Journal of Cloud Computing

RESEARCH Open Access

Enhancement of an IoT hybrid intrusion detection system based on fog-to-cloud computing

Doaa Mohamed[1] and Osama Ismael[1*]

Abstract

Nowadays, with the proliferation of internet of things-connected devices, the scope of cyber-attacks on the internet of things has grown exponentially. So, it makes it a necessity to develop an efficient and accurate intrusion detection system that should be fast, dynamic, and scalable in an internet of things environment. On the other hand, Fog computing is a decentralized platform that extends Cloud computing to deal with the inherent issues of the Cloud computing. As well, maintaining a high level of security is critical in order to ensure secure and reliable communication between Fog nodes and internet of things devices. To address this issue, we present an intrusion detection method based on artificial neural networks and genetic algorithms to efficiently detect various types of network intrusions on local Fog nodes. Through this approach, we applied genetic algorithms to optimize the interconnecting weights of the network and the biases associated with each neuron. Therefore, it can quickly and effectively establish a back-propagation neural network model. Moreover, the distributed architecture of fog computing enables the distribution of the intrusion detection system over local Fog nodes with a centralized Cloud, which achieves faster attack detection than the Cloud intrusion detection mechanism. A set of experiments were conducted on the Raspberry Pi4 as a Fog node, based on the UNSW-NB15 and ToN_IoT data sets for binary-class classification, which showed that the optimized weights and biases achieved better performance than those who used the neural network without optimization. The optimized model showed interoperability, flexibility, and scalability. Furthermore, achieving a higher intrusion detection rate through decreasing the neural network error rate and increasing the true positive rate is also possible. According to the experiments, the suggested approach produces better outcomes in terms of detection accuracy and processing time. In this case, the proposed approach achieved an 16.35% and 37.07% reduction in execution time for both data sets, respectively, compared to other state-of-the-art methods, which enhanced the acceleration of the convergence process and saved processing power.

Keywords Fog computing, Internet of things, Genetic algorithm, Intrusion detection, Neural network, Standard deviation

*Correspondence:
Osama Ismael
osama@fci-cu.edu.eg
[1]Department of Information Systems, Faculty of Computers and Artificial Intelligence, Cairo University, Giza, Egypt

Introduction

Nowadays, many companies and start-ups use Cloud computing because it's a cost-effective way of establishing and operating their own system resources. In addition to location awareness, there are some issues facing Cloud computing, such as low latency, geo-location and mobility support. In contrast, Fog computing technology is a simplified version of the Cloud computing paradigm that enables a wide range of applications and services with low latency, improved location service awareness, mobility, and flexibility [28]. Fog computing is a platform similar to Cloud computing that provides end users with processing, storage, and application services, as well as network services between Fog nodes and Cloud servers [29]. Due to its deployment in different locations where security is limited, Fog computing faces many security and safety issues. Fog devices, for example, are vulnerable to a variety of cyber-attacks, such as man-in-the-middle and port scan attacks, which violate the privacy of their data [14].

In addition, Fog computing is a distributed platform that can operate and process data at a regional level and may be deployed on a wide range of devices, making it ideal for Internet of Things (IoT) applications. The IoT has emerged as an industrial revolution in the last decade because of the popularity and usage of low-cost, energy-efficient devices like sensors, actuators, etc., combined with a variety of communication mediums. IoT can be considered as one of the most recent and widely used computing paradigms that have emerged to enhance data collection and introduce new types of data services [21]. It's worth mentioning that the internet of things exists in many aspects of our lives, with high challenges in security.

On the other hand, to assure service quality, intrusion detection systems (IDS) can be considered as an essential component of any security mechanism for Fog and IoT networks. As a result, IDS have become an indispensable part of Fog computing and the IoT to ensure the quality of service. IDS fall into two major categories: misuse detection and anomaly detection. Misuse detection, also known as signature-based, detects only known attacks but fails to detect newly created attacks because it mostly depends on the rules that the network administrator has set. The second category, anomaly detection, can solve the problem of the first because it can detect new and unknown attacks because it depends on a statistical approach. However, most of the IDS available use only misuse detection because most developed anomaly detectors face many challenges, such as generating a high rate of false positive alarms and expensive computation. So in order to build an intrusion detection system, we must first understand how an attack occurs, collect information, configure remote to local access, and launch an attack [3].

Intrusion detection systems use a variety of methods, but none of them are completely without flaws. There are two types of intrusion detection system software: network-based intrusion detection systems and host-based intrusion detection systems. By monitoring network traffic, network-based IDS attempt to detect malicious activities such as denial-of-service attacks and port scans. While the host-based IDS looks at actions and files on the host devices.

As mentioned earlier, the Internet of Things is vulnerable to a variety of attacks, including both internal and external intrusions [4]. Therefore, an effective intrusion detection system is highly desirable to deal with inappropriate use of computers and IoT devices that violate security and administrative policies. As a result, in this study, we propose a new methodology for improving the intrusion detection system for combating IoT attacks deployed over the Fog network. This was achieved through using a hybrid approach that consists of two algorithms, Back Propagation Neural Networks (BPNN) and Genetic Algorithms (GA) [31], to efficiently detect various types of network intrusions. Through this approach, we propose the optimization of the weights and biases in neural networks using Genetic algorithms to enhance the neural network performance. Therefore, it can quickly and effectively establish a back-propagation neural network model. Moreover, we will focus on the network-based IDS because it observes all data passing through the network before passing it to hosts inside the network.

The key contributions of this research can be summarized in the following points:

- Present a novel approach that uses the Genetic algorithms to optimize the Artificial Neural Network parameters before deploying it on the Fog network. This can aid in establishing a rapid and effective BPNN model that can be uploaded to the Fog network for anomaly classification purposes.
- The optimized model has a greater intrusion detection rate since the neural network error rate is lower and the true positive rate is higher.
- Achieving up to a 37.07% decrease in execution time when compared to other cutting-edge approaches, which speeds up the convergence process and saves computing power.

Related work

In recent years, many IDS have been proposed in the literature, which are used to monitor IoT-based networks against various attacks. However, IDS frequently experience poor detection accuracy because of composed attacks like DDoS and a lack of methodology through which the attack data can be exchanged between the

network nodes. Much research has been conducted to improve IDS performance in a variety of ways, including using neural networks and multi-layer perception [22, 28], combining multiple machine learning algorithms [1, 2, 12, 25], detecting Distributed Denial of Service (DDoS) and Denial of Service (DoS) attacks [5, 11], and other enhancements [8–10, 19, 24, 26].

Sudqi et al., [28] introduced a multilayer perceptron (MLP) model for intrusion detection. They used two different datasets to evaluate their system; the Australian Defense Force Academy Linux Dataset and the Australian Defense Force Academy Windows Dataset. They reduced the number of features by applying mutual information feature selection and a modified vector space representation via n-gram transformation. Also, they used the Raspberry Pi as a Fog device, in which the electrical current demand and the voltage were used to determine the Raspberry Pi's power consumption in the experiment's design.

