MethodsX 12 (2024) 102597

Contents lists available at ScienceDirect

MethodsX

journal homepage: www.elsevier.com/locate/methodsx

A secure edge computing model using machine learning and IDS to detect and isolate intruders

Poornima Mahadevappa[a], Raja Kumar Murugesan[a], Redhwan Al-amri[b], Reema Thabit[c], Abdullah Hussein Al-Ghushami[d,*], Gamal Alkawsi[e]

[a] School of Computer Science and Engineering, Taylor's University, Malaysia
[b] Centre for Professional Learning & Leadership, Institute of Education and Humanities, University of Wales Trinity Saint David, Swansea, UK
[c] Department of Computing, College of Computing and Informatics (CCI), Universiti Tenaga Nasional (UNITEN), Kajang, Selangor 43000, Malaysia
[d] Department of Information Technology, Community College of Qatar, Doha, Qatar
[e] Institute of Sustainable Energy, Universiti Tenaga Nasional, Kajang 43000, Malaysia

ARTICLE INFO

Method name:
Hybrid LDA-LR

Keywords:
Edge computing
Machine learning
Intrusion detection
Intrusion isolation
LDA-LR
Edge security

ABSTRACT

The article presents a secure edge computing model that utilizes machine learning for intrusion detection and isolation. It addresses the security challenges arising from the rapid expansion of IoT and edge computing. The proposed Intrusion Detection System (IDS) combines Linear Discriminant Analysis (LDA) and Logistic Regression (LR) to swiftly and accurately identify intrusions without alerting neighboring devices. The model outperforms existing solutions with an accuracy of 96.56%, precision of 95.78%, and quick training time (0.04 s). It is effective against various types of attacks, enhancing the security of edge networks for IoT applications.

- The methodology employs a hybrid model that combines LDA and LR for intrusion detection.
- Machine learning techniques are used to analyze and identify intrusive activities during data acquisition by edge nodes.
- The methodology includes a mechanism to isolate suspected devices and data without notifying neighboring edge nodes to prevent intruders from gaining control over the edge network.

Specifications table

Subject area:	Computer Science
More specific subject area:	Cyber security/Intrusion Detection in Edge Computing Environments
Name of your method:	Hybrid LDA-LR
Name and reference of original method:	M. Pohar Perme, M. Blas, and S. Turk, "Comparison of logistic regression and linear discriminant analysis : a simulation study," Metod. Zv., vol. 1, no. 1, pp. 143–161, 2004 - https://doi.org/10.51936/ayrt6204
Resource availability:	N.A.

* Corresponding author.
E-mail address: abdullah.alghushami@ccq.edu.qa (A.H. Al-Ghushami).

https://doi.org/10.1016/j.mex.2024.102597
Received 16 November 2023; Accepted 31 January 2024
Available online 13 February 2024

Method details

Introduction

In the era of IoT devices, there is a massive global growth of edge computing, which is projected to reach nearly 6.5 billion by 2030. This marks an increase of over four billion from the figures recorded in 2020 [1]. It advanced IoT and cloud computing development by supporting computing power and innovative services in the edge network. It provides a decentralized platform for connected devices that improves performance, reduces latency, and better supports real-time applications, making it a valuable solution for a variety of industries and use cases. However, edge computing caused a significant security threat and privacy issues [2]. Edge devices are usually deployed in places that are not of rigorous surveillance and protection. Therefore, conventional attacks like eavesdropping, data hijacking, man-in-the-middle, and many others become possible for a malicious intruder or user [3]. Some common approaches to address the security issues in edge computing are establishing trust, privacy preservation techniques, authentication and key agreement techniques, and intrusion detection systems (IDS). IDS is the most widely used method to identify and mitigate the attack. It is deployed at all levels of three-tier edge computing architecture [4]. It is used to analyze network traffic, and data inflow, monitor the behavior of edge nodes, IoT devices, or cloud servers, and generate a proactive, responsive approach to prevent the attack.

The proactive responses may be sending an alarm to other nodes, using an access control policy, blocking users' services, or sending notifications [5]. However, all the incidents that are detected as attacks may not be an attack. For example, a user may erroneously obtain access to a different system by entering a different address without authorization [6]. In another example, a cybercriminal might have performed the same action to exploit liabilities in the system. Therefore, there is a need to distinguish between malicious (cybercriminal) from non-malicious activity (a person erroneously acquiring access) [7]. Further, the false alarm or generating notification for the other edge nodes can create computation load and cause chaos in the edge layer by affecting characteristic features of edge computing and disrupting the typical function of the edge applications [8].

Machine learning (ML) and deep learning techniques have been highly used in IDS for feature extraction, detection, computer vision, or recommendation systems. Real-time data streaming in IoT and edge nodes requires real-time anomaly detection [9]. However using deep learning may require high memory and computing resources, so it may be challenging to adopt deep learning in edge networks [10]. Whereas in ML-based IDS, the feature selection process is a mandatory phase that possibly lessens computational complexity. This mandatory phase identifies the relevant features from the original data, improving overall classification accuracy [11]. In ML, Linear discriminant analysis (LDA) and Logistic Regression (LR) are broadly used for multivariate statistical data analysis. In LDA, normally distributed explanatory variables get better results when the normality assumption of intrusions is fulfilled [11]. Whereas in LR, there is no assumption on the distribution of the descriptive data. When the sample size is vast, the LDA predicts the attack estimation of mean and variance accurately. So, in our proposed model, LDA is used for feature extraction in the initial stage. On the contrary, LR is better in the case of a small sample size. Therefore, LR is used for the classification of the attack on the feature-reduced dataset [12]. Overall, these two methods are robust in categorization. Therefore, adopting these methods together in the edge layer can give better accuracy and response in terms of intrusion detection to support edge computing features [12].

This research proposes an intrusion detection system based on ML using the LDA-LR hybrid model that isolates the device and data identified as an intrusion. During this process, no alarm or notification is sent to any neighboring edge nodes. The edge node that sources the data from IoT devices in the edge layer performs the feature extraction and identification of the attack. Later the edge node informs the edge server to isolate the device and data from the other edge nodes. In this way, the proposed model is significant by not creating any chaos or disturbing the normal functioning of the neighboring edge nodes. The key contributions of this research are:

- A hybrid LDA-LR-based intrusion detection approach to perform feature extraction and isolate the attack.
- Compute the time complexity of the proposed hybrid LDA-LR model.
- Analyze the performance efficiency of the proposed model and compare it with the existing models.

The rest of this paper is organized as follows; Section II introduces similar IDS work in edge computing. The proposed model and its time complexity are explained in detail in section III. Section IV presents the experimental setup, results, and evaluation. Section V presents a comparison with existing methods. Section VI includes the discussion and conclusion of the proposed work with future work in section VII.

Related work

Edge computing has revolutionized cloud computing by deploying computing, storage, and networking services to the edge of the network. However, it encounters conventional security threats, which increases security and privacy disputes on the edge layer. Considerable research on security threats shows that security in edge computing is the most researched topic. There is a vast number of research on edge computing that mainly focuses on analyzing security threats and developing countermeasures [13–16]. There are many countermeasures like access control mechanisms, trust models, reputation models, and IDS to handle security issues [17]. Among all these IDS is an efficient network monitoring system without altering the network packets. The main advantages of IDS are that it can be a) highly adaptive using a knowledge base during anomaly changes, fluctuating user behavior, or identifying any gradual change, b) Online IDS support uninterrupted monitoring and are highly expandable, and c) batch audit-based IDS monitors anomalies using low periods of central processing usage.

