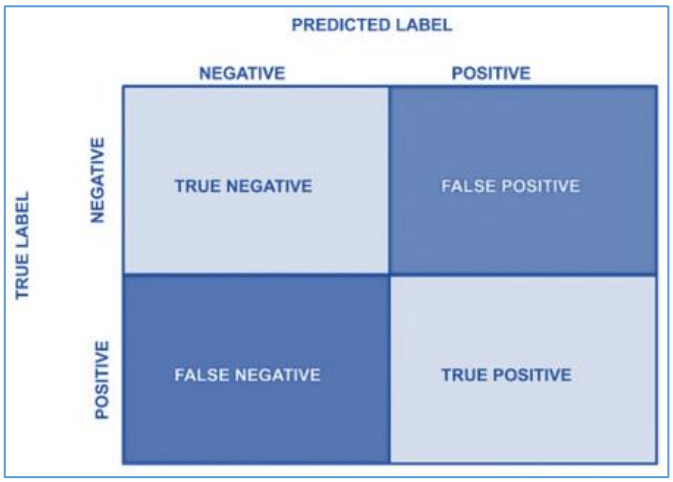


Figure 5. Confusion Matrix

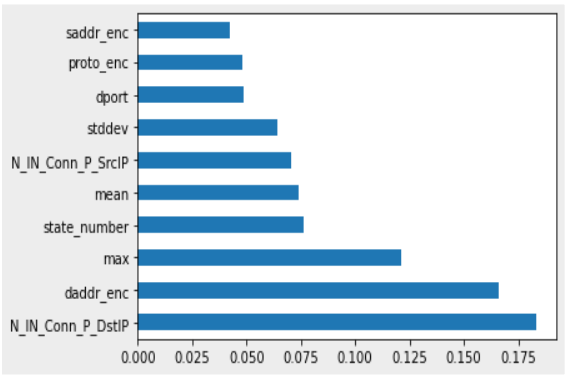
5. Empirical Study and Results

This section presents the results obtained as a result of the proposed method used in our study. A laptop MacBook with a 1.4 GHz Intel Core i5 processor, 4 gigabytes of RAM, and the macOS Big Sur operating system was utilized for the aim of carrying out feature selection and learning studies. A Python program version 3.8.8 with two environments: the root environment (base) and the TensorFlow environment. Fig. 6 shows the pseudocode that is used in programming, It summarizes in more depth all steps of the work.