Pacheco et al., [22] proposed an artificial neural network-based Anomalous Behavior Analysis (ABA-IDS) approach to create an adaptive intrusion detection system capable of recognizing when a Fog node has been hacked and then taking the appropriate precautions to ensure communication availability. Their approach comprises creating a node's profile based on the node's attributes, which is then passed into artificial neural networks that are built to characterize the node's regular functions.

Alghayadh and Debnath [2] proposed a hybrid intrusion detection system using multiple machine learning algorithms, including random forest, Xgboost, decision tree, K-nearest neighbors, and misuse detection techniques designed for smart homes based on user behavior profile patterns. They used the CSE-CIC-IDS2018 and NSL-KDD datasets to verify their model. Also, Kalaivani and Chinnadurai [12] proposed a multi-class attack classification model based on deep learning methods, called ICNN-FCID. They integrated the convolutional neural network (CNN) and the Long Short-Term Memory networks (LSTM) algorithms in order to predict the network attacks in the Fog computing layer. In addition, they used the NSL-KDD as a benchmark dataset.

Abbas et al., [1] proposed an ensemble-based intrusion detection model. The architecture of the proposed model consists of three machine learning algorithms instead of using artificial neural networks and deep learning techniques, which are logistic regression, naive Bayes, and decision trees. The proposed model has been evaluated and analyzed using the CICIDS 2017 dataset in both binary and multi-class contexts. Also, Ravi et al., [25] proposed using deep learning-based recurrent models to build a system for cyber–physical network attack detection and classification. The proposed model uses a kernel-based principal component analysis technique to select features and extract the hidden layer features of recurrent models. The model employs random forest and support vector machine on the extracted features, followed by logistic regression for network attack detection and classification. They used more than one benchmark network intrusion dataset to evaluate the performance of the proposed method, including the SDN-IoT, KDD-Cup-1999, UNSW-NB15, WSN-DS, and CICIDS-2017 datasets.

An et al., [5] introduced a hypergraph clustering model based on the Apriori algorithm to analyze the association between Fog nodes that are suffering from the threat of DDoS. Also, Jan et al., [11] introduced an attack detection strategy based on the Support Vector Machine (SVM) algorithm, with input given in the form of two or three features, to mitigate the most common DDoS attack in the IoT environment. They used only one attribute to classify the signal, which is the packet arrival rate at the node. Furthermore, they conducted a comparison of SVM-based classifiers with other machine learning-based classifiers, including Neural Network, k-Nearest Neighbors, and Decision Tree, to prove their view of the advantage of utilizing SVM over other techniques.

Illy et al., [9] described their proposed solution as combining multiple learners using different algorithms (i.e., KNN, Random Forest, Bagging and Boosting of decision trees) and building different classifiers for anomaly detection and attack classification. Also, they proposed a deployment architecture where anomaly detection takes place on the Fog node side while attack classification takes place on the Cloud side to guide the intrusion prevention tasks. They test their experiments on KDDTest+and KDDTest-21. Also, Narendra and Rakesh [19] proposed an intrusion detection system on Fog nodes for IoT applications. They applied a Persistent Regularization algorithm using Cholesky Factorization-based Online Sequential Extreme Learning Machines (CF-OSELM-PRFF). In which IoT-generated attacks are detected in local Fog nodes and reported to the cloud server.

Sadaf and Sultana [26] proposed the Auto-IF intrusion detection system approach for real-time intrusion detection in Fog computing environments based on isolation forest and auto-encoder. On the Fog devices, this technique conducts binary classification of incoming packets where the judgement distinguishes intrusions from regular packets. They used the NSL-KDD dataset to validate their method. Also, Rani et al., [24] proposed a network intrusion detection system based on the Random Forest classifier algorithm. They used two different datasets: NSL-KDD and KDD-CUP99, with minimal feature sets. Features are selected manually after analyzing different attacks and their characteristics. They implemented their

method on a synthesized generated network traffic dataset, so it may not be suitable for real network traffic.

Imrana et al., [10] proposed a bidirectional Long-Short-Term-Memory (BiDLSTM)-based intrusion detection system to detect different network intrusion types, especially User-to-Root (U2R) and Remote-to-Local (R2L) attacks. The model's performance has been evaluated and tested using the NSL-KDD benchmark dataset. Also, Houda et al. [8] proposed a framework consists of two phases: the first phase is a Deep Learning-based IoT-related Intrusion Detection System model, and the second phase is an Explainable Artificial Intelligence-based model, whose role is to give explanations about the model's decisions. The proposed framework uses three main Explainable Artificial Intelligence techniques (i.e., RuleFit, Local Interpretable Model-Agnostic Explanations, and SHapley Additive exPlanations), on top of the Deep Learning Neural Network-based Intrusion Detection System model. The proposed framework has been validated based on two different datasets, the NSL-KDD and UNSW-NB15 datasets.

Table 1 shows a summary of the main characteristics of the proposed methodology (EHIDS) compared to some of the state-of-the-art methodologies. As a conclusion, despite the fact that several Artificial Neural Network (ANN)-based intrusion detection methods have been proposed in recent years, and despite the fact that they claim to achieve a high performance rate, the above methodologies still have some shortcomings that must be addressed:

- Several of the previous works included DDoS in Fog computing and ignored other attacks, whereas our proposed model can deal with all types of attacks.
- Most of the methods in the literature work on the Host-based Intrusion Detection System. In contrast, we use Network-based IDS because they have a quicker response time than Host-based IDS. Furthermore, the Network-based IDS do not necessitate any changes to the existing infrastructure and monitor everything on a network segment, regardless of the target host's operating system.
- Most of the introduced methods use traditional feature selection methods (e.g., wrapper methods). Traditional feature selection methods fail to detect several sensitive features, resulting in a classifier that isn't as sensitive as it should be, which leads to incorrect detection. On the contrary, we use filter methods that are much faster, require less space, and have lower complexity.

Hybrid intrusion detection system

Motivated by the above-mentioned problems, we propose a novel hybrid approach called "Enhanced Hybrid Intrusion Detection System (EHIDS)" to be deployed on the Fog network to detect known and newly anomalous intrusions attempting to attack the Fog nodes. To build and utilize the model, three phases are undergone: the pre-processing phase, the optimization phase, and the classification phase. In the first phase, the data is prepared to be fit for the classification process. This phase consists of three steps: feature selection, categorical data

Table 1 Summary of the Main Characteristics of the Proposed Methodology versus other State-of-the-Art Methodologies

