An Efficient Intrusion Detection System to Combat Cyber Threats using a Deep Neural Network Model

Mangayarkarasi Ramaiah[1,*], C. Vanmathi[1], Mohammad Zubair Khan[2], M. Vanitha[1] & M. Deepa[1]

[1]School of Computer Science Engineering and Information Systems, Vellore Institute of Technology, Vellore, Tamilnadu, India, 632014
[2]Department of Computer Science and Information, Taibah University, Medina, Saudi Arabia, 41477
*E-mail: rmangayarkarasi@vit.ac.in

Abstract. The proliferation of Internet of Things (IoT) solutions has led to a significant increase in cyber-attacks targeting IoT networks. Securing networks and especially wireless IoT networks against these attacks has become a crucial but challenging task for organizations. Therefore, ensuring the security of wireless IoT networks is of the utmost importance in today's world. Among various solutions for detecting intruders, there is a growing demand for more effective techniques. This paper introduces a network intrusion detection system (NIDS) based on a deep neural network that utilizes network data features selected through the bagging and boosting methods. The presented NIDS implements both binary and multiclass attack detection models and was evaluated using the KDDCUP 99 and CICDDoS datasets. The experimental results demonstrated that the presented NIDS achieved an impressive accuracy rate of 99.4% while using a minimal number of features. This high level of accuracy makes the presented IDS a valuable tool.

Keywords: *artificial deep neural network; correlation tool; DDoS; machine learning; network intrusion detection system; RF-score; XGBoost-score.*

1 Introduction

The pervasive integration of the internet into daily life exposes communication networks to increasing vulnerabilities due to recent technological advancements. With rising internet usage generating substantial amounts of network data, attackers exploit this abundance to introduce novel threats, necessitating robust security measures. Intrusion detection systems (IDS) have emerged as critical tools for safeguarding networks by continuously monitoring traffic data [1]. The main functionality of IDS is to intelligently conceal data to protect valuable credentials from potential attacks. Advanced security challenges, such as zero-day outbreaks, particularly impact countries like the US and Australia, with

Received August 21st, 2022, Revised October 15th, 2023, Accepted for publication November 16th, 2023.
 ISSN: 2337-5787, DOI: 10.5614/itbj.ict.res.appl.2023.17.3.2

Symantec reporting three billion zero-day cases in 2016 [2]. Situational awareness is crucial for identifying devices and services to ensure cyber security [3]. IDS, broadly classified into anomaly-based and misuse-based detection, plays a vital role in detecting intrusion actions [4]. In contrast, the anomaly-based intrusion [5] detection system detects malicious actions that deviate from normal network patterns. Thus, network behavior is a major aspect in detecting intrusions. Various intrusion detection techniques, including statistical-based, data mining, machine learning, and deep learning methods, have been explored in the literature [6-9]. Reference [10] discusses the application of classical machine learning (ML) and data mining (DM) methods for simulating intrusion detection software. The study explains the merits of using the ML and DL methods on benchmark datasets along with the challenges involved.

As a global trend, developing countries are constructing smart cities, where IoT plays a pivotal role. The surge in smart devices connected to the internet raises security and privacy concerns, highlighting the demand for IDS's to mitigate attacks in smart IoT environments [11]. This reference reports a study on IoT security threats underscores the importance of robust network intrusion detection systems (NIDS) in smart IoT environments, where attacks vary in difficulty and diversity. An analysis report on state-of-the-art NIDS methodologies for IoT devices supports classical machine learning implementations [11]. Similarly, unmanned aerial vehicles (UAVs) play a crucial role in civilian and military missions, necessitating robust IDS mechanisms to detect vulnerabilities in interconnected UAVs [12]. With the evolution of IoT to Mobile IoT (M-IoT), the study reported in [13] investigated various solutions to ensure mobile network security and privacy.

Overall, these studies contribute to understanding the complex landscape of intrusion detection across diverse technological domains and offer insight into emerging threats and solutions. An implementation of a machine learning framework for detecting DDoS attacks aimed at distributed Fog environments [14]. The presented framework ensures load balancing, a crucial aspect in the IoT sector. In addition to the above, the framework efficiently handles DDoS attacks while implementing smart contracts through blockchain technology. The system's efficacy was experimented with using the BoT-IoT dataset and achieved a detection rate of 99.9%.

Blockchain technologies to ensure privacy and security for IoT devices are an emerging research topic. Reference [15] summarizes the possible security and privacy issues anticipated in smart IoT environments. Moreover, security issues in the IoT protocol stack and possible solutions to mitigate the same are also discussed.

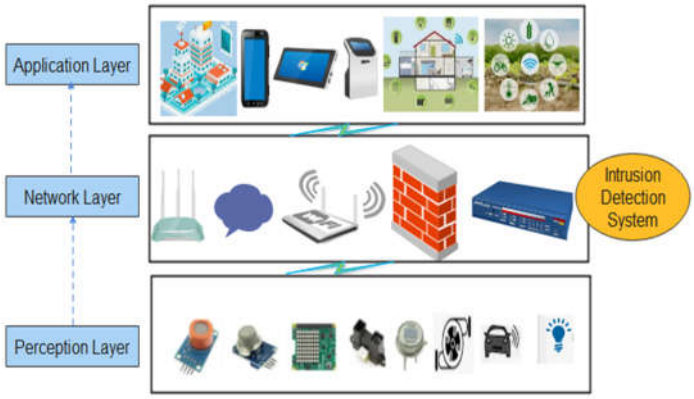

Figure 1 Intrusion detection system for IoT environments.

Shafiq, *et al.* [16] presented a novel feature selection module for improving an entropy-based attack classifier. The proposed framework extracts the effective features from the BoT-IoT dataset using correlation and the AUC metric. The effective features were validated against the labels with the Shannon Entropy-based classifier. Reference [17] presents a novel feature selection method to identify the most influential features from the BoT-IoT dataset. The experimental results obtained through a machine learning classifier showed an improvement compared to the presented feature selection method. The authors in [18] present IDS software specifically meant for IoT wireless networks. The predominant features for the IDS are detected through simulated annealing and the IDS software experimented with NaBIoT. A support vector machine learning model was built upon the selected features to detect attack traffic in wireless networks, achieving an accuracy of 95%.

The presented NIDS is more or less the same as the IDS presented in [19]. The presented work not only focuses on building an IDS with a minimal number of features but also aimed to achieve good accuracy in attack traffic detection. A robust smart network intrusion detection system is presented in this paper. The contribution of the NIDS in detecting and mitigating cyber-attacks anticipated in IoT networks is displayed in Figure 1. The proposed IDS framework comprises a hybrid feature selection phase and a deep neural based cyber-attack classifier. The main highlight of the presented work is the utilization of the merits of the bagging and boosting methods for detecting the predominant features. The results in the experimental section proved that the selected features improved the deep optimized neural network cyber-attack classifier. Such an NIDS framework is especially suitable for IoT networks, where identifying intruders with less features complements the cyber-threats.

The contributions of the proposed work are as follows:

1. A feature selection algorithm using bagging and boosting is proposed.
2. A neural classifier was built upon fine-tuned features.
3. The results were compared with other state-of-the-art counterpart techniques.