The local edge data centers or edge networks can employ IDS to monitor and analyze the system logs for unauthorized access in edge computing. The edge network located one hop away from mobile devices can efficiently mesh to form a security framework to detect any intrusion. Furthermore, by adopting software-defined networks, network virtualization can reduce network costs and scale the resources. Therefore, considering these advantages of IDS in edge computing to provide security, the proposed work focuses on developing an efficient IDS to secure edge-based applications.

Currently, there are many studies where edge computing is used as an intrusion detection layer to provide security to cloud servers or IoT applications. For example, in the cognitive fog computing layer [18]– the local edge nodes identify the intrusions using Online Sequential Extreme Machine Learning and the support of centralized cloud intelligence. The centralized cloud monitors the actions of edge nodes; thereby, the edge nodes cannot take any immediate action upon detecting attacks. However, the distributed architecture of edge computing reinforces the intrusion detection scheme's scalability, flexibility, and interoperability. IMPACT [19]– lightweight ML-based IDS is deployed on edge nodes to evaluate the data before passing it to the cloud. The resource-constrained edge nodes use a stacked autoencoder, a neural network for feature extraction, mutual information, and feature selection. And later, for detecting attacks, gradient descent optimization-based SVM (Support Vector Machine) is used. Though it is suitable for impersonation attacks, it cannot identify similar attacks like flooding or injection.

Empirical, semi-supervised, or cyber security-based IDS has demonstrated high detection accuracy, adequate computational performance, and close cooperation between the model and edge computing. They collect network data and perform feature extraction to remove redundant data that assist classifiers. The classifier finally uses these features to train the model, identify the attack with high accuracy, and raise the alarm to the server [20–22]. These approaches can be trained efficiently and made suitable for a particular model, but it might be challenging to adopt for all the models in common. Apart from this, few efficient methods identify intrusion based on their interactions with the edge nodes. The interaction is initiated through the backpropagation approach or honey pot techniques. They maintain a separate log to track their interactions and analyze their behavior based on their responses. These models can assist in identifying unknown attacks more precisely [23–26]. However, frequent interaction can make the system more vulnerable to attacks while identifying the motive of cyber criminals [27].

Machine Learning algorithms are commonly used to identify the attacks in IDS because the preprocessing used in these algorithms considerably improves detection accuracy and has superior performance [28]. Few approaches perform intrusion detection based on the resources allocated to each node. If nodes use more resources than the allocated capacity, then they are identified as an intrusion. This approach can easily detect DoS, impersonation, or multistage attacks [25,29]. During intrusion detection, to confirm and mitigate the intrusion, alarms are raised in IDS. There are probabilities that alarms might not be true in all instances. The attackers can send a flood of irrelevant data packets to the IDS host and create a false alarm. This can make IDS resources exhausted and make the system vulnerable [30]. Therefore, few approaches concentrate exclusively on alarm reduction [19,26,31]. These are collaborative IDS that cannot be efficient considerably. First, intrusion detection must be performed, and further alarm monitoring must be included. In addition, this collaboration will require extra security considerations to identify accurate attacks and falsely considered attacks. There can be instances where users may erroneously gain access to the network or send data packets with a few junks. Hence if IDS supports alarm monitoring, then it should also identify non-malicious users [32].

Autoencoder and isolation forest-based IDS are effectively integrated with two-staged detection to identify anomaly and normal data packets. They include historical data and perform categorizations that span from minutes to days. This categorization assists in isolating the intruders from the network and reducing their impact on the network. However, the cloud servers perform the complete analysis, and the edge nodes are loaded with the historical data [33]. The main drawback of these approaches is tracking all the read and write operations in separate log files. If any operations and transactions are lost, then recovery cannot be guaranteed by the edge nodes.

From the above analyses, it can be summarized that IDS is an efficient and straightforward intrusion monitoring system. They identify any simple deviation in the network and inform the servers to mitigate the attacks. IDS can be an appropriate method to alleviate any issues in edge computing if the identified gaps are addressed effectively. To summarize the identified research gaps, frequent interactions to identify intrusion will make the system vulnerable, and the attackers can understand the system flow. This results in the attacker gaining control over the system. Next, the attackers can create false alarms and make IDS resource exploitation. It is appropriate to include false alarms or stop alarm generation since it might require edge nodes to prepare to handle them. And finally, there are instances where non-malicious users are identified as malicious users. So, IDS should be capable of distinguishing between malicious and non-malicious users.

Table 1 shows the works centered around the development of advanced intrusion detection systems (IDS) for the Internet of Things (IoT) and edge computing environments. These environments are increasingly becoming the target of cyber threats, necessitating robust and efficient IDS. The study in [34] addressed the challenge of filtering and selecting transmitted data from IoT devices, which can often be unrelated, duplicated, or erroneous. A deep learning model that combines Convolutional Neural Networks (CNN) and Gated Recurrent Unit (GRU) algorithms was proposed and tested using the NSL-KDD and UNSW-NB15. Similarly, in [35] the study identified complex cyber-attacks. They proposed a two-layered distributed and lightweight IDS. The system was able to accurately characterize normal behavior within fog nodes and detect different attack types such as DDoS attacks with a high detection rate (99.98%) and low false alarms rate (less than 0.01%)

Autoencoder (AE) and Isolation Forest (IF) for the fog environment model focused on intrusion detection exclusively in edge frameworks. Their approach achieved a high accuracy rate of 95.4% as compared to many other state-of-art intrusion detection methods [33]. A novel wrapper FS model that uses the Emperor Penguin Colony (EPC) method to explore the issue space and a K-nearest neighbor classifier was proposed to solve FS for IoT challenges. The proposed EPC model is a novel approach to intrusion detection for IoT systems that achieves high accuracy and efficiency in filtering and selecting transmitted data [36]. The study in

Table 1
IDS in Edge computing environments.

Refs.	Problem	Research Method	Contribution
[34]	The data sent via IoT devices can be unrelated, duplicated, or erroneous, posing a challenge for performing required tasks. Therefore, filtering and selecting transmitted data are necessary to achieve the highest possible level of security	Design a deep learning model that combines CNN and GRU algorithms using the NSL-KDD and UNSW-NB15 datasets.	It achieves a detection rate that is 1.5 times more accurate than other IDSs.
[36]	The data sent via IoT devices can be unrelated, duplicated, or erroneous, posing a challenge for performing required tasks. Therefore, filtering and selecting transmitted data are necessary to achieve the highest possible level of security and address specific problem characteristics.	proposes a novel wrapper FS model that uses the emperor penguin colony (EPC) method to explore the issue space and a K-nearest neighbor classifier to solve FS for IoT challenges	The proposed EPC model is a novel approach to intrusion detection for IoT systems that achieves high accuracy and efficiency in filtering and selecting transmitted data
[35]	Identify complex cyber attacks	Proposes two layered distributed and lightweight IDS	The proposed system can characterize accurately the normal behavior within fog nodes and detect different attacks with a high detection rate (99.98%) and low false alarms rate (less than 0.01%).
[6]	high-dimensional nature of networking data exacerbated IDS	ML base Effective Seeker optimization model to choose optimal features for intrusion detection	Effectively identifies the occurrence of intrusions
[33]	Intrusion in edge framework	Deep learning approach using Autoencoder (AE) and Isolation Forest (IF) for the fog environment	achieves a high accuracy rate of 95.4% as compared to many other state-of-art intrusion detection methods
[37]	Addresses various intrusions, that can disrupt network resources and impede authenticated users.	Proposes integrated Convolutional Neural Network (CNN) with Long Short-Term Memory networks (LSTM)-based Fog Computing Intrusion Detection model	The model demonstrates a high attack detection accuracy of about 96.5%