The methodology	Algorithms used	Dataset used	Feature selection methodology	Fog devices used	Detection technique
Proposed methodology (EHIDS)	GA and BPNN	UNSW-NB15 and ToN_IoT	Filter methods (Standard Deviation)	Raspberry Pi4	Anomaly
CF-OSELM-PRFF [19]	Cholesky Factorization based Online Sequential Extreme Learning Machines with Persistent Regularization	NSL-KDD	N/A	Azure Cloud Service	Anomaly
ABA-IDS [22]	ANN	Legitimate Commands	Pearson Product-Moment Correlation Coefficient	Raspberry Pi3 model B	Anomaly
ICNN-FCID [12]	CNN and LSTM Networks	NSL-KDD	N/A	N/A	Anomaly
Ensemble-Based IDS [1]	logistic regression, naive Bayes, and decision trees	CICIDS 2017	LinearSVM feature selection	N/A	Anomaly
cyber–physical systems network intrusion detection model [25]	Random Forest, SVM and Logistic Regression	SDN-IoT, KDD-Cup-1999, UNSW-NB15, WSN-DS, and CICIDS-2017	kernel-based principal component analysis technique	N/A	Anomaly
BiDLSTM-based intrusion detection system [10]	bidirectional LSTM	NSL-KDD	N/A	N/A	Anomaly
XAI-based IDS [8]	RuleFit, Local Interpretable Model-Agnostic Explanations, and SHapley Additive exPlanations	NSL-KDD and UNSW-NB15	N/A	N/A	--

conversion to numerical data, and normalisation. In the second phase, we optimize the neural network parameters, namely, weights and biases, through using the Genetic algorithm instead of selecting them randomly. Finally, in the third phase, we build the classification model using BPNN, which is uploaded to the Fog network. Attacks generated by IoT devices are detected by local Fog nodes and reported to the intrusion history repository. In contrast, if the log is classified as normal, it will be passed to the cloud server to be processed. This had a positive impact on the predictability of the IDS. The next three subsections give a detailed description of these three phases.

Pre-processing phase

Both the training and testing datasets require this phase. It is divided into three major steps: feature selection, converting categorical to numerical data, and data normalization. These steps are very important to enhance the quality of the results. The network generates a huge volume of traffic, which slows down the intrusion detection process. For detection purposes, the data often includes some irrelevant and redundant information, so it is important to choose only the relevant information. As a result, feature selection is an essential part of any network intrusion detection system that can effectively identify a subset of the most relevant features in the dataset according to certain criteria in order to improve system performance and reduce computation time. Thus, removing those irrelevant features does not have a negative effect on the system's performance. Moreover, using all of the features increases the system's complexity while decreasing its accuracy. Thus, we aim to select the most suitable subset of features that are relevant to the required task.

As a result, the primary goal of feature selection is to make data mining algorithms more efficient. We can achieve several benefits by reducing the number of features and deleting unnecessary or redundant features, such as speeding up the classification model generation and enhancing system accuracy. There are two methods for feature selection: filter methods and wrapper methods. In the filter methods, features are ranked based on statistical methods, whereas the wrapper methods attempt to use only a subset of features to train the model. Despite the fact that the filter methods may occasionally be unable to identify the best subset of features, the wrapper methods can always identify the best subset of features [23]. But wrapper methods are not the most efficient feature selection tools to be considered in the case of large datasets because they are computationally very costly. Moreover, compared to utilizing the subset of features from the filter methods, using the subset of features from the wrapper techniques makes the model more vulnerable to overfitting. Overall, the particular strengths and weaknesses of each method mean that there is no one best method that is fit for all use cases; it depends on the dataset being used and the specific goals that the methodology aims to accomplish. Therefore, in our case, we recommend using filter methods over wrapper methods because they are much faster and do not require model training [7, 16].

In this research, we suggested using the standard deviation [32] to select the best features that achieved a higher standard deviation. According to the experiments we conducted, as illustrated in the following section, increasing the standard deviation can improve accuracy. Therefore, we set the minimum threshold of the standard deviation at 50. In actuality, a large standard deviation implies that the feature is extended over a wide range of values, whereas a low standard deviation implies that the feature values are extremely near the mean. As a result, choosing features with a higher standard deviation results in a more accurate prediction.

On the other hand, machine learning classifiers require each instance in the input data to be represented as a vector of real numbers. As a result, in the second step, all the categorical values are transformed into numerical form. Also, the class label is transformed into zero or one, for the normal and attack classes, respectively. Last but not least, when we have a huge dataset with thousands of rows in the training and testing files, data set normalization is required. In our proposed model, we normalize data using the min-max normalization method [20]. As a result, every item of data in the training data set has the same scale (e.g., between 0 and 1), significantly reducing the training time.

Optimization phase

During the learning and training of BPNN, there are two main factors that affect the model's performance. The first is the initial neural interconnecting weights and the initial biases associated with each neuron, and the second is their modified values. The initial interconnecting weights and biases of BPNN are often randomly generated, which may cause the classifier model to run into partial optimization and therefore decrease the likelihood of obtaining the best performance and results. Furthermore, since the equations used to adjust the BPNN's interconnecting weights and biases produce very small values, the network's convergence velocity is always slow, and sometimes it does not even converge. These BPNN shortages are required to be addressed, optimized, and improved.

The Genetic algorithm is being used to generate alternative solutions for a given problem and refine them over several generations. Each solution holds all the parameters (i.e., weights and biases) that may assist with the upgrading outcomes. For BPNN, weights in all layers

help achieve high accuracy, and bias is another parameter in the Neural Network that is used to adjust the output along with the weighted sum of the neuron's inputs. In this way, bias is a constant that aids the model in achieving the best fit regarding the given data. In a neural network, each neuron has a bias, the bias is learnable, and the bias also increases the flexibility of the model.

In the search space of the issue being optimised, a chromosome represents a single solution. With reference to our problem, a chromosome is nothing more than a collection of weight and bias values. Thus, the model generates a random population of alternative solutions and then evaluates each one for success, choosing the best chromosome to pass on their "genes" to the next generation, including minor mutations to add diversity. Each chromosome in a population is assessed and ranked based on its relative strength within the population; the goal of the evaluation of the chromosome is to calculate its fitness. The procedure is repeated until the programme produces a workable solution. As a result, the chromosome with the highest fitness level will be chosen and fed to the BPNN. In comparison to our proposed methodology, which uses GA to improve the BPNN by predicting the best weights and biases, there have been other research efforts that attempted to optimise and improve the artificial neural network by using GA over fog computing, but their efforts were limited to predicting the neural network weights only, e.g. [13] and [27].

Classification phase

For this phase, we aim to find a classification model that describes and distinguishes normal and intrusion classes so that we can use it to forecast the class whose label is unknown. Building the classification model is based on analyzing a collection of datasets. This dataset contains both normal and anomalous network traffic, which allows the classifier to determine patterns with a sufficient number of samples. For training and testing the classifier, the dataset is divided into two parts: training and testing, respectively.

In addition, the back-propagation neural network is one of the most common classification methods, and it has been proven to be successful in identifying various forms of intrusions. The BPNN is a network of linked input/output units with a weight assigned to each connection; it also conducts the learning process on a multilayer feed-forward neural network. The topology of the proposed multilayer feed-forward neural network consists of four layers, as follows: an input layer with a number of nodes equal to the number of chosen features; two hidden layers, the first with twenty nodes and the second with ten nodes; and an output layer with only one node. Each node has connections to every neuron in the previous layer. Each connection has a weight that represents how strongly any two nodes are connected, as shown in Fig. 1.