This paper is organized into five sections. Section 2 presents related works on the research problem. The proposed IDS framework is presented in Section 3. Section 4 discusses the results and their performance and is followed by the conclusion in Section 5.

2 Related Works

Recently, anomaly-based network intrusion detection has produced promising results in detecting cyber-attacks on IoT networks. Hence, this section discusses the state-of-the-art NIDS frameworks built upon AI techniques presented in various previous studies for the reader's perusal.

2.1 Machine Learning Enabled NIDS

Utilizing machine learning for cyber-attack detection is a powerful strategy, often involving offline models based on network flow or packet data attributes. Feature selection is critical for model effectiveness, and innovative approaches, such as pigeon-inspired optimization, have been explored for this purpose. Researchers, as exemplified in [20], employ datasets like KDDCUP99, NLS-KDD, and UNSW-NB15 for intrusion detection benchmarks. Decision tree-based attack detection models are trained with selected features to enhance accuracy and efficiency. The study reported in [21] improved intruder identification using PCA and hyper-parameter tuning with ANN and SVM on UNSW-NB15. RFE-RF was taken into account in [22] when deciding which UNSW-NB15 dominant characteristics should be used to train the ML-based NIDS. Addressing unbalanced samples in [23] reduces the training sample size through oversampling and identifies dominant features using statistical techniques. Feature modification, such as Bayesian-based transformation in [24], enhances SVM-based models on datasets like NSL-KDD and CICIDS2017. Reference [25] introduced TON_IoT, a novel dataset for dynamic NIDS, achieving improved RF performance. Similarly, [26] focuses on identifying prominent traits and addresses imbalanced samples using the UNSW-NB15 and TON_IOT datasets for ML model training. Overall, these studies reflect ongoing efforts to advance machine learning techniques in cyber security, which is crucial for our digital and interconnected world.

2.2 Deep Learning Enabled NIDS

Researchers responding to the anticipated rise in cyber threats have developed sophisticated intrusion detection systems with complex architectures. In one approach, as outlined in [27], a deep neural network-based NIDS is optimized with multiple layers to effectively identify cyber-attacks. Another strategy, presented in [28], employs ensemble models to mitigate the impact of cyber-attacks, demonstrating superior performance on the UNSW-NB15 dataset compared to deep learning techniques. Furthermore, [29] explored the benefits of deep learning, ensemble learning, and traditional machine learning in IDS design, with ensemble learning techniques proving more effective in attack detection on the same dataset.

A deep long term-short memory (DLSTM) based attack detector is proposed in [30]. The presented IDS model analyzes the feature dependencies through information gain. The NIDS framework presented in [31] leverages the CNN architecture. This approach treats network data as a time series pattern, which is a novel way to detect intrusions. Additionally, the framework has been compared with other architectures commonly used in intrusion detection systems, including multilayer perceptron (MLPs), long short-term memory (LSTM) networks and gated recurrent units (GRUs)

In parallel, specific emphasis on precision is evident in [32], where the proposed High-Precision Intrusion Detection System utilizes a K-dependency Bayesian network to minimize false alarms and accurately identify intrusion attempts in edge computing environments. Additionally, [33] tailored an IDS for vehicle networks, achieving low false-negative rates with a deep convolutional neural network (CNN) that outperformed classical machine learning methods. Moreover, [34] designed a CNN IDS specifically for detecting denial of service (DoS) attacks that achieved high precision, with an accuracy rate of 99.87%. The works presented in [35]-[37] use diverse machine learning methods upon KDDCup99 (KDD) and NSL-KDD (NSL) for the attempted task, weighing their pros and cons and providing insight into future directions for network intrusion detection.

Lastly, [38] introduced a deep learning-based NIDS for software-defined networking, emphasizing the importance of adapting intrusion detection systems to evolving network architectures and assessing feature relevance in software-defined networking (SDN) environments.

2.3 Federated Learning Based NIDS

Federated learning (FL) is emerging as a pivotal solution in the realm of machine learning, particularly addressing concerns of data privacy and security. By

enabling model training across decentralized devices while keeping data locally stored, FL circumvents the need to share sensitive information with a central server, thus safeguarding privacy. One of the challenges in FL is the communication delay incurred when transmitting model updates from devices to the central server and back. To reduce this delay, various techniques, including an attention mechanism, [39] can be employed. However, challenges like implementation complexity and data imbalance across devices must be addressed. FL operates by distributing a global model to participant nodes, allowing local training on individual data without [40] sharing it centrally. Model updates are then transmitted back to the central server, minimizing data exposure.

Moreover, in the context of intrusion detection, bio-inspired optimization techniques, as highlighted in References [41]-[43], showcase innovative approaches to enhance software functionality by optimizing features and parameters. The incorporation of ensemble methods like bagging and boosting in intrusion detection frameworks underscores the ongoing pursuit of accuracy improvement in this critical domain.

3 Proposed System

The proposed simple and reliable NIDS framework's schematic diagram can be seen in Figure 2. It first takes the characteristics from network flow data. Before being fed into the attack-detection model, the retrieved characteristics undergo preprocessing. The novel solution applies ensemble feature selection techniques to the extracted dataset to eliminate redundant features while maximizing the benefits of the bagging and boosting approaches. Thirty percent of the samples are used for testing, while another thirty percent are used for training the framework. A deep neural network model was created based on handpicked attributes to accurately categorize regular traffic despite the presence of malicious traffic. Finally, quantitative metrics were used to evaluate how effective the proposed framework was.

Pseudocode1: Pre-processing

Input: Data set D
Output: Preprocessed Data set D'

Step1: Load the dataset D
Step 2: Encode the string values of the D into numeric values X
X={0,1,2,3,4,……n-1} where n is number of distinct data values
Step 3: Normalize the encoded data X.
Step 4: Store the Normalized dataset S in D'
D'=S
Step 5: return D'

Pseudocode 2: Feature Selection

Input: Normalized features set $D'(F_1, F_2, F_3 ....F_k)$

Output: Dominant feature subset S_{best}

Step1: Compute correlation value C_{ij} for every pair of features F_i and F_j among k features
for $i = 1$ to k

$$C_{ij} = \frac{\sum (F_i - \overline{F}_i)(F_j - \overline{F_j})}{\sqrt{\sum (F_i - \overline{F_i})^2} \sqrt{\sum (F_j - \overline{F_j})^2}}$$

end

Step 2: If the correlation value C_{ij} > threshold then remove it from the input set $D'(f1, f2.f3....f_k)$

Step 3: Apply the XGBoost method on the resultant subset from Step 2

Step 4: Apply Random Forest classifier on the subset of features from Step 2

Step 5: Apply set union operation on the result from Step 3 and Step 4 to produce final set of features.

Step 6: Return the resultant feature set S_{best}

Pseudocode 3: A deep neural-based classification model

Input: Train, Test set.