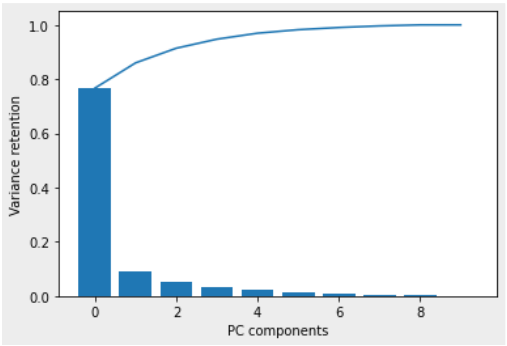
```
#Step 1: Import necessary libraries
#Step 2: Load training and testing of BOT-IoT 10-Best features Dataset
        train_data = pd.read_csv("train_dataset.csv")
        test_data = pd.read_csv("test_dataset.csv")
#Step 3: Drop irrelevant columns from both datasets
        train_data.drop(columns=["irrelevant_column1", "irrelevant_column2"],
        inplace=True(test_data.drop(columns=["irrelevant_column1", "irrelevant_column2"], inplace=True)
#Step 4: Identify and convert mixed data types to a uniform type (if needed)
#Step 5: Encode categorical features using LabelEncoder
        label_encoder = LabelEncoder()
        train_data["categorical_column"] = label_encoder.fit_transform(train_data["categorical_column"])
        test_data["categorical_column"] = label_encoder.transform(test_data["categorical_column"])
#Step 6: Standardize numerical features
        scaler = StandardScaler()
        train_data[["numerical_feature1", "numerical_feature2"]] = scaler.fit_transform(train_data[["numerical_feature1",
        "numerical_feature2"]])
        test_data[["numerical_feature1", "numerical_feature2"]] = scaler.transform(test_data[["numerical_feature1",
        "numerical_feature2"]])
#Step 7: Implement random oversampling and undersampling for class balance in training and testing data
        oversampler = RandomOverSampler()
        X_resampled_train, y_resampled_traisn =
oversampler.fit_resample(train_data.double(columns=["target_column"]), train_data["target_column"])
        X_resampled_test, y_resampled_test = oversampler.fit_resample(test_data.double(columns=["target_column"]),
test_data["target_column"])
        undersampler = RandomUnderSampler()
        X_resampled_train, y_resampled_traisn =
undersampler.fit_resample(train_data.drop(columns=["target_column"]), train_data["target_column"])
        X_resampled_test, y_resampled_test = undersampler.fit_resample(test_data.drop(columns=["target_column"]),
test_data["target_column"])
 #Step 8: Initialize and train an ExtraTreesClassifier RF model
        rf_model = ExtraTreesClassifier()
        rf_model.fit(X_resampled_train, y_resampled_train)
#Step 9: Drop less important features based on model analysis RF
#Step 10: Apply PCA for dimensionality reduction
        pca = PCA(n_components=num)  # Adjust the number of components as needed
        X_train_pca = pca.fit_transform(X_resampled_train)
        X_test_pca = pca.transform(X_resampled_test)
#Step 11: Transform training and testing data using PCA
#Step 12: Define a custom callback for training termination
        custom_callback = CustomCallback()  # Define your custom callback as needed
#Step 13: Build a simple neural network model
        model = keras.Sequential([
                keras.layers.Dense(64, activation='relu', input_shape=(num)),
               keras.layers.Dense(32, activation='relu')
                keras.layers.Dense(1, activation='sigmoid')
                    ])
#Step 14: Train the model with callbacks
        model.compile(optimizer='adam,'
          loss='binary_crossentropy,'
          metrics=['accuracy'])
```

Figure 6: Complete Pseudo-code

In the cleansing step we clean the data by removing columns that do not contribute significantly to the significance of the data, namely pkSeqID and seq, because they only represent the sequence of records in the table, have no value and can affect the validity of the results. In addition to the target columns that we referred to in Section 3.1. The results obtained from the application of RF mechanics were to summarize our resulting database into 10 features only, as shown in Fig. 7A, which were then translated by PCA into 9 new dimensions, as shown in Fig. 7B.



A



B

Figure 7: A-B The importance of features

The 9 attributes are entered along with the target column into the MLP classifier, the results showed an accurate performance as expected, and therefore it is possible to draw the confusion matrix as shown in (Table. 3). The result of the false negative was only 4 cases not DDoS attacks, but the model showed readings of up to 287 cases of false positives, and therefore there is a service that was withheld from a certain number of legitimate users by mistake, but it is a small percentage about the safety provided by the model. (Table. 4) represents a reading of the evolution of accuracy in our work, and shows the results of the Precision, Recall and F1 evaluators for each of the four types of attacks. Each type corresponds to the number of samples taken, and we will call it (Samples).

Table 3. The Confusion Matrix of our results after Appling the MLP Model

		Predict label			
		DDoS	DoS	Normal	Reconnaissance
Actual label	DDoS	14361	142	0	81
	DoS	12	14565	0	7
	Normal	287	0	14297	0
	Reconnaissance	57	41	4	14482

The best results were obtained on iteration No. 27, where the accuracy rate was 99.29%, with a time of 2 msec for one record, with a rate of 16 sec to train and test the entered dataset.