The features chosen for each training tuple correspond to the network's inputs. These inputs are transferred to a second layer (the "hidden layer") after passing through the input layer and being weighted. The outputs of the first hidden layer units represent the inputs to the second hidden layer. Each output unit receives as input a weighted sum of the outputs from the previous hidden layer's units. The hidden and output nodes are mathematical devices that compute the weighted sum of their inputs, in addition to the bias term, and then produce an output. As experiments showed, the initial optimized values of the weights and biases are generated by the genetic algorithm, which has a great effect on the resulting accuracy. The weights and biases are adjusted for each training tuple in order to reduce the mean squared error between the network's prediction and the actual target value. These adjustments are made in a "backwards" way,

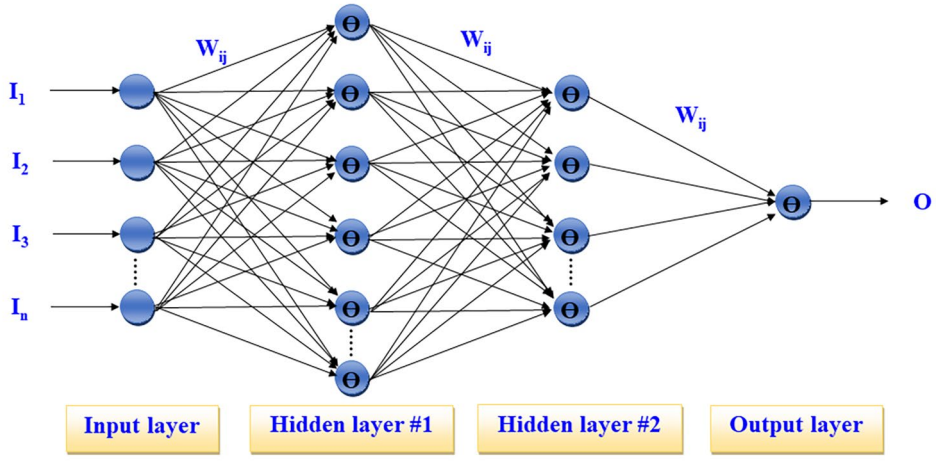

Fig. 1 The multilayer feed-forward neural network topology

starting with the output layer and working down through each hidden layer to the first hidden layer [6].

Each input linked to the unit is multiplied by its associated weight to calculate the net input to the unit, which is then aggregated. The net input, $\mathbf{I_j}$, to unit **j** is computed using Eq. 1:

$$I_j = \sum_i w_{ij} o_i + \theta_j \tag{1}$$

Where $\mathbf{w_{ij}}$ is the weight of the previous layer's link from unit **i** to unit **j**, and $\mathbf{O_i}$ is the previous layer's output from unit **I**, and $\boldsymbol{\theta_j}$ is the bias of the unit **j**. Given the net input $\mathbf{I_j}$ to unit **j**, then $\mathbf{O_j}$, the output of unit **j**, is computed using Eq. 2:

$$O_j = \frac{1}{1 + e^{-I_j}} \tag{2}$$

The output value is calculated for each hidden layer, up to and including the output layer, which yields the network's prediction, where $\mathbf{e^x}$ is the exponential function. To treat the network's prediction error, the prediction deviation is computed and propagated backward by modifying the weights and biases. For a unit **j** in the output layer, the error $\mathbf{Err_j}$ is computed using Eq. 3:

$$Err_j = O_j (1 - O_j) (T_j - O_j) \tag{3}$$

Where $\mathbf{T_j}$ is the known goal value of the provided training tuple, and $\mathbf{O_j}$ is the actual output of unit **j**. For a unit **j** in the hidden layer, the error is computed using Eq. 4:

$$Err_j = O_j (1 - O_j) \sum_k Err_k w_{jk} \tag{4}$$

Where $\mathbf{w_{jk}}$ is the weight of the link from unit **j** to unit **k** in the next higher layer, and $\mathbf{Err_k}$ represents unit **k**'s error. Therefore, the weights and biases are updated to reflect the propagated errors. Equations (5) and (6) are used to update weights, where $\boldsymbol{\Delta w_{ij}}$ is the weight change in $\mathbf{w_{ij}}$:

$$\Delta w_{ij} = (L)\, Err_j O_i \tag{5}$$

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{6}$$

Where the variable L is the learning rate, the learning rate set to **1 / t**, where **t** is the number of iterations through the training set so far. Also, biases are updated by using Eqs. (7) and (8), where $\boldsymbol{\Delta\theta_j}$ is the change in **bias** $\boldsymbol{\theta_j}$:

$$\Delta\theta_j = (L)\, Err_j \tag{7}$$

$$\theta_j = \theta_j + \Delta\theta_j \tag{8}$$

In conclusion, the proposed methodology (EHIDS) aims to develop a system that is able to identify any abnormal behavior. Therefore, it focuses on recognizing any potential attacks that can affect its functionality and preventing it from system intrusion and malfunctioning.

Experimental evaluation

The proposed model (EHIDS) was implemented in Python V. 3.7.6. We tested our work on a Raspberry Pi4 node (Model B) [17], which is a low-cost, credit-card sized computer that runs Linux. Also, it provides a set of general purpose input/output pins to control and explore the Internet of Things. From other hand, in order to make networks more secure, intrusion detection systems attempt to detect intrusions by achieving two goals: high detection and low false-alarm rates. So, in this section, we will show the outcomes obtained from the proposed hybrid intrusion detection system, compared with the outcomes of three another recent approaches, the Cholesky Factorization based Online Sequential Extreme Learning Machines with Persistent Regularization (CF-OSELM-PRFF) [19], the Anomaly Behavior Analysis Intrusion Detection System (ABA-IDS) [22], and the Integrated CNN with LSTM-based Fog Computing Intrusion Detection (ICNN-FCID) model [12], which was introduced previously in the section of the related work.

Testbed description

As illustrated before, the first step in the preprocessing phase is selecting the relevant features according to their standard deviation values. An extensive set of experiments was carried out to determine the best fit value for the standard deviation, which showed that there is a directly proportional relationship between the standard deviation and the accuracy and that an increase in the standard deviation leads to more accurate performance, as illustrated in Table 2. Therefore, we choose features with a minimum standard deviation of 50 because it helps improve accuracy.

On the other hand, the Genetic Algorithm is implemented in such a manner that an initial population of 2000 random chromosomes is generated and evaluated, and the 100 fittest chromosomes are chosen as the starting point for the next generation. We used the following customized genetic algorithm:

Table 2 The Relationship between Standard Deviation and Accuracy

Standard deviation	30	40	50	60	70	80
Accuracy	93.11	93.88	96.47	94.9	94.1	93.2

1. Initialize the population.
2. Evaluate the population by calculating the fitness of each chromosome in the population.
3. Sorting the population, bringing the fittest to the front.
4. Check for termination,
   If the best fitness chromosome exceeds a predefined threshold
   Then return the chromosome.
   Else continue.
5. Select the elitism that represents the core of the next generation of the population.
6. Complete the rest of the population by merging the elitism chromosomes (crossover).
7. Select one gene randomly for mutation.
8. Go to step 2.