Output: Accuracy, F1-score, Precision, Recall

Step 1: Load the dataset

Step 2: Train the model using the training samples

Step 3: Continue the training until it achieves the good accuracy

Step 4: Display the results

3.1 Dataset Description

The KDDCUP99 and CICSDDoS datasets served as the basis for the prospective NIDS system. The KDDCUP99 dataset is made up of 38 numerical features and three category features [44]. In total, there were 1,35,973 samples in the experimental dataset. Attack methods covered by KDDCUP99 include DoS (Denial of Service), R2L (unauthorized remote access), U2R (unauthorized local superuser privileges access), and probing. Additionally, Table 1 contains examples of typical network activities that are part of routine traffic. Figure 3(a) shows the number of samples against the various forms of attacks.

Table 1 Categories of attacks as per KDDCUP 99 dataset.

Categories	Meanings
Normal	Normal Traffic – no intrusion found
DoS	Denial of Service
Probing	Monitoring other's behaviors to particularly obtain their data
R2L	Illegal access to data from a remote machine
U2R	Unauthorized access for escalation of privileges

The data from the other dataset, CICDDoS [45], was compiled from the CSE-CIC-IDS2018-AWS, CICIDS2017, and CICDoS (2016) datasets to test the resilience of the presented proposal. The experimental dataset consisted of 101,295 rows of samples with 84 columns against two labels, 'Benign' and 'DDoS'. The sample distribution statistics against the output labels are shown in Figure 3(b).

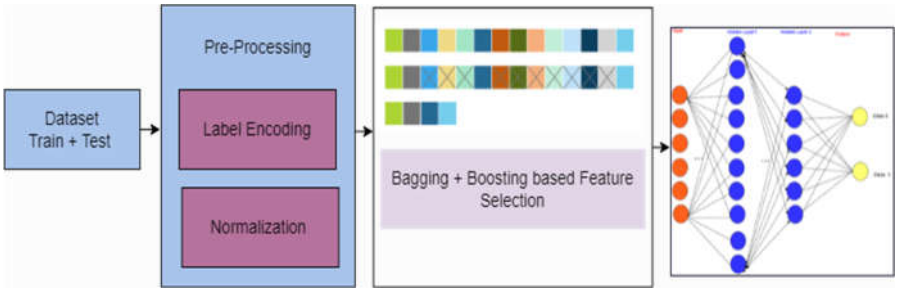

Figure 2 Outline of the proposed framework.

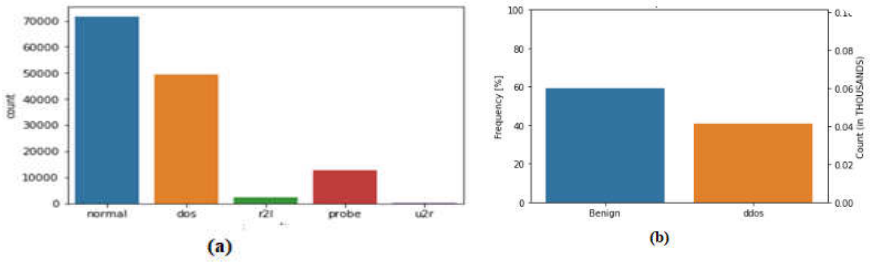

Figure 3 (a) Samples distribution from KDDCUP99, (b) sample distribution from CICDDoS.

3.2 Data Pre-processing

The investigation started with data preparation, or data engineering. Occasionally, feeding the models with raw data can result in a prediction that is false. The datasets do not contain any duplicate or null values, and the data preprocessing stage only includes feature categorical encoding, feature scaling, and feature selection. The experimental datasets KDDCUP99 and CICDDoS consisted of many categorical variables. Since the machine learning process uses only numerical values, the presented framework uses label encoding to convert object type into numeric type. For example the column 'Protocol type' is converted into a number value using label encoding. UDP, TCP and ICMP are

the alternative values in the column. The TCP, UDP and ICMP values are changed into 2, 1, and 0 respectively.

3.2.1 Feature Scaling and Normalization

Some of the features in the datasets have highly erratic magnitudes, ranges, and units. Some algorithms make incorrect predictions as a result of the highly variable characteristics of the datasets. Feature scaling is an essential preprocessing step in many machine learning algorithms to ensure that the range of independent variables (features) is consistent and appropriate for the model. Two commonly used feature scaling techniques are standardization and normalization. The presented framework deploys normalization according to the sample distribution. Data normalization is applied to feature columns before using them to train machine learning models so that the trained models become less sensitive to the scale of the features. Normalization facilitates the model for converging to better weights, whereby the designed model predicts outputs more accurately. The data normalization is accomplished using Eq. (1). For the given vector x with n components, the L2 norm takes the sum of squared value and the square root at the end. In normalization, each element in the vector is squared to produce the sum value of 1.

$$\|x\|_2 = \left(\sum_{i=1}^{N} |x_i|^2\right)^{\frac{1}{2}} \tag{1}$$

3.2.2 Feature Selection

Superfluous and irrelevant features slow down the network intrusion detection process and prevent the classifier from making a correct classification. Correlation feature selection (CFS) yields promising results in obtaining the predominant feature variable to a certain extent. CFS [30] is based on a hypothesis, the features from the set correlate well with the output label and not with each other. The proposed framework uses the CFS tool to detect highly correlated features. The idea is that the initial set of redundant features is removed through CFS. Highly correlated pairs of features are chosen based on a threshold value and either one of them is retained for choosing the output class label. In the experiment, features with a correlation value greater than or equal to 0.8 were removed. This promising step deleted ten redundant features from the KDDCUP99 dataset and fifteen features from the CICDDoS dataset. Removing irrelevant features is essential for neural based classifiers. Then further dominant features can be derived through the XGboost technique, which makes use of a gradient descent algorithm. The core idea of the technique is to learn the current pattern with the previous step's result facilitating performance improvement.

Another benefit of this technique is that it does not require scaling since it depends on trees to capture the complex non-linearity pattern and interactions. Such a technique can handle any type of data and is not affected by redundant independent variables. While designing a classification model, two facts are essential: feature selection and feature transformation; the first step is used to detect the dominant independent features, and the latter step is used to refine the data to gain insight. These two steps are built into the XGboost technique. The technique starts with a set of base learners, which can be represented as F $=\{f_1,f_2,f_3\ldots\ldots\ldots\ldots f_n\}$. The initial model was designed to predict the output variable Y using Eq. (2). For the initial model mentioned in the equation, the residual value is computed using the following expression:

$$Fo(x) = argmin \sum_{i=1}^{n} L(yi, \gamma) \tag{2}$$

$$r_{im=} \delta L(yi, F(xi))/\delta F(xi) \tag{3}$$

In the next step, a new base learner h_m is identified, with the residual obtained using Eq. (2). Each $h_m(x)$ is fit on the gradient obtained at each step by combining the initial model F_0, while the new base learner h_m defines the boosted version of the initial model F_0. The new model F_1 can be defined using the following expression:

$$F1(x) <- F0(x) + h1(x) \tag{4}$$

After including the multiplicative factor Υ_m derived for each terminal node, the boosted model $F_m(x)$ is defined as in Eq. (4). Here, m denotes the number of iterations. Thus the final feature selection model is defined after m iterations as follows:

$$F_m(x) = F_{m-1}(x) + \Upsilon_m h_m(x) \tag{5}$$

Another method deployed by the proposed feature selection component is Random Forest, a bagging method that fits many models of training samples and combines the results. For the given the training set X and responses Y, RF selects a random sample from the training set with B times of replacement and then fits these samples. After training, the prediction of the unseen samples $x^{'}$ is done by calculating the average of the prediction values from all the individual regression trees as follows:

$$\dot{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x') \tag{6}$$

The mathematical expression in Eq. (6) removes the noise pattern and subsets of the features that are selected randomly by splitting each candidate from the resultant set. A classification problem with *n* features uses $\sqrt{n}$ features in each split. Highly correlated features are selected using many B-trees to improve the accuracy. After the randomization of the trees, impurities in the data generated from extremely randomized trees are removed by selecting a random cutting

point rather than computing the optimal cutting point using Gini Index or information gain. From the randomly generated tree splits, the place that produces the highest score is used to make a split node. The experiment was established with the RF model. First it was trained with the 41 features and then the top influential features were found. The bagging method facilitates a decrease of the complexity of models that overfit on the training samples. The proposed feature selection combines the merits of the bagging and boosting ensemble algorithms to detect highly influential features. In contrast, boosting increases the robustness of models that may underfit the training samples. The set of feature columns obtained through the boosting and bagging methods using the KDDCUP99 and CICDDoS datasets are given in Tables 2 and 3.

Table 2 Top twenty-two features from KDDCUP99.

Feature columns
src_bytes , srv_count, Service, Hot , dst_bytes, diff_srv_rate ,dst_host_srv_count, num_file_creations, Count, dst host_srv_diff_host_rate, dst_host_same_src_port_rate wrong_fragment, protocol_type, root_shell, Duration, rerror_rate, Flag, srv_diff_host_rate, dst_host_diff_srv_rate, logged_in, dst_host_count, same_srv_rate

Table 3 Dominant features from CICDDoS dataset.

Feature columns
Flow ID, Src IP, Dst IP, Fwd Seg Size Min, Init Bwd Win Byts Init Fwd Win Byts, Timestamp, Fwd Seg Size Avg, Fwd Pkt Len Mean Tot Fwd Pkts

3.3 Classification

The proposed deep neural network model performs classification in supervised mode. The model transforms the input features into output classes, which can be expressed as follows:

$$f(.) = R^m \rightarrow R^n \tag{7}$$

where m is the number of dimensions in the input and n is the number of dimensions in the output. The deep neural network models learn non-linear approximation to transform the set of input features X = x_1,x_2,x_3…..x_m to o_1,o_2…o_n. Figure 4 shows the proposed optimized deep neural network architecture. The proposed framework has three hidden layers and one input layer. The hidden layer equation for nonlinear transformation is given by Eq. (8) and the output class vector classification is computed using Eq. (9). The proposed attack classification is depicted as pseudo-code as below.

$$h_j = f(\sum_i w_{ij} \ . X_i) \tag{8}$$

where weights W_{ij} are assigned to hidden layer j with weights assigned to hidden neuron i, and X_i represents input vector i = 1, 2, …n and h_j refers to the j^{th} hidden layer, where j varies from 1 to n.

$$Y_k = f\left(\sum_j w_{jk} . h_j\right) \tag{9}$$

The output layer neuron is the dot product of hidden layer output and hidden layer weights, where weights W_{ik} are assigned to hidden layer j with weights assigned to hidden neuron k, and h_j represents the output of the j^{th} hidden layer, where j varies from 1 to n.

4 Results and Discussion

This section discusses the efficacy metrics used to measure the reliability of the proposed system. To prove the effectiveness of the proposed NIDS framework, the experimented results were compared with its counterpart NIDS presented in various previous studies.

4.1 Efficacy Metrics

Most of the researchers use the below-mentioned metrics to evaluate the performance of NIDS models. Recall (R), precision (P), and F1-score. The true positive rate (TPR) represents the ability of the classifier to recognize all positive cases and is calculated using Eq. (10):

$$\text{Recall}_{\text{Rec}}® = \frac{\text{TP}}{\text{TP+FN}} \tag{10}$$

A true positive (TP) occurs when a positive example is classified accurately. A false negative (FN) occurs when a negative example is misclassified. Precision (P) assesses the capacity of the classifier to avoid positive instance misclassification and can be represented as in Eq.(11). The expression for computing the accuracy of the proposed model can be found in Eq. (12).

$$Precision_{Pre} = \frac{TP}{TP+FP} \tag{11}$$

$$Accuracy_{AC} = \frac{TP+TN}{TP+TN+FP+FN} \tag{12}$$

where FP represents how often negative examples were classified as positive. The F1 score is the harmonic mean of the past two qualities, as portrayed in Equation 10. A high F1 score means excellent performance of the classifier.

$$F1Score = 2\frac{Precision.Recall}{Precision+Recall} \tag{13}$$

4.2 Experimental Setup

Utilizing the Keras library in Python 3.6, a deep neural network classifier was crafted for addressing high-dimensional problems. The model incorporates a precise configuration with an input layer of 1,000 neurons, accommodating 41 and 22 feature vectors, and three hidden layers capturing variations in feature instances. The output layer is created for multiclass and binary labels, employing the rectified linear unit (ReLU) activation function in the input layer for enhanced computation unit performance. The ReLU activation function can be presented in simple form as in Eq. (14), where x represents the input features and f(x) represents the ReLU activation function.

$$f(x) = x^{+} = \max(0, x) \tag{14}$$

The hidden layer of the data is activated through the SoftMax function, which turns the data values into probabilities that sum up to unity. In general, the input for the function is a vector of real numbers and uses a normalization technique that consists of *K* probabilities to map the values into a probability distribution. The probabilities for vector *K* can be calculated using the standard SoftMax function. Each input element Z_j of the vector *Z* uses standard exponentiation. Then, the values are normalized by dividing them by the sum of the exponentials of each data set in the vector. The normalization confirms that the sum of the values is always calculated to 1 using Eq. (15), where σ denotes the SoftMax function, Z denotes the input vector, e^{z_j} represents the standard exponential function for the input vector, *K* denotes the number of classes in the multi-class classifier, and e^{z_k} represents the standard exponential function for the output vector.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \; for \; j = 1, \ldots K \tag{15}$$

The designed model is compiled with an efficient optimizer to search through numerous weighted networks and any other metrics that need to be collected during the training of the model. The proposed experiment uses the Adam optimizer, a first-order gradient-based optimization algorithm. The reason for choosing the Adam optimizer is that it is straightforward, efficient, and well-suited for large volumes of data, and has minimum memory requirements. The established model was tested with the feature columns derived through the proposed feature selection package and the whole set of features came with the dataset. The neural network model was set up for binary as well as multiclass classification attack detection. In the next step, the designed model was fit with 150 of epochs (KDDCUP99). Figure 4 shows the iteration versus accuracy relationship for 41 and 22 feature columns. As the model is trained with a varying number of iterations it learns the patterns gradually until it reaches the optimal accuracy value. At the 150th epoch, the proposed model achieved 99.5 as training

accuracy, which is highly competitive in detecting cyber threat vulnerabilities with 41 features (see Figure 4(a)). And after deploying the presented feature selection package, the model was trained with 22 unique features. The model accuracy plot for this case can be seen in Figure 4(b), which shows that the testing accuracy was 99.4, which is highly commendable.