[6] addressed the high-dimensional nature of networking data that exacerbated IDS. They proposed an ML-based Effective Seeker optimization model to choose optimal features for intrusion detection. Finally, an integrated Convolutional Neural Network (CNN) with Long Short-Term Memory Networks (LSTM)-based Fog Computing Intrusion Detection model addressed various intrusions that can disrupt network resources and impede authenticated users. The model demonstrated a high attack detection accuracy of about 96.5% [37].

Despite these significant contributions, there are some gaps in these studies. One notable gap is that none of the papers have discussed isolating the intrusions identified during detection. Isolating intrusions could potentially improve the efficiency of data analysis by allowing for a more focused examination of suspicious activities without the noise of normal network traffic [38]. This could lead to faster response times in mitigating threats and less computational resources spent on analyzing benign activities [39].

Furthermore, as these models mostly rely on datasets for testing, it is uncertain how applicable they are in the actual world. It is also not covered in detail how these models scale with the increasing number of IoT devices. The capacity of these models to adapt to various network configurations is also not thoroughly investigated. These studies also fail to fully address privacy issues relating to data transfer in IoT networks. Lastly, practical challenges related to the implementation and deployment of these systems in real-world environments are not discussed.

Proposed LDA-LR intrusion detection method

A brief explanation of every factor of the proposed IDS model is given in the following sub-divisions.

Data preprocessing

The initiation of intrusion detection begins through data analysis. Continuous data and discrete data are two categories of data available. For continuous data, normalization is performed before data reduction and classification. This involves covariance calculation and the Z-score normalization method to eliminate dimensional variance and covariance.

Eq. (1) shows the Z-score normalization.

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where x is the input data stream, z is the normalized output data stream; μ and σ are the mean and variance of the continuous and discrete input datasets.

For discrete data, the One-Hot Encoder is used to encode categorial integer features using a one-hot scheme. Categorial features of data are passed as input, and a sparse matrix representing data features in an individual column is the output. This approach reduces normalizing multiple parameters and improves the nonlinear capability of the algorithm.

Linear discriminant analysis

It is a dimensionality reduction technique that involves removing duplicate and related features from the dataset. They transform the features from higher dimensionality to lower dimensionality. The separability between different classes is first calculated to achieve these features, and then the distance between means of other classes. This is called between-class variance or between-class matrix (S_b). Next, we calculate the distance between the mean and samples of each class, called a within-class variance or within-class matrix. Finally, a transformation matrix of lower-dimensional space is constructed, maximizing the between-class matrix and minimizing the within-class matrix [40].

In a given N train sample, $D= \{x_k, t_k\}$, $k = 1,2,3\ldots, N$. Presume that $x_{ij} \in \{x_k\}$, $t_{ij} \in \{t_k\}$, $i = 1,2, 3,\ldots, c$; $j = 1,2,3\ldots n_i$. x_{ij} is the jth sample feature vector of ith class and t_{ij} is the sample corresponding to x_{ij} with d dimensions. The feature matrix can be expressed as X^{Nxd} with c types and n_i is the sample with class i where $N = \sum_{i=1}^{c} n_i$. Finally, the mean sample of vector u and the class u_i is

$$u = \frac{1}{N} \sum_{i-1}^{c} \sum_{j=1}^{n_i} x_{ij} \tag{2}$$

$$u_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ij} \tag{3}$$

Defining between-class matrix (S_b): The separation distance between classes with varied spatial similarities and the sample's center is the between-class matrix (S_b). It denotes the range of the high dimensional data spatial similarity measurement function and the dispersion between classes. It is expressed as

$$S_b = \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} f_{ij} n_i n_j \left(\mu_i - \mu_j\right) \left(\mu_i - \mu_j\right)^T \tag{4}$$

$$f_{ij} = \sum_{k=1}^{d} \frac{1}{1 + \left|u_{i,k} - u_{j,k}\right|} \tag{5}$$

where f_{ij} is spatial similarity dimension function with high dimensional data. It represents the spatial similarity of (μ_i, μ_k) data. While the mean values of the data are represented as ($\mu_{i,k}, \mu_{j,k}$) with i and j in k dimensions, d is feature dimensions. The number of samples here are n_i and n_j.

Defining within-class matrix (S_w): To minimize within-class variance, the within-class matrix (S_w) is defined as the difference between the mean and the sample of the class. It is the mean square error of the distance between each class of the sample and its center. It shows the degree of diffusion of the same class of sample. It is expressed as

$$S_w = \sum_{i=1}^{c} \sum_{j=1}^{n_j} \left(x_{ij} - u_i\right)\left(x_{ij} - u_i\right)^T \tag{6}$$

Defining transformation matrix with lower dimension space: The lower-dimensional space is constructed by maximizing the between-class matrix and minimizing the within-class matrix. The transformation matrix A^* is expressed as

$$A^* = argmax_A \frac{A^T \left(S_b - S_w\right) A}{A^T I A} \tag{7}$$

I is the identity matrix, A^* is the projection matrix. Based on the Rayleigh quotient, eigenvectors and corresponding eigenvalues of $I^{-1}(S_b - S_w)$ are calculated and then combined into a matrix to obtain the optimal transformation matrix A^* [41]. Finally, through Eq. (8) dimensionality, the reduced feature vector is obtained. This simplifies computations by transforming high-dimensional data into a lower-dimensional space, thereby enhancing computational efficiency [42].

$$y_k = x_k A^* \tag{8}$$

y_k is the equivalent feature vector and x_k is the actual vector that must be reduced. The final dimensionality-reduced feature matrix is represented as Y^{NxM} represents the transformation of the overall N samples into a new dataset. Overall, N sample is transformed to form a new dataset with $D' = \{y_k, t_k\}$ features. The application of Linear Discriminant Analysis (LDA) contributes to the mitigation of overfitting and underfitting by emphasizing class separability and capturing the underlying structure of the data. This intrinsic focus aids in the development of a more robust and generalizable model.

Logistic regression

It is a probabilistic classifier method to calculate categorial results by considering independent variables. Weighted matrix w and a bias b are categorized using this probabilistic classifier [43]. It is expressed as

$$t = sgn\left(w^T y + b\right) \tag{9}$$

The augmented weight matrix for the classifier can be obtained during the training phase of the LR model, and it is expressed as

$$t = sgn\left(\theta^T y\right) \tag{10}$$

$D' = \{y_k, t_k\}$ is the feature-reduced dataset passed to the LR model with the target output t_k and y_k is training data. The weight of the matrix is initialized as $\theta = 1$, and it is expressed as

$$\theta_j(n) = \theta_j(n-1) + \alpha.\vartheta_j \tag{11}$$

α is the learning rate and ϑ_j is expressed as

$$\vartheta_j = \sum_{i=1}^{m} \left(t^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j^{(i)} \tag{12}$$

where m is the number of samples available in the existing feature reduce dataset, $j \in \{1, 2, \ldots, m\}$ and $h_\theta(x)$ is the logistic function given as $h_\theta(x) = \frac{1}{(1+ e^{-\theta^T x})}$.