As mentioned above, all chromosomes are evaluated according to the evaluation function for each generation. After a number of generations have elapsed or reached a predefined threshold that represents the minimum accepted fitness, the best chromosome from the population is chosen to represent the optimum possible solution to the task under consideration. The values of the best chromosome represent the first initial interconnecting weights of the network and the initial biases associated with each neuron.

The artificial neural network topology will be configured with four layers: one input layer, two hidden layers, and one output layer. Now, the BPNN topology is constructed and ready to use. Firstly, during the training phase, the weight and bias parameters are adjusted so that the mean squared error between the network's prediction and the real goal value is as small as possible. Once the classifier model has been trained and its accuracy is considered acceptable, the classifier model is ready to work and to be uploaded to the Fog nodes. The selected features are used as the input value of BPNN, and the output value is one of the attack labels. The types of attacks will be divided into normal data and attack data, with normal data being numbered 1 and attack data being numbered 2.

Evaluating classifier performance

To determine how effective or "accurate" the classifier is at detecting intrusion attacks, we must test its performance. The following evaluation measures were used in our experiments:

- The accuracy of a classifier.
- Precision.
- True positive rate or recall.
- F-score.

Choosing a suitable performance metric is one of the most important factors to be considered. The accuracy of the system is the most widely used statistic for demonstrating its efficiency. The number of cases successfully identified divided by the total number of cases is known as the classifier's accuracy.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

The probability of the model correctly classifying any instance is known as precision or positive predictive value.

$$Precision = \frac{TP}{TP + FP}$$

The proportion of test results correctly identified by the model is known as the "true positive rate." It is also known as the "recall rate" or "detection rate." It is also most likely known as sensitivity.

$$Recall = \frac{TP}{TP + FN}$$

Finally, the harmonic mean of accuracy and recall is the F-score.

$$Fscore = \frac{2 * (recall * precision)}{(precision + recall)}$$

Table 3 UNSW-NB15 data set distribution

Category	Training set	Testing set
Normal	56,000	37,000
Analysis	2,000	677
Backdoor	1,746	583
DoS	12,264	4089
Exploits	33,393	11,132
Fuzzers	18,184	6,062
Generic	40,000	18,871
Reconnaissance	10,491	3,496
Shellcode	1,133	378
Worms	130	44
Total Records	175,341	82,332

Data sets

Evaluating IDS using traditional benchmark data sets such as KDD Cup 99 [30] and NSL-KDD [33] yields unsatisfactory results owing to three fundamental issues: (1) a lack of current low-footprint attack techniques; (2) a lack of modern typical traffic patterns; and (3) a disparity in the distribution of training and testing data sets. Therefore, for the experimental evaluation purpose of the proposed approach, we used two different datasets, the UNSW-NB15 data set [18] and the ToN_IoT data set [34].

UNSW-NB15 Data Set

This collection of data covers nine types of recent attacks and normal patterns, as well as 49 features that compose the flow based between hosts and the network packet investigation to distinguish between normal and abnormal behaviours. As shown in Table 3, the entire number of records is 2,540,044 records. The used portion of this dataset, about 257,673 records, was divided into an

Table 4 ToN_IoT data set distribution

Category	Training an Testing set
Normal	300,000
Backdoor	20,000
doS	20,000
ddos	20,000
injection	20,000
mitm	1,043
password	20,000
ransomware	20,000
scanning	20,000
xss	20,000
Total Records	461,043

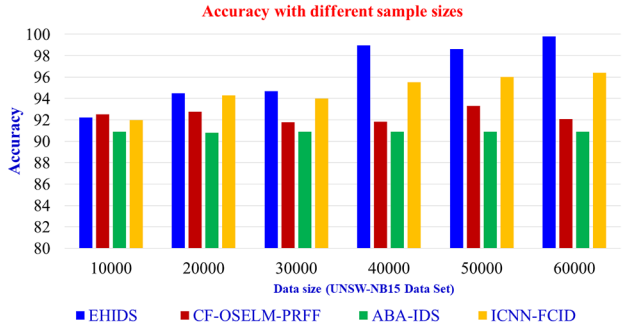

Fig. 2 Accuracy for EHIDS, CF-OSELM-PRFF, ABA-ID,S and ICNN-FCID with different sample sizes

approximate 68%: 32% ratio of the training and testing data sets, respectively. In this case, the training dataset has 175,341 records, of which 56,000 are normal connection records, while the test dataset has 82,332 records, 37,000 of which are normal connection records. Moreover, to achieve the authenticity of IDS evaluations, there are no redundant records among the training and testing sets.

ToN_IoT Data Set

The ToN_IoT (UNSW-IoT20) datasets are new generations of Internet of Things (IoT) and Industrial IoT (IIoT) datasets for evaluating the accuracy and effectiveness of various artificial intelligence-based cybersecurity applications. The datasets include several normal and cyberattack events that were gathered from heterogeneous data sources, i.e., telemetry datasets of IoT and IIoT sensors. There are a total of 22,339,021 records in the datasets. As shown in Table 4, the used portion of this dataset, about 461,043 records, was divided into an approximate 68%:32% ratio of the training and testing data sets, respectively. Moreover, to achieve the authenticity of IDS evaluations, there are no redundant records among the training and testing sets.

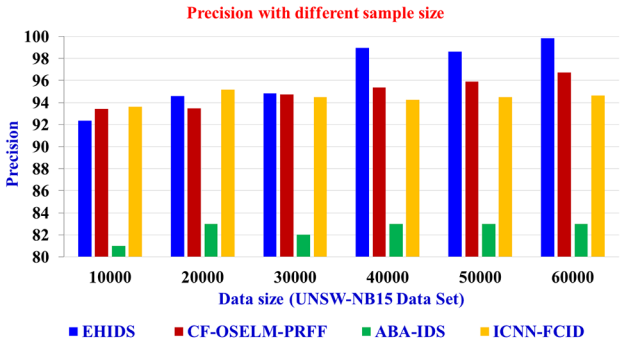

Fig. 3 Precision for EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID with different sample sizes

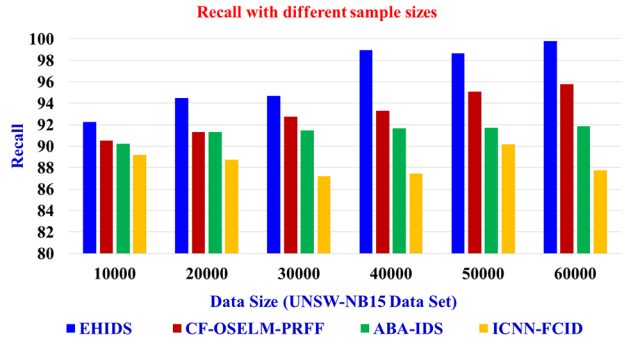

Fig. 4 Recall for EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID with different sample size

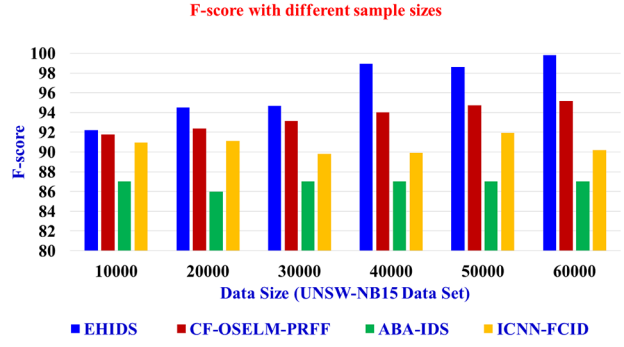

Fig. 5 F-score for EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID with different sample sizes

The experiments

A set of experiments on both the UNSW-NB15 and ToN_IoT datasets, were conducted to compare the performance evaluation measures of the proposed methodology, "EHIDS," and three other recent approaches: "CF-OSELM-PRFF," "ABA-IDS," and "ICNN-FCID." To check the effect of varying the size of the data set on the experimental outcomes, the experiments were repeated with different sample sizes for each performance measure, as shown in Figs. 2, 3, 4, and 5 for the UNSW-NB15 data set and 6, 7, 8, and 9 for the ToN_IoT data set, respectively.