Table 4. The performance results obtained after implementing the MLP model

	Precision	Recall	F1-score	Samples
DDoS	0.98	0.98	0.98	14584
DoS	0.99	1.00	0.99	14584
Normal	1.00	0.98	0.99	14584
Reconnaissance	0.99	0.99	0.99	14584

We then returned the same data from the previous experiment, but this time on a DNN classifier, so that the confusion matrix was as shown in (Table. 5). The results showed faster performance from

the MLP classifier; thus, it is possible to plot the results appear as shown in (Table. 6), which represents a reading of the evolution of accuracy in our work.

Compared to the MLP model, DNN false negatives were only 3 cases not DDoS attacks, but the model showed reads of up to 137 cases of DoS false positives, so there is a service that has been withheld from a certain number of legitimate users by mistake, but it is also small percentage compared to the safety offered by the model. The best results were obtained on iteration No. 95, where the accuracy rate was 99.73%, with a time of 2 msec for one record, with a rate of 20 sec to train and test the entered dataset. We also noticed that there was a large saturation that occurred after reaching the highest accuracy, which we explain by the model reaching a state of overfitting. Fig. 8 shows a comparison between the accuracy of the model with each iteration and the amount of loss for the same moment in the MLP classifier, while Fig. 9 shows the same comparison in the DNN classifier. The curves indicate the progress of accuracy relative to the number of iterations for the training and test data that we used in validation.

Table 5. The Confusion Matrix of our results after applying the DNN Model

		Predict label			
		DDoS	DoS	Normal	Reconnaissance
Actual label	DDoS	14477	39	0	68
	DoS	26	14554	0	2
	Normal	0	137	14447	0
	Reconnaissance	9	13	3	14559

Table 6. The results we obtained after implementing the DNN model

	Precision	Recall	F1-score	Samples
DDoS	1.00	0.99	1.00	14584
DoS	0.99	1.00	0.99	14584
Normal	1.00	0.99	1.00	14584
Reconnaissance	1.00	1.00	1.00	14584

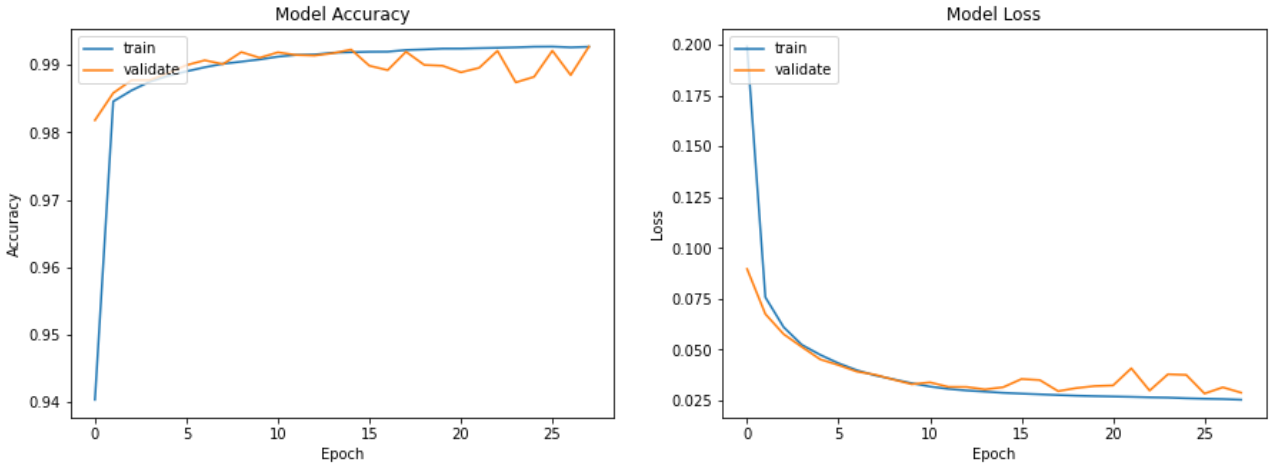


Figure 8: Accuracy and loss curves for MLP (Best 10 features Dataset)