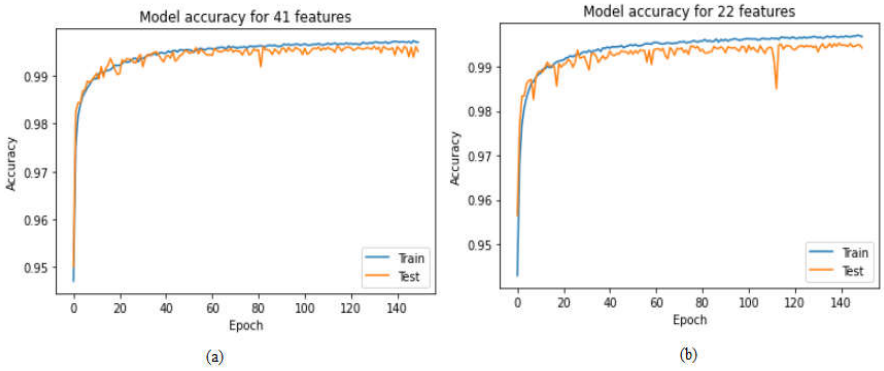

Figure 4 Accuracy versus number of epochs for 41 and 22 feature columns (KDDCUP99).

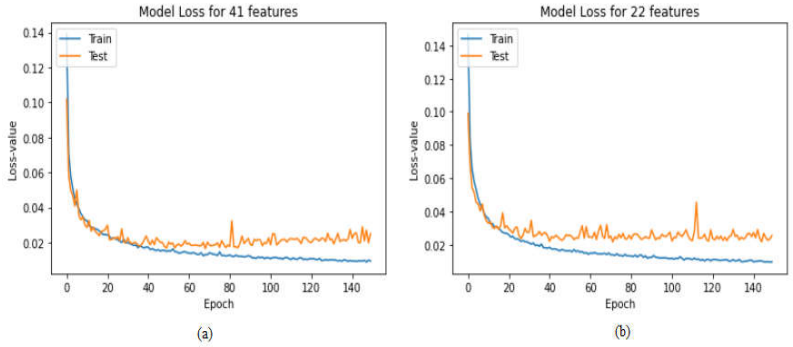

Figure 5 Model loss versus the number of epochs for 41 and 22 feature columns (KDDCUP99).

Figures 4 and 5 show the designed multiclass deep neural attack detection model's learning path with and without the presence of the proposed feature selection package. The deviation between training and testing should be minimal to ensure better learning. Figure 5 illustrates the relationship between model loss values versus the number of epochs for the 41 and 22 features columns. Figure 5 shows that as the number of epochs increases the loss values decreases. The

plotting curve reveals that with 22 features the proposed neural network model performed well. The model with 22 features is comparable to the model with 41 features in terms of accuracy. Then, a binary deep neural network attack detection model was designed and tested with 150 epochs; the results in terms of accuracy and loss were in line with the multiclass attack detection model. The model training details are plotted in Figure 6 for the reader's perusal. The proposed deep neural classifier is theoretically very complex and requires good expertise to fine-tune the setup. An improper setup may end up with either overfitting or underfitting.

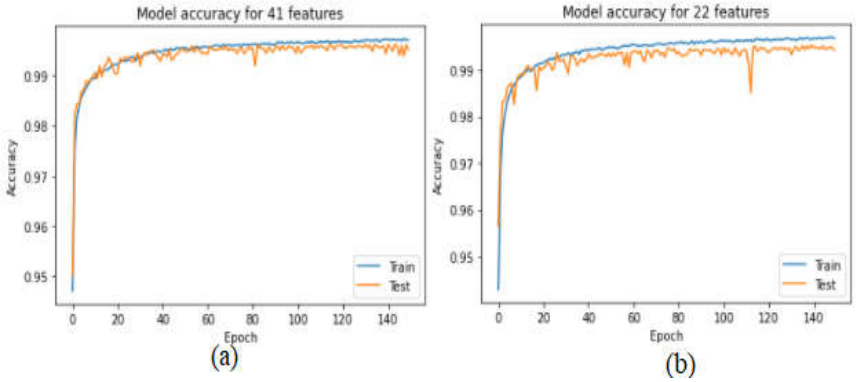

Figure 6 Model accuracy versus number of epochs curve for 41 and 22 features (Binay).

To feed the proposed NIDS framework, the CICDDoS dataset was also used to prove its resilience against attacks. The learning curves generated by the DNN model that were built on the features (as mentioned in Table 3) selected by the proposed ensemble feature selection technique can be found in Figures 7 and 8. As the number of epochs increased, both training and testing began to converge and finally at 100 epochs both the training and the testing accuracy were almost the same. Hence, the curves rendered in Figure 7(a) show that the DNN model learned from the data accurately and could successfully be deployed to mitigate cyber-attacks. The same kind of outcome can be seen in Figure 7(b), i.e., as the number of epochs increased, the loss value started to decrease. One more notable thing about the curves displayed in Figure 7 is that the experimental model was built upon unbalanced samples.

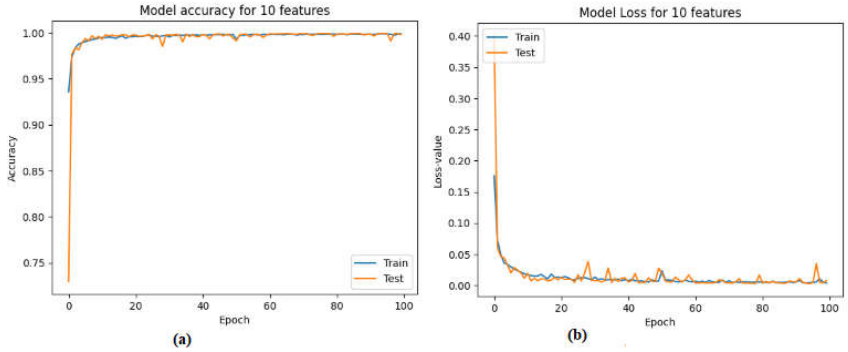

Figure 7 DNN model training curve on CICDDoS dataset (unbalanced samples).

In another experiment, the samples were balanced against two output class labels. In this context, the model's training accuracy and testing accuracy curves displayed in Figure 8 also converged at a distinct point of time. It is expected that the deviation between both curves (training and testing) should be as minimal as possible. According to this fact, the results provided in Figures 4 to 8 prove the model's learning ability, i.e., the deviation is minimal.