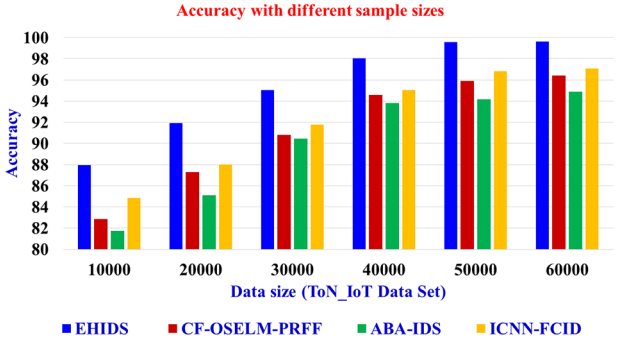

Fig. 6 Accuracy for EHIDS, CF-OSELM-PRFF, ABA-ID,S and ICNN-FCID with different sample sizes

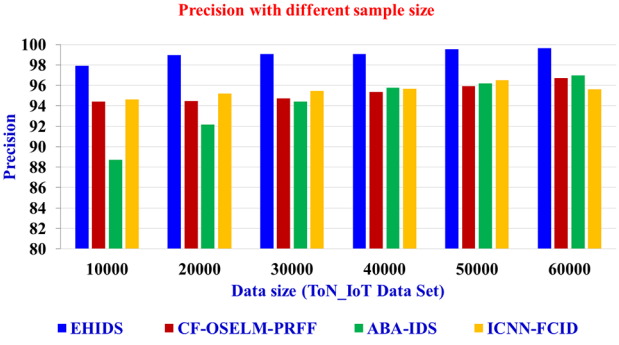

Fig. 7 Precision for EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID with different sample sizes

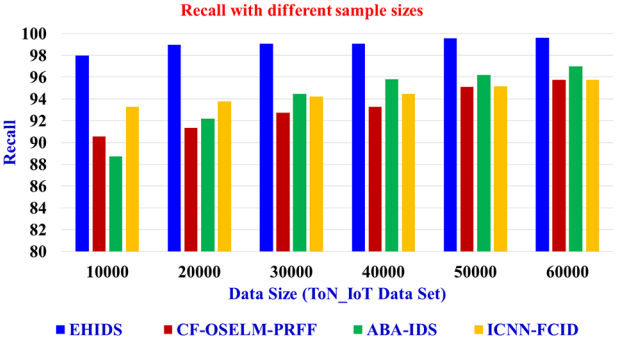

Fig. 8 Recall for EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID with different sample sizes

As shown in Figs. 2, 3, 4, 5, 6, 7, 8 and 9, a comparison is made between the performances of "EHIDS," "CF-OSELM-PRFF," "ABA-IDS," and "ICNN-FCID" through a different set of experiments with different measures, in which the x-axis represents the sample size and the y-axis denotes the performance measure. The experiments were applied to different sample sizes to verify the effect of varying data set sizes on the experimental results. As we note, with the small data volume, the model was not trained enough to be able to show efficient performance (i.e., 10–30 K). This means that the model is not trained to our expectations (the model has low training errors

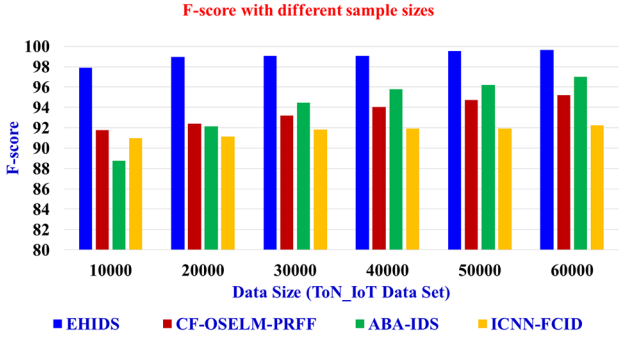

Fig. 9 F-score for EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID with different sample sizes

Table 5 the binary classification results of EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID.

Data set	Comparison criteria	EHIDS	CF-OSELM-PRFF	ABA-IDS	ICNN-FCID
UNSW-NB15 Data Set	Accuracy	96.47%	94.70%	90.88%	92.38%
	Precision score	96.53%	94.94%	82.50%	94.44%
	Recall score	96.47%	93.12%	91.37%	88.41%
	F-score	96.47%	93.54%	86.83%	90.66%
ToN_IoT Data Set	Accuracy	95.36%	91.31%	90.03%	92.26%
	Precision score	99.02%	95.27%	94.05%	95.51%
	Recall score	99.04%	93.12%	94.05%	94.43%
	F-score	99.02%	93.54%	94.05%	91.66%

and high testing errors). When the training data volume reaches a sufficient volume that enables the model to enhance its performance, we will notice an improvement in the performance of the model with the increase in the data volume (i.e., 40–60 K). Furthermore, it can be observed that the proposed approach (EHIDS) outperforms other related work approaches across all measures. Table 4 shows the average assessment of each methodology for each performance measure.

As shown in Table 5, the proposed methodology EHIDS outperforms other related methodologies in terms of accuracy, which shows how our system is more powerful in detecting incoming intrusions. Also, EHIDS achieved higher precision, which reflects the ability of EHIDS to detect the correct classification. Also, EHIDS achieved a higher recall score than other methodologies, which reflects the ability of EHIDS to classify the normal connection instances better than others. Finally, the results show that the EHIDS has a higher value of F-score, which indicates that the EHIDS is performing better on recall and precision.