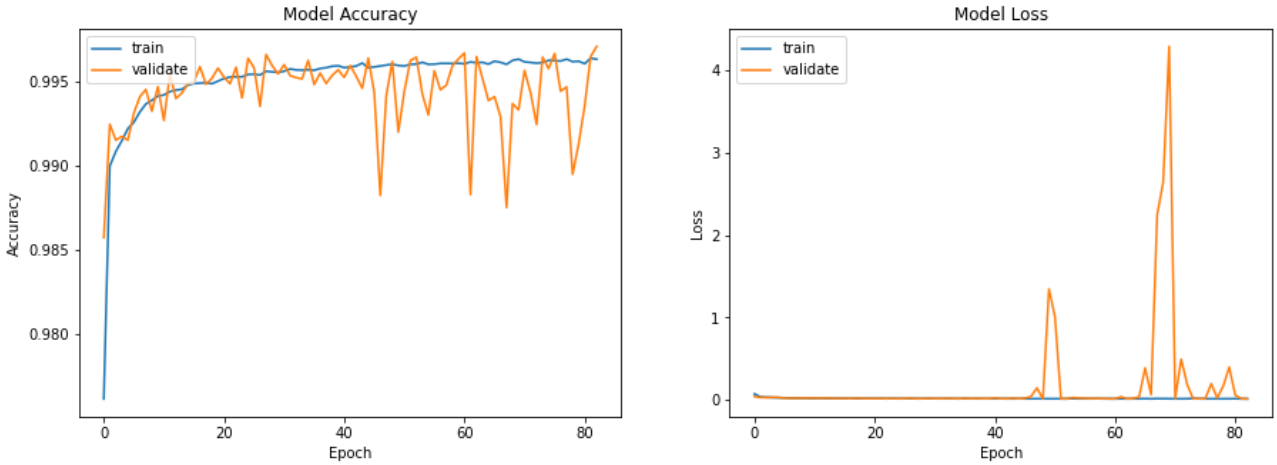


Figure 9: Accuracy and loss curves for DNN (Best 10 features Dataset)

The final step includes validating our results by comparing them to the results of previous researches to demonstrate their importance and scientific value, as shown in Fig. 10. The accuracy of the system was calculated programmatically according to the number of correct classifications that were calculated, detection accuracy, and error rates, and verifying the values using the confusion matrix, which was later used to calculate the accuracy of the four target categories detection.

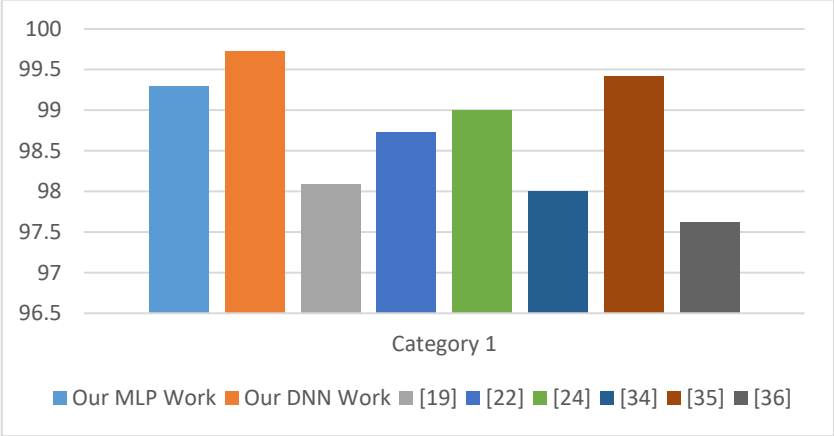


Figure 10: Comparison with the best similar researches

While the accuracy of the training results in both classifiers showed 99.99%. The results of the MLP classifier showed results that were close to those of similar researches, while the results of the DNN classifier showed that our results were superior to the six closest similar modern research in terms of the data set and the convergence of the methods used in classification, which indicated the ideal work of our system. The MLP classifier showed an accuracy of 99.29%, while the DNN classifier showed an accuracy of 99.73% in the test results for the 10 best features subset of the BOT-IoT dataset.