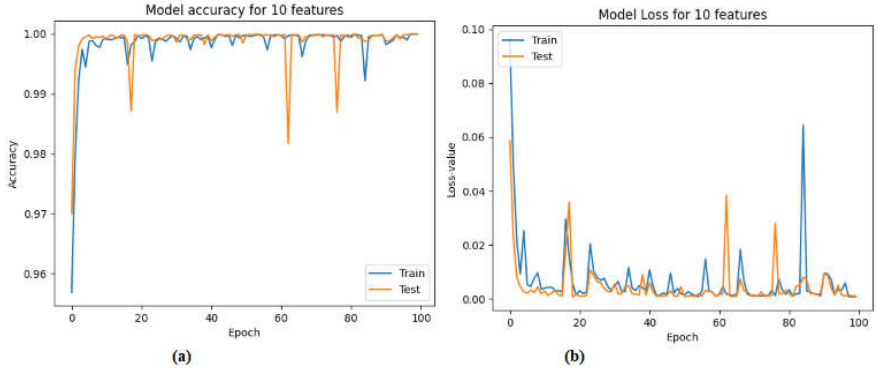

Figure 8 DNN Model training curve on CICDDoS dataset (balanced samples).

4.3 Discussion on Results

This section discusses the performance of the proposed NIDS (P-NIDS) tested on the KDDCUP99 and CICDDoS datasets against quantitative metrics. To identify the results obtained through the KDDCUP99 and CICDDoS datasets, the following notation was used: P-NIDS-KDD, P-NIDS-CIC. The test results of the

proposed NIDS framework in binary and multiclass mode along with the results obtained from some of the states of the NIDS techniques can be found in Table 4. The results in [36] were obtained using an ANN model, which uses the same network structure for both binary and multi-class attack detection. The framework presented in [38] is an NIDS built on the UNSW-NB15 dataset and the provided results were obtained for six features.

Table 4 Tested results (binary, multiclass) through neural-based models.

(a) NIDS (Binary)

Method	AC	Pre	Rec	F1-score
ANN [21]	0.92	0.93	0.92	0.92
ANN [36]	0.92	0.99	0.91	0.95
DNN4 [37]	0.93	0.99	0.91	0.95
DNN5 [37]	0.92	0.99	0.91	0.95
FE-DNN [38]	0.87	0.87	0.96	0.91
P-NIDS-KDD	0.994	0.994	0.994	0.994
P-NIDS-CIC	0.998	0.998	0.998	0.998

(b) NIDS (Multi-Class)

Method	AC	Pre	Rec	F1-score
DNN2 [36]	0.926	0.944	0.926	0.920
DNN3 [36]	0.93	0.920	0.935	0.925
DNN4 [36]	0.929	0.911	0.929	0.918
DNN5 [36]	0.925	0.934	0.925	0.921
P-NIDS-KDD	0.994	0.994	0.994	0.994

The presented feature selection component had twenty-two predominant features in the KDDCUP99 dataset and ten in the CICDDoS dataset. The authors in [36] used a hierarchical cluster approach for selecting the dominant features, after which the selected features were used to enhance the performance of the ANN (artificial neural network). Table 4(a) reports the results for the NIDS built in binary mode. Based on analysis of various neural-based NIDS frameworks presented in different previous studies, the proposed model built on the KDDCUP99 and CICDDoS datasets showed promising results in terms of accuracy (AC), precision (Pre), recall (Rec), and F1-score. The numeric data presented in Table 4(a) is displayed in Figure 9. Table 4(b) presents the attack multiclassification framework's results. The proposed technique recorded the best performance compared to the others.

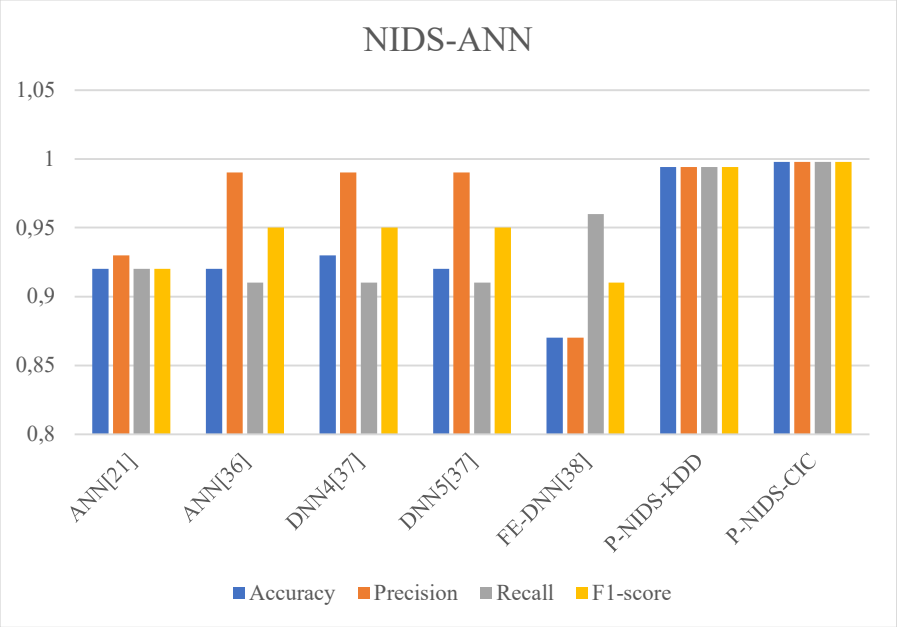

Figure 9 Neural-based NIDS results.

Table 5 Test results with existing ML-based NIDS (binary).

(a)

Method	AC	Pre	Rec	F1-score
SVM [21]	0.92	0.92	0.92	0.91
NB-SVM [24]	0.93	0.93	0.94	0.92
GI-DT [26]	0.999	0.999	0.999	0.999
KNN [36]	0.925	0.998	0.909	0.952
AB [36]	0.925	0.996	0.910	0.951
RF [36]	0.927	0.999	0.911	0.953
P-NIDS-KDD	0.994	0.994	0.994	0.994
P-NIDS-CIC	0.998	0.998	0.998	0.998

(b)

Method	AUROC (%)
SVM [8]	96.80
Chi_SVM [8]	99.5
LR [8]	92.70
P-NIDS-KDD	99.4

Table 5 illustrates the performance of a network intrusion detection system (NIDS) built on various machine learning techniques, particularly focusing on

the KDDCUP99 and CICDDoS datasets. Figure 10 presents the result from Table 5. It showcases superiority over the benchmark NIDS techniques, except for the Gini Impurity-based Random Forest Decision Tree model for attack classification.