In addition to using the traditional evaluation methods, e.g., accuracy, detection rate, precision, and false alarm

Table 6 The confusion matrix for binary classification

		Predicted label	
	Attack type	**Normal**	**Intrusion**
Actual class	Normal	True negative (14,606)	False positive (4712)
	Intrusion	False negative (49)	True positive (40,633)

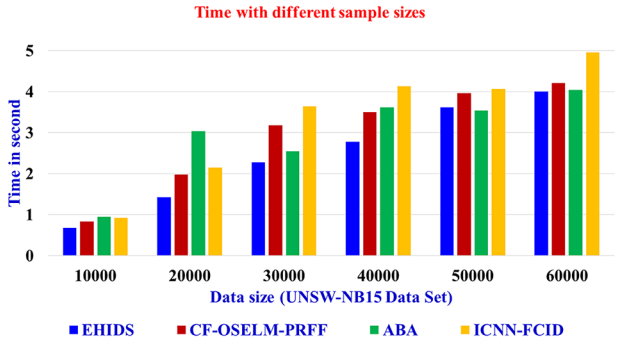

Fig. 10 the execution times for EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID with different sample sizes (UNSW-NB15 data set)

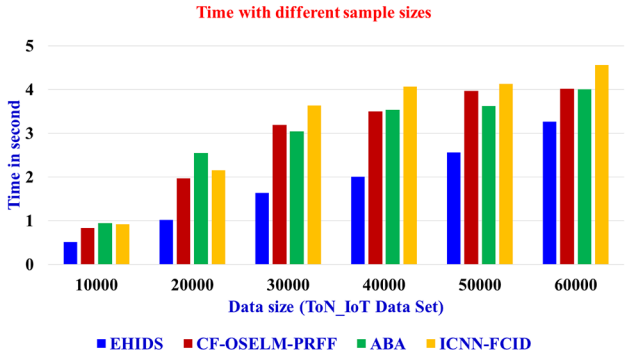

Fig. 11 the execution times for EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID with different sample sizes (ToN_IoT Data Set)

rate, to evaluate the performance of IDS, we also used the confusion matrix to evaluate the proposed methodology performance. The confusion matrix is a tabular structure that depicts both the predicted and actual categorization. There are four prospective outputs, as illustrated in Table 6, with a sample of 60,000 records. In which, true positive (TP) refers to the number of actual attacks classified as such; true negative (TN) refers to the number of normal connections classified as such; false positive (FP) refers to the number of normal connections classified as attack class; and false negative (FN) refers to the number of actual attacks classified as normal connection. Both (TP) and (TN) are regarded as guides for the IDS's correct behavior. Furthermore, (FP) and (FN) rates reduce the effectiveness of the IDS, where (FP) reduces the system's detection capability and (FN) makes the system vulnerable to intrusion [15]. As a result, in order for IDS to be effective, the (TP) and (TN) rates should be maximized while the (FP) and (FN) rates should be minimized.

Table 7 Average execution times in seconds for EHIDS, CF-OSELM-PRFF, ABA-IDS, and ICNN-FCID.

Data Set	EHIDS	CF-OSELM-PRFF	ABA-IDS	ICNN-FCID
UNSW-NB15	2.461	2.942	2.951	3.309
ToN_IoT	1.832	3.242	2.946	2.911

After considering all the performance measurements used in our experiments, we have come to the conclusion that our proposed approach - which uses GA as an optimization method to generate the initial interconnecting weights of the network and the initial biases associated with each neuron - leads to achieving a higher detection rate and a lower false positive rate.

Furthermore, Figs. 10 and 11 show a comparison between the execution times of "EHIDS," "CF-OSELM-PRFF," "ABA-IDS," and "ICNN-FCID" through a different set of experiments with different dataset sizes and based on both the UNSW-NB15 and ToN_IoT datasets. Where the x-axis represents the sample size and the y-axis denotes the execution time in seconds. It can be observed that the proposed approach (EHIDS) achieves the lowest execution time on both data sets compared to the other related work methods.

Although the Raspberry Pi4, as a platform for evaluating the model's performance, is much slower than standard computers because it has limited computing resources, this approach has brought other benefits, which can be concluded in an 16.35% and 37.07% reduction of average execution time compared to the best average execution time of other related approaches, for both data sets, respectively, enhancing the acceleration of the convergence process and saving processing power, as illustrated in Table 7.

On the whole, through applying the proposed methodology, we could achieve an enhancement in both the performance metrics and the execution time compared to other state-of-the-art related methodologies. Which means that the improvement in both the performance metrics and the execution time is due to the fact that the optimization of the interconnecting weights of the network and the biases associated with each neuron causes a quick and effective establishment of a back-propagation neural network model. Moreover, the distributed architecture of Fog computing enables the distribution of the intrusion detection system over local Fog nodes with a centralized Cloud, which achieves faster attack detection than the Cloud intrusion detection mechanism. Which reveled in enhancing the performance of the classification algorithm in addition to the titanic reduction in the execution time.

Conclusion & future work

The features of neural networks make them ideal for addressing difficult pattern classification challenges. However, its applicability to various real-world issues has been impeded due to a lack of supporting algorithms that reliably find a nearly globally optimal set of weights and biases in a relatively short time. A genetic algorithm is a type of optimization method that can be considered perfect for exploring a large and complex space in an effective way to find a value that is close to the global optimum. In this paper, we propose a methodology that optimizes the artificial neural network parameters through the use of a genetic algorithm and then deploys it over the Fog network. In which the network's linking weights and the biases associated with each neuron are optimised using a genetic algorithm. This can help to establish a quick and effective BPNN model that could be uploaded to the Fog network for intrusion classification purposes. Classification experiments on both the UNSW-NB15 and ToN_IoT data sets based on the Fog node Raspberry Pi4 revealed that the optimized model has a higher intrusion detection rate through decreasing the neural network error rate and increasing the true positive rate. Furthermore, achieving up to 37.07% reduction in the execution time compared to other state-of-the-art methods, which accelerates the convergence process and saves processing power. For future work, we recommend performing further experiments on multi-class classification based on diverse datasets to validate the suggested model. Moreover, we are planning to apply other effective deep learning algorithms and support our proposed model with more powerful statistical analysis techniques, aiming to enhance the proposed methodology. Finally, conducting more experiments using different evaluation criteria, such as energy and power consumption rates.

List of abbreviations

ABA-IDS	Anomaly Behavior Analysis Intrusion Detection System
ANN	Artificial Neural Network
BPNN	Back Propagation Neural Networks
CF-OSELM-PRFF	Cholesky Factorization-based Online Sequential Extreme Learning Machines
CNN	Convolutional Neural Network
DDoS	Distributed Denial of Service
DoS	Denial of Service
EHIDS	Enhanced Hybrid Intrusion Detection System
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
ICNN-FCID	Integrated CNN with LSTM-based Fog Computing Intrusion Detection
IDS	Intrusion Detection Systems
IoT	Internet of Things
IIoT	Industrial Internet of Things
LSTM	Long Short-Term Memory networks
MLP	Multilayer Perceptron
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

Authors' contributions

Doaa Mohamed proposed the methodology; did all the experiments; wrote the main manuscript text; prepared all the figures except Fig. 1; and prepared all the tables. Osama Ismael gave some valuable insights on the proposed methodology, prepared Fig. 1, and reviewed the manuscript.

Funding

There are no sources of funding.

Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations

Competing interests

There is no conflict of interest regarding all the authors.