6. Conclusion and Future Works

Identifying crucial attributes holds significant importance in this study. By employing RF-PCA, the high-dimensional vector comprising 16 diverse features was efficiently reduced to lower configurations, leading to an impressive classification accuracy of 99.73% as per the DNN classifier. The primary objective is to detect intrusions, specifically DDoS attacks, and train the model to anticipate and defend against potential zero-day attacks. To facilitate the learning of ANN algorithms, it was necessary to simplify the set of data and find solutions to reduce it, so we used MLP and DNN algorithms with an optimized set of new properties. The outcomes demonstrate the remarkable performance of the RF-PCA combination in feature reduction while preserving the dataset's core, as evident from the substantially higher classification accuracy compared to previous articles. The experiment gave importance to the

time factor too, yielding promising results. Notably, the dataset used in the experiment aligns closely with IoT operations, leading to satisfactory findings. In the subsequent research phase, we aim to expand the experiment by incorporating additional data sets or employing more efficient classifiers, thereby seeking to attain even greater accuracy with reduced time consumption.

Acknowledgments

This research is supported by Azarbaijan Shahid Madani University and conducted within the framework of the MSc program.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Reference

[1] A. Holst, "Number of Connected Devices Worldwide 2030; Statista," *Statista: Hamburg, Germany*, 2018.

[2] M. Wazzan, D. Algazzawi, O. Bamasaq, A. Albeshri, and L. Cheng, "Internet of things botnet detection approaches: Analysis and recommendations for future research," *Applied Sciences (Switzerland)*, vol. 11, no. 12. MDPI AG, Jun. 02, 2021. doi: 10.3390/app11125713.

[3] M. A. Ferrag and L. Maglaras, "DeepCoin: A Novel Deep Learning and Blockchain-Based Energy Exchange Framework for Smart Grids," in IEEE Transactions on Engineering Management, vol. 67, no. 4, pp. 1285-1297, Nov. 2020, doi: 10.1109/TEM.2019.2922936.

[4] A. Yaseen Abdulrazzak, S. Latif Mohammed, A. Al-Naji, and J. Chahl, "Computer-Aid System for Automated Jaundice Detection," *Journal of Techniques*, vol. 5, no. 1, pp. 8–15, Mar. 2023, doi: 10.51173/jt.v5i1.1128.

[5] Asaad Yaseen Ghareeb, S. K. Gharghan, A. H. M. Mutlag, and Rosdiadee Nordin, "Wireless Sensor Network-Based Artificial Intelligent Irrigation System: Challenges and Limitations," *Journal of Techniques*, vol. 5, no. 3, pp. 26–41, Aug. 2023, doi: 10.51173/jt.v5i3.1420.

[6] P. Manso, J. Moura, and C. Serrão, "SDN-based intrusion detection system for early detection and mitigation of DDoS attacks," *Information (Switzerland)*, vol. 10, no. 3, 2019, doi: 10.3390/info10030106.

[7] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "RDTIDS: Rules and decision tree-based intrusion detection system for internet-of-things networks," *Future Internet*, vol. 12, no. 3, Mar. 2020, doi: 10.3390/fi12030044.

[8] M. Aamir and S. M. A. Zaidi, "DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation," *Int J Inf Secur*, vol. 18, no. 6, pp. 761–785, Dec. 2019, doi: 10.1007/s10207-019-00434-1.

[9] R. Biswas and S. Roy, "Botnet traffic identification using neural networks," *Multimed Tools Appl*, vol. 80, no. 16, pp. 24147–24171, Jul. 2021, doi: 10.1007/s11042-021-10765-8.

[10] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures," *IEEE Access*, vol. 7. Institute of Electrical and Electronics Engineers Inc., pp. 82721–82743, 2019. doi: 10.1109/ACCESS.2019.2924045.

[11] M. M. , T. M. A. , & E.-S. A. B. Sakr, "An efficiency optimization for network intrusion detection system. International Journal of Computer Network and Information Security," vol. 11(10), no. 1, 2019, doi: 10.5815/ijcnis.2019.10.01.