The average cost function J(θ) for the logistic regression model is expressed as

$$J(\theta) = (1/m)\sum_{i=1}^{m}\left(Cost\left(h_\theta\left(y^{(i)}\right), t^{(i)}\right)\right) \tag{13}$$

and $Cost(h_\theta(y^{(i)}), t^{(i)})$ is expressed as

$$Cost\left(h_\theta\left(y^{(i)}\right), t^{(i)}\right) = \begin{cases} -log\left(h_\theta(y)\right) & if\ t = 1 \\ -\log\left(1 - h_\theta(y)\right) & if\ t = 0 \end{cases} \tag{14}$$

To obtain the minimum average cost for the logistic regression model, an iterative expression for $J(\theta)$ is employed, and it is expressed as

$$J(\theta) = J(\theta) + \left(-\frac{1}{m}\right)\left(\sum_{i=1}^{m} t^{(i)} log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - t^{(i)}\right) log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right) \tag{15}$$

The iterative equation is terminated when optimum weights are obtained, and the model is ready for testing. In other words, ε denotes the acceptable threshold for the function, then (15) terminates when $|J(\theta)| \leq \varepsilon$

Algorithm1 hybrid LDA-LR

Input: train dataset $D = \{x_k, t_k\}\ k = 1, 2, 3 \ldots N$ test dataset; $= \{Tx_k, Tt_k\}\ k = 1, 2, 3 \ldots n_t$.
Output: projected classification matrix T

1. Create a feature matrix X for D.
2. $X = Z - score(X)$ – (1)
3. Compute S_b and S_w – (4) and (6).
4. Obtain A^* by solving – (7).
5. Determine $Y = XA^*$, obtain the new train data $D = \{y_k, t_k\}$.
6. Set the following variables: Cost Function, ε, number of iterations N_{max}, Initialize augmented weighted matrix, $\theta = 1$; and Cost function $J(\theta) = 0$.
7. Update the augmented weighted matrix, θ using (11).
8. Find the cost function using (15).
9. If $|J(\theta)| \leq \varepsilon\ (or)\ N = N_{max}$, go to Step 10.
10. Else go to Step 7 to update the augmented weighted matrix.
11. Optimum weights are obtained for θ.
12. Testing: Find the output using optimum weights and test inputs $t = sgn(w^T y)$.
13. Return T.

Time complexity analysis

The complexity of algorithm 1 is evaluated by considering all the steps with their respective iterations, where N means the number of executions, d means the tasks, and c means the class type. The training times of the following equations are considered in the algorithm:

1. To compute normalization in (1) - for each of the c types, there are Nd operations and 1 division. Thus, in total, there are $Nd + c$ operations.
2. Compute S_b in (4) - $N(d + d^2 + d^2)$, the first d is for $f_{ij}n_i n_j$, the second d^2 is for $(\mu_i - \mu_j)$ and the third d^2 is for $(\mu_i - \mu_j)^T$.
3. Compute S_w in (6) - $N(d + d^2)$, the first d is for $(x_{ij} - u_i)$ and the second d^2 is for $(x_{ij} - u_i)^T$.
4. Compute A^* - $d^2 + d^2 + d^2$, the first d^2 is for $(S_b - S_w)$, the second d^2 is for $A^T(S_b - S_w)\ A$ and the third d^2 is for the division of with projection, Identity matrix, and Transpose.

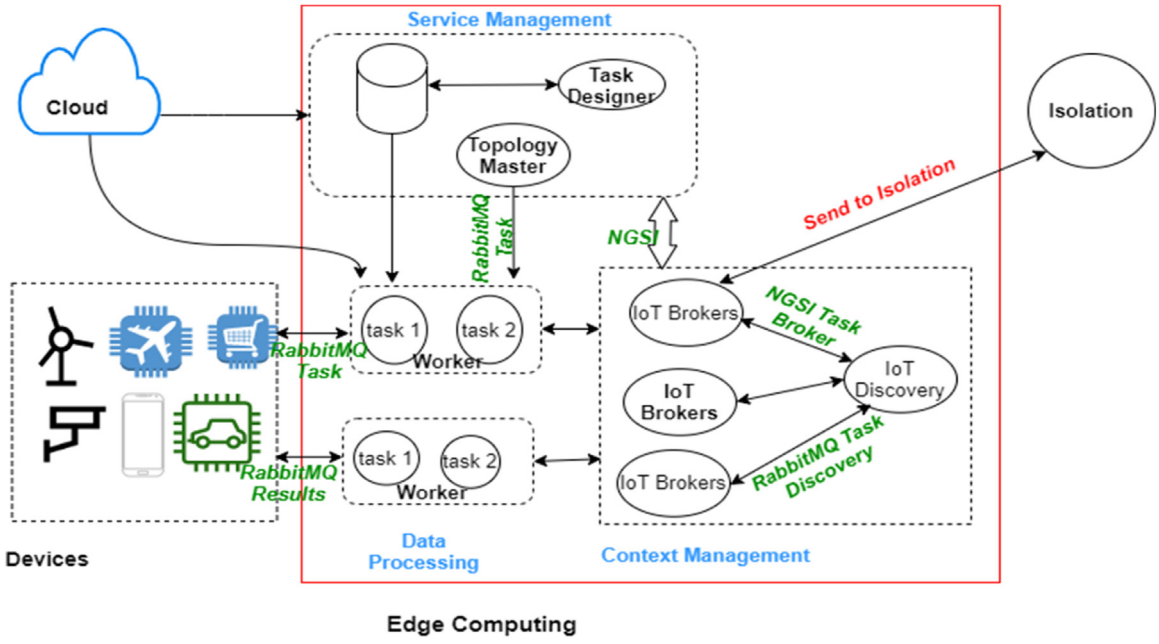


Fig. 1. System Model of the proposed framework.

For LDA, we have $N > d > C$ thus, the dominated terms are Nd^2 the computational complexity is $O(d^2)$. For LR, the training time complexity is solving the optimization problem.

5. Update the augmented weighted in (11) matrix $\theta_j(n) = \theta_j(n-1) + \alpha.\vartheta_j$ is d
6. Find the cost function in (15) $J(\theta)$ for N iterations which is Nd

For LR, solving the optimization problem is $O(Nd)$. Overall, for the LDA-LR algorithm complexity is $O(d^2)$.

Method validation

The edge computing framework deployed in the proposed work is based on FogFlow, as shown in Fig. 1. It is a standard framework deployed in NEC Laboratory for IoT services over the cloud and edge nodes. It has common interfaces for sharing and reusing contextual data across services. Service management, data processing, and context management are the three logical divisions. These divisions include task designer, and topology master to monitor IoT services and service orchestration among edge nodes. The topology master assigns data processing tasks to edge nodes and workers. For context management, IoT discovery and a set of IoT brokers are assigned. These components manage contextual data such as worker availability, topology, task, and generated data stream, as well as establishing data flow across tasks. The communication is usually through the RabbitMQ protocol [44]. The simulated real network environment of the FogFlow framework used in the proposed work is shown in Fig. 1.