Received: 1 August 2022 / Accepted: 11 March 2023

References

1. Abbas A, Khan MA, Latif S et al (2022) A New Ensemble-Based intrusion detection system for internet of things. Arab J Sci Eng 47:1805–1819. https://doi.org/10.1007/s13369-021-06086-5
2. Alghayadh F, Debnath D (2021) A hybrid intrusion detection system for Smart Home Security based on machine learning and user behavior. Adv Internet Things 11(01):10–25. https://doi.org/10.4236/ait.2021.111002
3. Aljawarneh S, Aldwairi M, Yassein M (2018) Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J Comput Sci 25:152–160. https://doi.org/10.1016/j.jocs.2017.03.006
4. Amaral J, Oliveira L, Rodrigues J, Han G (2014) and Shu L. "Policy and network-based intrusion detection system for IPv6-enabled wireless sensor networks", IEEE International Conference on Communications (ICC). doi: https://doi.org/10.1109/icc.2014.6883583
5. An X, Su J, Lü X, Lin F (2018) Hypergraph clustering model-based association analysis of DDOS attacks in fog computing intrusion detection system. EURASIP J Wirel Commun Netw 2018(1). https://doi.org/10.1186/s13638-018-1267-2
6. Han J, Kamber M, Pei J (2012) "Data Mining: Concepts and Techniques", 393–442. doi: https://doi.org/10.1016/b978-0-12-381479-1.00009-5
7. Hindy H, Brosset D, Bayne E, Seeam A, Tachtatzis C, Atkinson C, Bellekens X (2018) "A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets", Working paper arXivorg
8. Houda Z, Brik B, Khoukhi L (2022) "Why should I trust your IDS?": an Explainable Deep Learning Framework for Intrusion Detection Systems in Internet of Things Networks. IEEE Open Journal of the Communications Society 3:1164–1176. https://doi.org/10.1109/OJCOMS.2022.3188750
9. Illy P, Kaddoum G, Miranda C, Kaur K, Garg S (2019) "Securing Fog-to-Things Environment Using Intrusion Detection System Based on Ensemble Learning", IEEE Wireless Communications and Networking Conference (WCNC). doi: https://doi.org/10.1109/wcnc.2019.8885534
10. Imrana Y, Xiang Y, Ali L., and, Abdul-Rauf Z (2021) A bidirectional LSTM deep learning approach for intrusion detection, vol 185. Expert Systems with Applications
11. Jan S, Ahmed S, Shakhov V, Koo I (2019) Toward a Lightweight Intrusion Detection System for the internet of things. IEEE Access 7:42450–42471. https://doi.org/10.1109/access.2019.2907965
12. Kalaivani K, Chinnadurai M (2021) "A Hybrid Deep Learning Intrusion Detection Model for Fog Computing Environment", Intelligent Automation & amp; Soft Computing, vol. 29, no. 3, pp. 1–15. doi: https://doi.org/10.32604/iasc.2021.017515
13. Ke G, Hong H (2014) The Research of Network Intrusion Detection Technology based on genetic algorithm and BP neural network. Appl Mech Mater 599–601. https://doi.org/10.4028/www.scientific.net/amm.599-601.726

14. Khan S, Parkinson S, Qin Y (2017) Fog computing security: a review of current applications and security solutions. J Cloud Comput 6(1). https://doi.org/10.1186/s13677-017-0090-3
15. Khater B, Abdul Wahab A, Idris M, Hussain M, Ibrahim A, Amin M, Shehadeh H (2021) Classifier performance evaluation for Lightweight IDS using Fog Computing in IoT Security. Electronics 10(14):1633. https://doi.org/10.3390/electronics10141633
16. Khraisat A, Gondal I, Vamplew P, Kamruzzaman J (2019) "Survey of intrusion detection systems: techniques, datasets and challenges", Cybersecurity, 2(1). doi: https://doi.org/10.1186/s42400-019-0038-7
17. Ltd R (2022) Raspberry Pi. from https://www.raspberrypi.com/
18. Moustafa N, Slay J (2016) Inform Secur Journal: Global Perspective 25(1–3):18–31. https://doi.org/10.1080/19393555.2015.1125974. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set"
19. Narendra M, Rakesh K (2019) A novel intrusion detection technique based on Fog Computing using Cholesky Factorization based Online Sequential Extreme Learning Machines with persistent Regularization", IJCA. 12:117–1266
20. Nayak S, Misra B, Behera H (2014) Impact of data normalization on stock index forecasting. Int J Comput Inform Syst Industrial Manage Appl 6:357–369
21. Pacheco J, Hariri S (2016) "IoT Security Framework for Smart Cyber Infrastructures", IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W). doi: https://doi.org/10.1109/fas-w.2016.58
22. Pacheco J, Benitez V, Felix-Herran L, Satam P (2020) Artificial neural networks-based intrusion detection system for internet of Things Fog Nodes. IEEE Access 8:73907–73918. https://doi.org/10.1109/access.2020.2988055
23. Pudjihartono N, Fadason T, Kempa-Liehr AW, O'Sullivan JM (2022) A review of feature selection methods for machine learning-based Disease Risk Prediction. Front Bioinform 2:927312. https://doi.org/10.3389/fbinf.2022.927312
24. Rani D, Kaushal N (2020) "Supervised Machine Learning Based Network Intrusion Detection System for Internet of Things", 11Th International Conference on Computing, Communication and Networking Technologies (ICCCNT). doi: https://doi.org/10.1109/icccnt49239.2020.9225340
25. Ravi V, Chaganti R, Alazab M (2022) "Recurrent deep learning-based feature fusion ensemble meta-classifier approach for intelligent network intrusion detection system",Computers and Electrical Engineering, Volume 102
26. Sadaf K, Sultana J (2020) Intrusion detection based on autoencoder and isolation forest in Fog Computing. IEEE Access 8:167059–167068
27. Srinivasu P, Avadhani P (2012) Genetic algorithm based weight extraction algorithm for Artificial neural network classifier in intrusion detection. Procedia Eng 38:144–153. https://doi.org/10.1016/j.proeng.2012.06.021
28. Sudqi B, Abdul Wahab A, Idris M, Abdulla M, Ahmed A (2019) A Lightweight Perceptron-Based intrusion detection system for Fog Computing. Appl Sci 9(1):178. https://doi.org/10.3390/app9010178
29. Systems C (2016) "Fog Computing and the internet of things: extend the cloud to where the Things Are",www.Cisco.Com,
30. Tavallaee M, Bagheri E, Lu W, Ghorbani A (2009) "A Detailed Analysis of the KDD CUP 99 Data Set", Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)
31. Whitley D (1994) A genetic Algorithm Tutorial. Stat Computing" 4(2). https://doi.org/10.1007/bf00175354
32. Yousefpour A, Ibrahim R, Abdul Hamed H, Zaki U, Mohamed K (2017) "Feature subset selection using mutual standard deviation in sentiment mining", IEEE Conference on Big Data and Analytics (ICBDA), doi:https://doi.org/10.1109/icbdaa.2017.8284100
33. NSL-KDD dataset, https://www.unb.ca/cic/datasets/nsl.html
34. ToN_IoT datasets (2020) https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-ton-iot-Datasets/, January

Publisher's Note