[12] M. A. Cheema, H. K. Qureshi, C. Chrysostomou, and M. Lestas, "Utilizing Blockchain for Distributed Machine Learning based Intrusion Detection in Internet of Things," in *Proceedings - 16th Annual International Conference on Distributed Computing in Sensor Systems, DCOSS 2020*, Institute of Electrical and Electronics Engineers Inc., May 2020, pp. 429–435. doi: 10.1109/DCOSS49796.2020.00074.

[13] B. Hajimirzaei and N. J. Navimipour, "Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm," *ICT Express*, vol. 5, no. 1, pp. 56–59, Mar. 2019, doi: 10.1016/j.icte.2018.01.014.

[14] C. , C. R. L. , S. J. , W. C. , A. M. , S. A. , & P. A. Dietz, "IoT-botnet detection and isolation by access routers," in *In 2018 9th International Conference on the Network of the Future (NOF)* , 2018, pp. 88–95. doi: 10.1109/NOF.2018.8598138.

[15] J. , S. K. M. K. C. D. S. and R. D. Vandarkuzhali, "Hybrid RF and PCA method: The number and Posture of piezoresistive sensors in a multifunctional technology for respiratory monitoring," *Sensors*, vol. 29, no. 100832, 2023. doi: 10.1016/j.measen.2023.100832.

[16] K. M. Z. F. H. and W. L. Bian, "RF-PCA: A new solution for rapid identification of breast cancer categorical data based on attribute selection and feature extraction," *Front Genet*, vol. 11, no. 566057, 2020. doi: 10.3389/fgene.2020.566057.

[17] R. Gopi *et al.*, "Enhanced method of ANN based model for detection of DDoS attacks on multimedia internet of things," *Multimed Tools Appl*, vol. 81, no. 19, pp. 26739–26757, Aug. 2022, doi: 10.1007/s11042-021-10640-6.

[18] Peterson, J. M., Leevy, J. L., and Khoshgoftaar, "A review and analysis of the bot-iot dataset," in *In 2021 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, IEEE, Aug. 2021, pp. 20–27. doi: 10.1109/SOSE52839.2021.00007.

[19] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep learning-based intrusion detection for IoT networks," in *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing, PRDC*, IEEE Computer Society, Dec. 2019, pp. 256–265. doi: 10.1109/PRDC47002.2019.00056.

[20] B. Susilo and R. F. Sari, "Intrusion detection in IoT networks using deep learning algorithm," *Information (Switzerland)*, vol. 11, no. 5, Jun. 2020, doi: 10.3390/INFO11050279.

[21] Ge, M., Syed, N. F., Fu, X., Baig, Z., & Robles-Kelly, A. (2021). Towards a deep learning-driven intrusion detection approach for Internet of Things. Computer Networks, 186, 107784. https://doi.org/10.1016/j.comnet.2020.107784.

[22] S. Aldhaheri, D. Alghazzawi, L. Cheng, B. Alzahrani, and A. Al-Barakati, "DeepDCA: Novel network-based detection of iot attacks using artificial immune system," *Applied Sciences (Switzerland)*, vol. 10, no. 6, Mar. 2020, doi: 10.3390/app10061909.

[23] I. Ullah and Q. H. Mahmoud, "Design and Development of RNN Anomaly Detection Model for IoT Networks," *IEEE Access*, vol. 10, pp. 62722–62750, 2022, doi: 10.1109/ACCESS.2022.3176317.

[24] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico and A. Pescapé, "A Hierarchical Hybrid Intrusion Detection Approach in IoT Scenarios," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, Taipei, Taiwan, 2020, pp. 1-7, doi: 10.1109/GLOBECOM42002.2020.9348167.

[25] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher and M. Portmann, "E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT," NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 2022, pp. 1-9, doi: 10.1109/NOMS54207.2022.9789878.