The proposed system model is used to control the malicious IoT devices and data passed by the attacker. To achieve this, two separate virtual networks were deployed. The first production network includes the cloud, edge layer, and devices. The second virtual network is the isolation network, which is entirely separated from the production network. The optimal strategy is applied to prevent the attackers from gaining control of the production network. Practically, the strategies applied are MAC address starting from –B5:A2, specifying Operating system (Ubuntu), specifying the physical environment, and security patches in their OS with 17.2 or above. The devices that do not meet this requirement are denied access to the production network and automatically sent to the isolation network. In our case, the worker nodes evaluate the packets sent by the IoT devices through hybrid LDA-LR IDS. If the workers predict the data is generated from the attacker, the IoT broker isolates the IoT devices by immediately placing them in the isolation network. Hence the traffic generated from those malicious devices is isolated from the production environment.

Experimental setup

The proposed intrusion detection model is implemented on Python-based simulator YAFS (Yet Another Fog Simulator). It models mobility, sensors, and actuator and supports dynamic changes during execution using JSON files. This simulator is ideal for modeling network failures, dynamic allocation of modules, service placement problems, and designing robust networks [45]. The simulation setup includes a cloud node of 10GB RAM and 16 GHz CPU, 4 edge nodes of 2GB RAM and 3 GHz CPU, and 50–400 IoT devices of 500 MB RAM and 1 GHz CPU power. The link bandwidth is 3–10 Mbits, a packet size of 200 bytes, and $100\ X\ 10^8$ packets per instruction.

Table 2
Performance Metrics of the proposed model.

Metrics	Value
Training Time	0.04 s
Accuracy	96.56%
Precision	95.78%
Recall	92.69%
F-measure	94.21%
Isolation Accuracy	93.36%

NSL KDD dataset

KDD CUP and NSL KDD are the datasets used widely in intrusion detection. The disadvantages of KDDCUP data, such as huge, replicated records, are addressed in NSL KDD and hence it has gained popularity in many intrusion detection systems. In the NSL-KDD dataset, there are four different attack types: DoS, Probe, U2R, and R2L, and 21 different types of attack examples. More importantly, the testing dataset contains attacks that do not exist in the training set [46]. This significantly helps to check the training model competence effectively when facing unknown attack types. Therefore, the proposed work is developed using the NSL-KDD dataset, a public dataset from the Canadian Institute for Cybersecurity. Recent works on intrusion detection in edge computing environments are evaluated using this dataset [21,33,47].

Evaluation metric

The performance of the proposed machine learning-based IDS is evaluated using a confusion matrix. The performance in terms of training time, accuracy, precision, recall, and f-measure was evaluated using the confusion matrix. The overall performance of the proposed model is tabulated in Table 2 and the following are the definitions of these metrics:

Training Time: The total time required to train the model is measured in seconds called training time (s).

Accuracy: The percentage of test cases correctly identified is the accuracy of a procedure on a particular test set. It is expressed as

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

Precision: The ratio of all positive labeled instances to the notion of the positive cases that the model accurately recognizes is known as precision. It is expressed as

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

Recall: The proposition of the positive cases that the model correctly recognizes is called recall. It is expressed as

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

F-measure: A harmonic mean of precision and recall is used to calculate the F-measure and test accuracy. It is expressed as

$$\frac{2X\ Precision\ X\ Recall}{Precision+Recall}$$

Isolation Accuracy: The percentage of the number of devices isolated from the edge network. The devices predicted as an intrusion are isolated without notifying them or alarming the neighboring edge nodes and moved to the isolation zone. Isolation accuracy is expressed as

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

The isolation of the intruded devices limits their accessibility in the network and avoids their interactions with the other edge nodes. This significantly reduces the intensity of the attack and secures the edge resources from the attackers. Later, the isolated devices can be further analyzed to differentiate the TP and FP classifications and pass the FP devices back to the edge network to regain services.

TP (True Positive): The number of cases that have been correctly classified as an attack.
FP (False Positive): The number of cases that have been incorrectly classified as an attack.
TN (True Negative): The number of normal cases correctly classified.
FN (False Negative): The number of normal cases incorrectly classified.

Choosing a suitable machine learning algorithm

Machine Learning classifiers in IDS predict the modeling problem for the given input and classify it as intrusions or normal. There are many classification algorithms for predictive modeling problems, but there is no good concept to map the algorithm into the

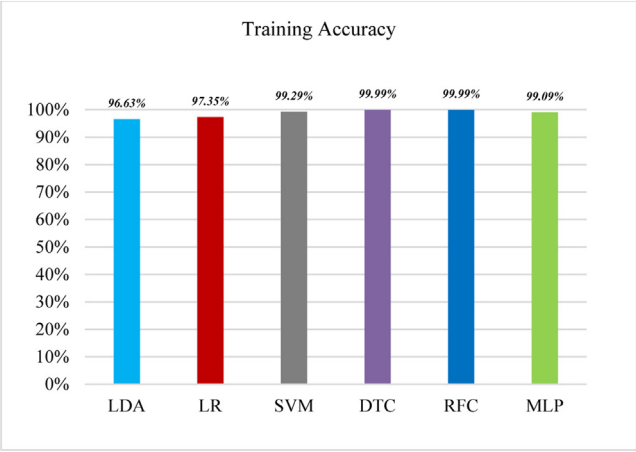


Fig. 2. Accuracy of ML classifiers.

Table 3
Training Time of ML classifiers.

ML Models	Training Time(s)
LDA	2.5448
LR	2.7746
SVM	193.2697
DTC	1.7338
RFC	12.609
MLP	1.2025

Table 4
Training time comparison with evaluated ML methods.

Iterations	LR (s)	SVM (s)	MLP (s)	LDA-LR (s)	LDA-SVM (s)	LDA-MLP (s)
20	0.6723	1.2995	19.0033	0.1254	0.197	12.6315
40	1.2608	2.3372	39.4815	0.0963	0.2666	28.8201
60	1.8019	3.3221	58.1655	0.1038	0.3613	44.537
80	2.5627	4.3194	78.8476	0.0888	0.4792	38.1013
100	3.285	5.4656	98.3079	0.1038	0.588	80.7532

problem in practice. Therefore, the controlled experiments are performed in the proposed work to discover an appropriate algorithm, and their configuration results in the best performance for a given task. The ML classifiers considered for the experiments are Linear Discriminant Analysis (LDA), Logistic Regression (LR), Support Vector Machine (SNM), Decision Tree Classifier (DTC), Random Forest Classifier (RFC), Multi-Layer Perceptron (MLP).