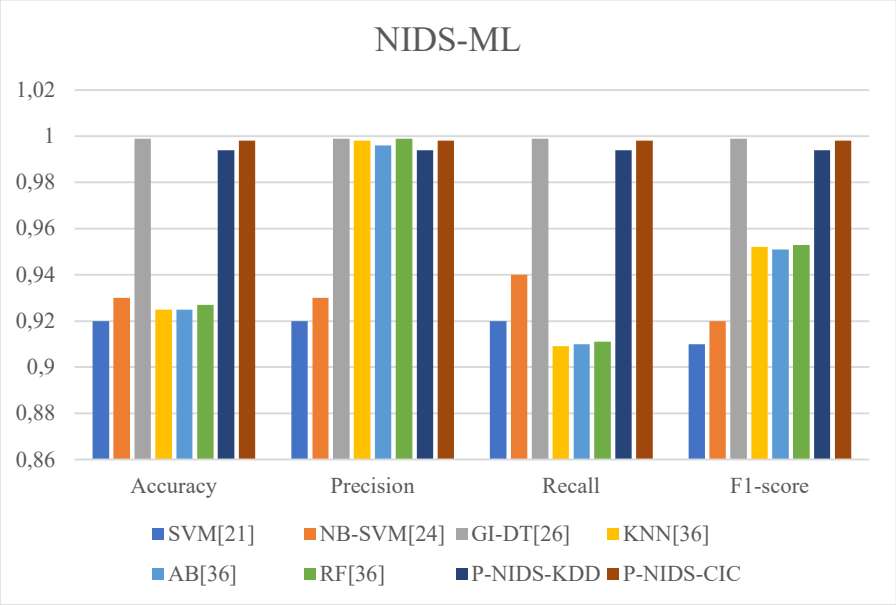

Figure 10 Machine learning-based NIDS.

Figure 11 shows the confusion matrix; the metrics reveal the effectiveness of the presented NIDS, especially when employing balanced samples, yielding 99.9% accuracy compared to 99.8% with unbalanced samples. In contrast to the pigeon-inspired optimization-based model, which achieved 94% accuracy, the proposed method attained 99.8% accuracy with unbalanced samples. Additionally, the presented technique outperformed the others in terms of the AUROC metric, emphasizing its suitability for countering cyber-attacks in IoT networks through ensemble feature selection and an optimized deep neural network classifier. Table 5(b) reports some of the results from previous studies for comparison. In terms of the AUROC metric, the of the proposed technique were better than those of the others. By comparing the frameworks published recently, the proposed framework indicated ensemble feature selection followed by the optimized deep neural network classifier as the most appropriate solution to mitigate cyber-attacks anticipated in IoT networks.

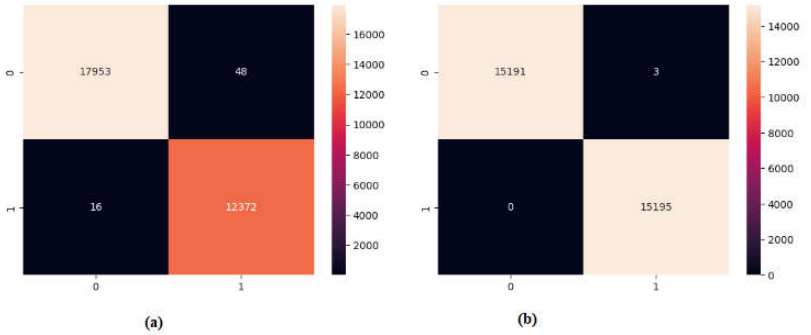

Figure 11 (a) Classification results for unbalanced samples; (b) results for balanced samples (CICDDoS).

5 Conclusion and Future Work

The present research developed an intelligent NIDS system to defend against potential assaults anticipated in IoT and wireless networks. The NIDS has an ensemble feature selector that makes use of the advantages of the boosting and bagging techniques. A refined deep neural network attack detection model is included in the NIDS framework's second phase in order to speed up the mitigation procedure. Two datasets, KDDCUP99 and CICDDoS, were used in the experiment to see if the model could be made more resilient. Comparing the quantitative metric findings with those produced using state-of-the-art techniques demonstrated how well the proposed framework performed. On KDDCUP99, the proposed approach achieved a classification accuracy of 99.4%. However, the output class labels for the KDDCUP99 dataset are unbalanced and the dataset itself contains some inherent ambiguities. Therefore, the CICDDoS dataset was used in experiments to improve its suitability for dynamic DDoS attack detection.

The proposed structure achieved 99.9% accuracy in that endeavour. In a future study, it will be determined whether the proposed methodology can be used to mitigate cyber-attacks against bitcoin networks. To be more precise, additional characteristics peculiar to the blockchain platform will be included in order to lower the misclassification rate.

References

[1] Ahmed, M., Mahmood, AN. & Hu, J., *A Survey of Network Anomaly Detection Techniques,* Journal of Network and Computer Applications, **60**(1), pp. 19-31, Jan. 2016.

[2] Alazab, J., Abawajy M., Hobbs, R., Layton & Khraisat, A., *Crime Toolkits: The Productisation of Cybercrime*, IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 1626-1632, 2013.

[3] Lastovicka, M. & Celeda, P., *Situational Awareness: Detecting Critical Dependencies and Devices in a Network*, In IFIP International Conference on Autonomous Infrastructure, Management and Security, pp. 173-178, 2017.

[4] Gong, Y., Mabu, S., Chen C., Wang Y. & Hirasawa, K., *Intrusion Detection System Combining Misuse Detection and Anomaly Detection using Genetic Network Programming*, in 2009 ICCAS-SICE IEEE, pp. 3463-3467, 2019.

[5] Hall, J., Barbeau, M. & Kranakis, E., *Anomaly-Based Intrusion Detection using Mobility Profiles of Public Transportation Users*, IEEE International Conference on Wireless and Mobile Computing, Networking and Communications IEEE, pp. 17-24, 2005.

[6] Lee, J., Moskovic, S. & Silacci, L., *A Survey of Intrusion Detection Analysis Methods*, University of California, San Diego, pp. 1-9, 1999.

[7] Ektefa, M., Memar, S., Sidi, F. & Affendey, L.S., *Intrusion Detection Using Data Mining Techniques,* International Conference on Information Retrieval & Knowledge Management (CAMP), pp. 200-203, 2010.

[8] Othman S.M., Ba-Alwi, F.M., Alsohybe, N.T. & Al-Hashida A.Y., *Intrusion Detection Model Using Machine Learning Algorithm on Big Data Environment,* Journal of Big Data. **5**(1), Pp.1-12, Dec 2018.

[9] Althubiti, S.A., Jones, E.M. & Roy, K., *LSTM For Anomaly-Based Network Intrusion Detection,* in 28th International Telecommunication Networks and Applications Conference (ITNAC), pp. 1-3, 2018.

[10] Buczak, A.L. & Guven, E., *A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,* IEEE Communications Surveys & Tutorials, **18**(2), pp. 1153-1176, 2015

[11] Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C. & Faruki, P., *Network Intrusion Detection for IoT Security Based on Learning Techniques*, IEEE Communications Surveys & Tutorials, **21**(3), pp. 2671-2701, 2019.

[12] Choudhary, G., Sharma, V., You, I., Yim, K., Chen, R. & Cho, J.H., *Intrusion Detection Systems for Networked Unmanned Aerial Vehicles: A Survey*, 14th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 560-565, 2018.

[13] Sharma, V., You, I., Andersson, K., Palmieri, F., Rehmani, M.H. & Lim, J., *Security, Privacy and Trust for Smart Mobile-Internet of Things (M-IoT): A Survey*, IEEE Access, **8** (1), pp. 167123-167163, Sep. 2020.

[14] Kumar, P., Kumar, R., Gupta, G.P. & Tripathi, R., *A Distributed Framework for Detecting DDoS Attacks in Smart Contract-Based*

Blockchain-Iot Systems by Leveraging Fog Computing, Transactions on Emerging Telecommunications Technologies, **32**(6), pp.1-12, Jun. 2020.