[26] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "IoT malicious traffic identification using wrapper-based feature selection mechanisms," *Comput Secur*, vol. 94, Jul. 2020, doi: 10.1016/j.cose.2020.101863.

[27] Fatani *et al.*, "IoT intrusion detection system using deep learning and enhanced transient search optimization," *IEEE* , vol. 9, no. 123448–123464, 2021, doi: 10.1109/ACCESS.2021.3109081.

[28] O. Alkadi, N. Moustafa, B. Turnbull, and K. K. R. Choo, "A Deep Blockchain Framework-Enabled Collaborative Intrusion Detection for Protecting IoT and Cloud Networks," *IEEE Internet Things J*, vol. 8, no. 12, pp. 9463–9472, Jun. 2021, doi: 10.1109/JIOT.2020.2996590.

[29] B. A. Bhuvaneswari and S. S., "Anomaly detection framework for Internet of things traffic using vector convolutional deep learning approach in fog environment," *Future Generation Computer Systems*, vol. 113, pp. 255–265, Dec. 2020, doi: 10.1016/j.future.2020.07.020.

[30] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "An anomaly mitigation framework for iot using fog computing," *Electronics (Switzerland)*, vol. 9, no. 10, pp. 1–24, Oct. 2020, doi: 10.3390/electronics9101565.

[31] N. and A. G. Guizani, "A network function virtualization system for detecting malware in large IoT based networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1218–1228, 2020, doi: 10.1109/JSAC.2020.2986618.

[32] T. T. Huong, T. P. Bac, D. M. Long, B. D. Thang, T. D. Luong, and N. T. Binh, "An Efficient Low Complexity Edge-Cloud Framework for Security in IoT Networks," in *ICCE 2020 - 2020 IEEE 8th International Conference on Communications and Electronics*, Institute of Electrical and Electronics Engineers Inc., Jan. 2021, pp. 533–539. doi: 10.1109/ICCE48956.2021.9352046.

[33] H. Alyasiri, J. A. Clark, A. Malik and R. d. Fréin, "Grammatical Evolution for Detecting Cyberattacks in Internet of Things Environments," 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 2021, pp. 1-6, doi: 10.1109/ICCCN52240.2021.9522283.

[34] N. Sarwar, I. S. Bajwa, M. Z. Hussain, M. Ibrahim, and K. Saleem, "IoT Network Anomaly Detection in Smart Homes Using Machine Learning," *IEEE Access*, vol. 11, pp. 119462–119480, 2023, doi: 10.1109/ACCESS.2023.3325929.

[35] I. Kerrakchou, A. A. El Hassan, S. Chadli, M. Emharraf, and M. Saber, "Selection of efficient machine learning algorithm on Bot-IoT dataset for intrusion detection in internet of things networks," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 31, no. 3, pp. 1784–1793, Sep. 2023, doi: 10.11591/ijeecs.v31.i3.pp1784-1793.

[36] E. I. Elsedimy and S. M. M. Abohashish, "FCM-SWA : Hybrid Intelligent Approach Combining Fuzzy C-Means and Sperm Whales Algorithm for Cyber-Attack Detection in IoT Networks," 2023, doi: 10.21203/rs.3.rs-3515647/v1.

[37] N. F. Syed, Z. Baig, A. Ibrahim, and C. Valli, "Denial of service attack detection through machine learning for the IoT," *Journal of Information and Telecommunication*, vol. 4, no. 4, pp. 482–503, 2020, doi: 10.1080/24751839.2020.1767484.

[38] F. Hussain, S. A. Hassan, R. Hussain and E. Hossain, "Machine Learning for Resource Management in Cellular and IoT Networks: Potentials, Current Solutions, and Open Challenges," in IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1251-1275, Secondquarter 2020, doi: 10.1109/COMST.2020.2964534.

[39] Dataset link: https://research.unsw.edu.au/projects/bot-iot-dataset.

[40] Koroniotis, "Designing an effective network forensic framework for the investigation of botnets in the Internet of Things," 2020, doi: 10.26190/unsworks/2194