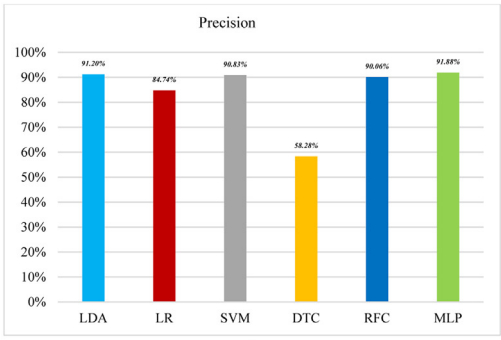
Table 3 shows the training time, and it can be observed that MLP and DTC have efficient training times compared to the other classifiers. Likewise, Fig. 2. shows the accuracy of these classifiers, which is above 95% for all the classifiers. This shows that all the considered classifiers have an efficient classification accuracy.

Further test precision, recall, F-measure, and isolation accuracy are analyzed to evaluate the efficiency of the classification. Fig. 3(a)–(d) show the evaluation of the performance metrics. In the evaluation process, precision is a good measure to determine the false positive instances. This determination confirms that non-intruders are not falsely identified as an intruder. Hence from Fig. 3(a) and (b), other classifiers have significant precision and recall except DTC.

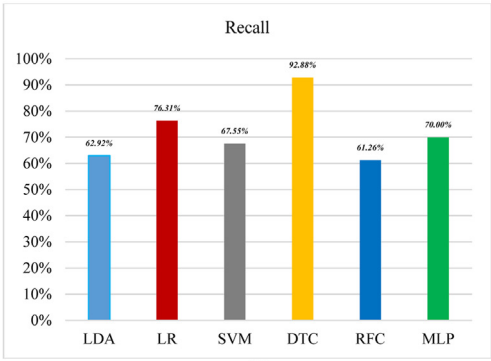
Significantly Fig. 3(c) shows the LDA, LR, and MLP have a harmonic mean balance between precision and recall. Finally, the main aim of the proposed work is to isolate the identified intrusions. Therefore, considering these metrics, the classifiers are further evaluated for isolation accuracy. Fig. 3(d) shows the isolation accuracy; apart from DTC and RFC, the other classifiers have above 75% accuracy. This illustrates that true positive intrusions are successfully isolated.

Further, to identify the hybrid combination of the classifiers, LR, SVM, and MLP are combined with LDA and evaluate their performance. Since LDA has significantly efficient metrics in precision, recall, and isolation accuracy, it is considered the main combination with the other classifiers.

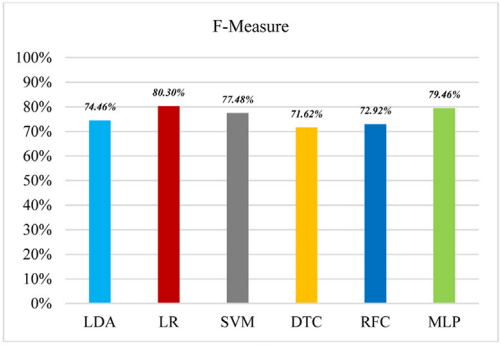
To evaluate the hybrid sequence of the classifiers, the model was evaluated with various number of iterations to assess training time and isolation accuracy, and they are depicted in Tables 4 and 5. In Table 3. the training time of SVM can be observed to be high, but the combination of LDA-SVM delivers an efficient time likewise LDA-LR. However, the accuracy of LDA-SVM is not constant,



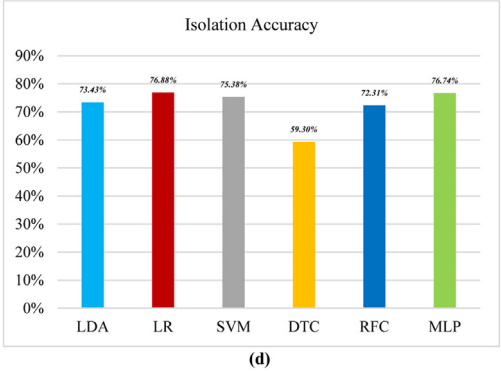
(a)



(b)



(c)



(d)

Fig. 3. (a): Precision of ML classifiers (b): Recall of ML classifiers (c): F-Measure of ML classifiers (d): Isolation Accuracy of ML classifiers.

Table 5
Isolation accuracy comparison with evaluated ML methods.

Iterations	LR%	SVM%	MLP%	LDA-LR%	LDA-SVM%	LDA-MLP%
20	78.79	16.13	78.59	93.54	37.67	91.56
40	79.02	23.05	79.21	93.54	38.76	91.55
60	79.16	34.64	77.90	93.54	26.35	91.39
80	79.09	49.85	77.39	93.54	19.19	91.37
100	78.69	36.01	77.00	93.54	70.40	91.98

Table 6
Comparison with existing IDS.

Refs.	Models	Training Time (s)	Training Accuracy	Test Accuracy	Precision	Recall	F-Measure
[23]	ELM – IDS	4.52	–	99.07%	–	–	–
[19]	MLP- IDS	1.2025	99.09%	79.40%	91.88%	70%	79.46%
[28]	Autoencoder and isolation forest	–	–	95.4%	94.81%	97.25%	96.01%
	Proposed model	0.04	96.56%	93.51%	95.78%	92.69%	94.21%

while LDA-LR remains the same irrespective of the iterations. It can be observed that the isolation accuracy, training accuracy, and training time of 93.54%, 96.56%, and 0.04 s, respectively, and LDA-LR, are efficient compared to the other existing methods. Hence, the proposed model considers LDA-LR for intrusion detection and isolation.

Comparison with the other state-of-the-art methods

The identification of intrusions in the proposed model is based on the LDA-LR hybrid model. The efficiency of the proposed model is analyzed by comparing it with the existing MLP-based IDS, ELM-based IDS, and autoencoder and isolation-based model [24,28,33]

The comparison of the proposed IDS with the currently available systems is tabulated in Table 6. The proposed IDS has a better training time of 0.04 s. However, the sample selected ELM-based IDS has an efficient testing accuracy of 99.07% due to the efficient generalization of the ELM algorithm [28]. Similarly, MLP-based IDS has a better training accuracy of 99.09% because they generalize the classification of unknown patterns with known patterns. But their test accuracy is 79.4%, and precision is 91.88% [24]. Above all these methods, Autoencoder and isolation forest-based IDS use a similar NSL-KDD dataset and have an accuracy of 95% and a precision of 94.81% [33]. However, the proposed LDA-LR-based IDS and isolation method have 96.56% test accuracy and 95.78% of precision. Precision quantifies the number of positive predictions that belong to positive classes. In an IDS, predicting the positive instance is more important to guarantee the system's detection capability. Therefore, it can be determined that the proposed model has more efficient training time and precision compared to the current IDS methods.