[15] Deep, S., Zheng, X., Jolfaei, A., Yu, D., Ostovari, P. & Kashif Bashir, A., *A Survey of Security and Privacy Issues in the Internet of Things from the Layered Context*, Transactions on Emerging Telecommunications Technologies, pp. 1-22, Feb, 2020.

[16] Shafiq, M., Tian, Z., Bashir, A.K., Du, X. & Guizani, M., *Corrauc: A Malicious Bot-IoT Traffic Detection Method in IoT Network Using Machine Learning Techniques*, IEEE Internet of Things Journal, **8**(5), pp.3242-3254, Jun. 2020.

[17] Shafiq, M., Tian, Z., Bashir, A.K., Du, X. & Guizani, M., *IoT Malicious Traffic Identification Using Wrapper-Based Feature Selection Mechanisms*, Computers & Security, **94**(1), pp. 1-22, Jul. 2020.

[18] Seong, T.B., Ponnusamy, V., Jhanjhi, N.Z., Annur, R. & Talib, M.N., *A Comparative Analysis on Traditional Wired Datasets and the Need for Wireless Datasets for IoT Wireless Intrusion Detection,* Indonesian Journal of Electrical Engineering and Computer Science, **22**(2), 1165-1176, 2021

[19] Vigneswaran, K.R., Vinayakumar, R., Soman, K.P., Poornachandran P., *Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security,* 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-6, 2018.

[20] Alazzam, H., Sharieh, A. & Sabri, K.E., *A Feature Selection Algorithm for Intrusion Detection System Based on Pigeon Inspired Optimizer,* Expert Systems with Applications, **148**, 113249, 2020.

[21] Aboueata, N., Alrasbi, S., Erbad, A., Kassler, A. & Bhamare, D., *Supervised Machine Learning Techniques for Efficient Network Intrusion Detection*, in 2019 28th International Conference on Computer Communication and Networks (ICCCN), pp. 1-8, IEEE, July 2019

[22] Meftah, S., Rachidi, T. & Assem, N., *Network Based Intrusion Detection using The UNSW-NB15 Dataset,* International Journal of Computing and Digital Systems, 8(5), pp. 478-487, 2019.

[23] Injadat, M., Moubayed, A., Nassif, A.B. & Shami, A., *Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection,* IEEE Transactions on Network and Service Management, **18**(2), pp. 1803-1816, 2020.

[24] Gu, J.& Lu, S., *An Effective Intrusion Detection Approach using SVM with Naïve Bayes Feature Embedding*. Computers & Security, **103**, 102158. 2021.

[25] Moustafa, N., *A New Distributed Architecture for Evaluating AI-Based Security Systems at the Edge: Network TON_IoT Datasets.* Sustainable Cities and Society, **72**, 102994, 2021.

[26] Disha, R.A. & Waheed, S., *Performance Analysis of Machine Learning Models for Intrusion Detection System Using Gini Impurity-Based Weighted Random Forest (GIWRF) Feature Selection Technique,* Cybersecurity, **5**(1), 1, 2022.

[27] Vigneswaran, K.R., Vinayakumar, R., Soman, K.P. & Poornachandran P., *Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security,* 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-6, 2018.

[28] Gautam, R.K.S. & Doegar. E.A., *An Ensemble Approach for Intrusion Detection System Using Machine Learning Algorithms*, 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pp. 14-15, 2018.

[29] Adeyemo, V.E., Abdullah, A., Jhan Jhi, N.Z., Supramaniam, M. & Balogun, A.O., *Ensemble and Deep-Learning Methods for Two-Class and Multi-Attack Anomaly Intrusion Detection: An Empirical Study,* International Journal of Advanced Computer Science and Applications, **10**(9), pp.520-528, 2019.

[30] Kasongo, S.M. & Sun, Y., *A Deep Long Short-Term Memory Based Classifier for Wireless Intrusion Detection System,* ICT Express, **6**(2), pp. 98-103, Jun. 2020.

[31] Vinayakumar, R., Soman, K.P & Poornachandran, P., *Applying Convolutional Neural Network for Network Intrusion Detection,* International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1222-1228, 2017.

[32] Yin, H., Xue, M., Xiao, Y., Xia, K. & Yu, G., *Intrusion Detection Classification Model on an Improved k-Dependence Bayesian Network*, IEEE Access, **7**(1), pp. 157555-157563, Oct. 2019.

[33] Song, H. M., Woo, J. & Kim, H.K., *In-Vehicle Network Intrusion Detection Using Deep Convolutional Neural Network*, Vehicular Communications, 21, pp. 100198- 100210, Jan. 2020.

[34] 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, **32**(1), pp. 1-29, Jan. 2020

[35] Sapr, S., Ahmadi, P. & Islam, K., *A Robust Comparison of the KDD Cup99 and NSL-KDD IoT Network Intrusion Detection Datasets Through Various Machine Learning Algorithms*, arXiv Preprint arXiv:1912.13204, pp. 1-8, Dec. 2019.

[36] Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., Al-Nemrat A. & Venkatraman, S., *Deep Learning Approach for Intelligent Intrusion Detection System,* IEEE Access, **7**, pp. 41525-41550, Apr. 2019.

[37] Zhang, L., Liu, K., Xie, X., Bai, W., Wu, B. & Dong, P., *A Data-Driven Network Intrusion Detection System Using Feature Selection and Deep Learning,* Journal of Information Security and Applications, **78**, 103606, 2023.

[38] Man, D., Zeng, F., Yang, W., Yu, M., Lv, J. & Wang, Y. *Intelligent Intrusion Detection Based on Federated Learning for Edge-Assisted Internet of Things,* Security and Communication Networks, 1-11, 2021.

[39] Hamdi, N., *Federated Learning-Based Intrusion Detection System for Internet of Things*, International Journal of Information Security, **22**(6), 1937-1948, 2023.

[40] Bhattacharya, S., Kaluri, R., Singh, S., Alazab, M. & Tariq, U., *A Novel PCA-Firefly Based XGBoost Classification Model for Intrusion Detection in Networks Using GPU*, Electronics, **9**(2), pp. 1-16, Feb. 2020.

[41] Alazzam, H., Sharieh, A. & Sabri, K.E., *A Feature Selection Algorithm for Intrusion Detection System Based on Pigeon Inspired Optimizer*, Expert Systems with Applications, **148**, pp. 1-14, Jun. 2020.

[42] Thaseen, I.S., Chitturi, A.K., Al-Turjman, F., Shankar, A., Ghalib, M.R. & Abhishek, K., *An Intelligent Ensemble of Long-Short-Term Memory with Genetic Algorithm for Network Anomaly Identification*, Transactions on Emerging Telecommunications Technologies, pp. 1-21, Oct. 2020.

[43] *KDD Cup 1999 Data, Intrusion Detection*, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (6 June 2021).

[44] *DDoS Dataset*, https://www.kaggle.com/datasets/devendra416/ddos-datasets (11 Sep 2023)