Critical reflection

The proposed LDA-LR hybrid ML model for intrusion detection has an accuracy of 96.56% with a precision of 95.78%, which is significantly more efficient than the other conventional methods. This demonstrates that the intrusion detection of the anomaly is detected accurately with all the positive labels. The recall accuracy obtained for the proposed model is 92.69%, whereas the existing lightweight MLP-based IDS is 70% for the similar four edge nodes environmental setup [24]. This shows that LDA-LR has significantly better recall accuracy. Further, 94.21% of F-measure confirms that there is a balance between recall and precision. The main aim of the proposed article is to isolate the intrusion device and data from the other edge nodes in the edge layer. Therefore, the proposed model is assessed for isolating the devices and data. To determine the efficiency of the isolation, the proposed model is compared with other conventional methods and its hybrid combination. Currently, there are no methods that adopt isolation in the edge computing framework, and it can be considered a novel contribution from this research. From Figs. 3(a) and 3 (b), it can be observed that DTC isolates 7725 data packets and 135 IoT devices, which is a higher number of isolations compared to other methods. However the isolation accuracy in Fig. 3(c) shows that DTC is 59.30%, whereas the proposed LDA-LR method is 93.36% accurate by isolating 5774 data packets and 101 IoT devices. Overall, this LDA-LR model, compared with other hybrid algorithms, exhibits an average training time of 1.1036s and isolation accuracy of 93.54% for varied iterations. The time complexity of the proposed algorithm is $O(d^2)$, which shows that the algorithm has a quadratic running time. As the input data stream grows, the running time of the algorithm also grows. However, the limitation of the proposed work would be the time complexity of the algorithm. This will be further enhanced in future work. Nevertheless, it proves that the proposed model is significantly suitable for isolating intrusive data and devices, thereby accomplishing better isolation.

Conclusions

The security and privacy issues are inherited from the cloud to edge computing along with the new issues. To support edge computing applications, an accurate and rapid intrusion detection solution is required. The proposed LDA-LR model has efficient

training time and accuracy in intrusion detection. The preprocessing of data at the initial stage has significantly supported in reducing the training time. LDA provides a better classification to decrease the data features when the variables are simulated from a normal distribution. Further, LR classifies the covariates depending on their correlation with the feature-reduced variable. Thereby, the combination of LDA-LR supports robustness towards categorization and improves accuracy. The categorized data and devices are isolated from the other edge nodes and devices at a faster rate. This process ensures that intruders have limited time to stay in the edge network. Thereby it reduces the chances of the intruder understanding the workflow of the framework or gaining control of other nodes or networks. In addition, we have compared our work with state-of-the-art conventional methods and existing MLP-based IDS. Experiment findings show that the proposed approach outperformed both training time and accuracy compared to the existing methods. In future work, the proposed model could be used to investigate its applicability and performance across additional datasets to increase the credibility and generalization of the findings. In addition, other machine learning algorithms could be employed to improve accuracy and assess if IoT devices are wrongly considered as an intruder or true cyber criminals. This property makes it especially useful for evaluating the robustness and generalization capabilities of intrusion detection models.

Ethics statements

All authors declare that this work complies with ethical guidelines set by MethodsX

Declaration of competing interest

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

CRediT authorship contribution statement

Poornima Mahadevappa: Software, Validation, Visualization, Investigation, Writing – original draft, Conceptualization, Methodology. **Raja Kumar Murugesan:** Supervision, Conceptualization, Methodology. **Redhwan Al-amri:** Software, Validation, Visualization, Investigation, Writing – original draft. **Reema Thabit:** Writing – review & editing. **Abdullah Hussein Al-Ghushami:** Writing – review & editing. **Gamal Alkawsi:** Writing – review & editing, Software, Validation.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the Open Access funding provided by Qatar National Library.

References

[1] “statistic_id1259878_total-global-edge-enabled-iot-devices-2020-2030-by-market.pdf 2023”.
[2] P. Singh, P. Jishnu Jaykumar, A. Pankaj, R. Mitra, Edge-detect: edge-centric network intrusion detection using deep neural network, in: Proceedings of the IEEE 18th Annual Consumption Communication Network Conference CCNC 2021, 2021, pp. 1–6, doi:10.1109/CCNC49032.2021.9369469.
[3] I. Stojmenovic, S. Wen, X. Huang, H. Luan, An overview of Fog computing and its security issues, Concurr. Comput. Pract. Exp. 28 (10) (Jul. 2016) 2991–3005, doi:10.1002/cpe.3485.
[4] J.A. Perez-Diaz, I.A. Valdovinos, K.K.R. Choo, D. Zhu, A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning, IEEE Access 8 (2020) 155859–155872, doi:10.1109/ACCESS.2020.3019330.
[5] M. Mukherjee, et al., Security and privacy in fog computing: challenges, IEEE Access 5 (2017) 19293–19304, doi:10.1109/ACCESS.2017.2749422.
[6] O.A. Alzubi, J.A. Alzubi, M. Alazab, A. Alrabea, A. Awajan, I. Qiqieh, Optimized machine learning-based intrusion detection system for fog and edge computing environment, Electron 11 (19) (2022) 1–16, doi:10.3390/electronics11193007.
[7] S. Anwar, et al., From intrusion detection to an intrusion response system: fundamentals, requirements, and future directions, Algorithms 10 (2) (2017), doi:10.3390/a10020039.
[8] F. Haouari, R. Faraj, J.M. Alja’Am, Fog computing potentials, applications, and challenges, in: Proceedings of the International Conference Computation Applied ICCA, 2018, 2018, pp. 399–406, doi:10.1109/COMAPP.2018.8460182.
[9] X. Wang, Y. Han, V.C.M. Leung, D. Niyato, X. Yan, X. Chen, Convergence of edge computing and deep learning: a comprehensive survey, IEEE Commun. Surv. Tutorials 22 (2) (2020) 869–904, doi:10.1109/COMST.2020.2970550.
[10] B. Sharma, L. Sharma, C. Lal, Anomaly detection techniques using deep learning in IoT: a survey, in: Proceedings of the International Conference Computer Intelligence Knowledge Economy ICCIKE, 2019, pp. 146–149, doi:10.1109/ICCIKE47802.2019.9004362. 2019.
[11] A.D. Bellingegni, et al., NLR, MLP, SVM, and LDA: a comparative analysis on EMG data from people with trans-radial amputation, J. Neuroeng. Rehabil. 14 (1) (2017) 1–17, doi:10.1186/s12984-017-0290-6.
[12] M. Pohar Perme, M. Blas, S. Turk, Comparison of logistic regression and linear discriminant analysis : a simulation study, Metod. Zv. 1 (1) (2004) 143–161.
[13] S. Khan, S. Parkinson, Y. Qin, Fog computing security: a review of current applications and security solutions, J. Cloud Comput. 6 (1) (2017), doi:10.1186/s13677-017-0090-3.
[14] J. Ni, K. Zhang, X. Lin, X. Shen, Securing fog computing for internet of things applications: challenges and solutions, IEEE Commun. Surv. Tutorials 20 (1) (2018) 601–628, doi:10.1109/COMST.2017.2762345.
[15] A. Yousefpour, et al., All one needs to know about fog computing and related edge computing paradigms: a complete survey, J. Syst. Archit. 98 (December 2018) (2019) 289–330, doi:10.1016/j.sysarc.2019.02.009.
[16] K. A., Privacy and security problems in fog computing, Commun. Appl. Electron. 4 (6) (2016) 1–7, doi:10.5120/cae2016652088.
[17] D. Liu, Z. Yan, W. Ding, M. Atiquzzaman, A survey on secure data analytics in edge computing, IEEE Internet Things J 6 (3) (2019) 4946–4967 Jun, doi:10.1109/JIOT.2019.2897619.

[18] S. Prabavathy, K. Sundarakantham, S.M. Shalinie, Design of cognitive fog computing for intrusion detection in internet of things, J. Commun. Networks 20 (3) (2018) 291–298 Jun, doi:10.1109/JCN.2018.000041.
[19] S.J. Lee, et al., IMPACT: impersonation attack detection via edge computing using deep autoencoder and feature abstraction, IEEE Access 8 (2020) 65520–65529, doi:10.1109/ACCESS.2020.2985089.
[20] S. Xu, Y. Qian, R.Q. Hu, A semi-supervised learning approach for network anomaly detection in fog computing, IEEE Int. Conf. Commun. 2019-May (2019), doi:10.1109/ICC.2019.8761459.
[21] S. Das, D. Venugopal, S. Shiva, F.T. Sheldon, Empirical evaluation of the ensemble framework for feature selection in DDoS attack, in: Proceedings of the 7th IEEE IEEE International Conference on Cyber Security and Cloud Computing -IEEE CSCloud 2022, 2020, pp. 56–61, doi:10.1109/CSCloud-EdgeCom49738.2020.00019.
[22] Z. Liu, X. Yin, Y. Hu, CPSS LR-DDoS detection and defense in edge computing utilizing DCNN Q-learning, IEEE Access 8 (3) (2020) 42120–42130, doi:10.1109/ACCESS.2020.2976706.
[23] A.S. Sohal, R. Sandhu, S.K. Sood, V. Chang, A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments, Comput. Secur. 74 (2018) 340–354, doi:10.1016/j.cose.2017.08.016.
[24] B.S. Khater, A.W.B.A. Wahab, M.Y.I. Bin Idris, M.A. Hussain, A.A. Ibrahim, A lightweight perceptron-based intrusion detection system for fog computing, Appl. Sci. 9 (1) (2019), doi:10.3390/app9010178.
[25] X. Cao, Y. Fu, B. Chen, Packet-based intrusion detection using Bayesian topic models in mobile edge computing, Secur. Commun. Netw. 2020 (2020), doi:10.1155/2020/8860418.
[26] Q. Li, J. Hou, S. Meng, H. Long, GLIDE: a game theory and data-driven mimicking linkage intrusion detection for edge computing networks, Complexity 2020 (2020), doi:10.1155/2020/7136160.
[27] J. Jang-Jaccard, S. Nepal, A survey of emerging threats in cybersecurity, J. Comput. Syst. Sci. 80 (5) (2014) 973–993, doi:10.1016/j.jcss.2014.02.005.
[28] X. An, X. Zhou, X. Lü, F. Lin, L. Yang, Sample selected extreme learning machine based intrusion detection in fog computing and MEC, Wirel. Commun. Mob. Comput. 2018 (2018), doi:10.1155/2018/7472095.
[29] F. Lin, Y. Zhou, X. An, I. You, K.K.R. Choo, Fair resource allocation in an intrusion-detection system for edge computing: ensuring the security of internet of things devices, IEEE Consum. Electron. Mag. 7 (6) (2018) 45–50, doi:10.1109/MCE.2018.2851723.
[30] K. Khan, A. Mehmood, S. Khan, M.A. Khan, Z. Iqbal, W.K. Mashwani, A survey on intrusion detection and prevention in wireless ad-hoc networks, J. Syst. Archit. 105 (December 2019) (2020) 101701, doi:10.1016/j.sysarc.2019.101701.
[31] Y. Wang, W. Meng, W. Li, Z. Liu, Y. Liu, H. Xue, Adaptive machine learning-based alarm reduction via edge computing for distributed intrusion detection systems, Concurr. Comput. 31 (19) (2019) 1–12, doi:10.1002/cpe.5101.
[32] R. Vinayakumar, M. Alazab, K.P. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman, Deep learning approach for intelligent intrusion detection system, IEEE Access 7 (2019) 41525–41550, doi:10.1109/ACCESS.2019.2895334.
[33] K. Sadaf, J. Sultana, Intrusion detection based on autoencoder and isolation forest in fog computing, IEEE Access 8 (2020) 167059–167068, doi:10.1109/ACCESS.2020.3022855.
[34] Z.M. Al-Khuzaie, S.A.K. Albermany, M.A. AbdlNibe, Intrusion detection in the IoT-fog adopting the GRU and CNN: a deep learning-based approach, in: Proceedings of the Micro-Electronics and Telecommunication Engineering: Proceedings of 6th ICMETE 2022, Springer, 2023, pp. 379–389.
[35] Y. Labiod, A.A. Korba, N. Ghoualmi, Fog Computing-based intrusion detection architecture to protect IoT networks, Wirel. Pers. Commun. 125 (1) (2022) 231–259, doi:10.1007/s11277-022-09548-7.
[36] M. Alweshah, A. Hammouri, S. Alkhalaileh, O. Alzubi, Intrusion detection for the internet of things (IoT) based on the emperor penguin colony optimization algorithm, J. Ambient Intell. Humaniz. Comput. 14 (5) (2023) 6349–6366.
[37] K. Kalaivani, M. Chinnadurai, A hybrid deep learning intrusion detection model for fog computing environment, Intell. Autom. Soft Comput. 30 (1) (2021) 1–15, doi:10.32604/iasc.2021.017515.
[38] C.V. Zhou, C. Leckie, S. Karunasekera, A survey of coordinated attacks and collaborative intrusion detection, Comput. Secur. 29 (1) (2010) 124–140, doi:10.1016/j.cose.2009.06.008.
[39] W. Meng, E.W. Tischhauser, Q. Wang, Y. Wang, J. Han, When intrusion detection meets blockchain technology: a review, IEEE Access 6 (2018) 10179–10188, doi:10.1109/ACCESS.2018.2799854.
[40] U. Rl, “Lin e a r di s c ri mi n a n t a n aly sis : a d e t ail e d t u t o ri al linear discriminant analysis : a detailed tutorial,” pp. 0–22, 2017.
[41] D. Zheng, Z. Hong, N. Wang, P. Chen, An improved LDA-based ELM classification for intrusion detection algorithm in IoT application, Sensors 20 (6) (2020) 1706 Mar, doi:10.3390/s20061706.
[42] G. Karatas, O. Demir, O.K. Sahingoz, increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset, IEEE Access 8 (2020) 32150–32162, doi:10.1109/ACCESS.2020.2973219.
[43] P. Rao, J. Manikandan, Design and evaluation of logistic regression model for pattern recognition systems, in: Proceedings of the IEEE Annual India Conference INDICON 2016, 2017 no. 6, doi:10.1109/INDICON.2016.7839010.
[44] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, A. Kitazawa, FogFlow: easy programming of IoT services over cloud and edges for smart cities, IEEE Internet Things J 5 (2) (2018) 696–707 Apr, doi:10.1109/JIOT.2017.2747214.
[45] I. Lera, C. Guerrero, C. Juiz, YAFS: a simulator for IoT scenarios in fog computing, IEEE Access 7 (2019) 91745–91758, doi:10.1109/ACCESS.2019.2927895.
[46] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: Proceedings of the IEEE Symposium on Computational Intelligence in Cyber Security (IEEE CICS). CISDA, 2009, 2009, doi:10.1109/CISDA.2009.5356528.
[47] J. fu Cui, H. Xia, R. Zhang, B. xu Hu, X. guo Cheng, Optimization scheme for intrusion detection scheme GBDT in edge computing center, Comput. Commun. 168 (July 2020) (2021) 136–145, doi:10.1016/j.comcom.2020.12.007.