Wireless Personal Communications (2022) 125:231–259
https://doi.org/10.1007/s11277-022-09548-7

Fog Computing-Based Intrusion Detection Architecture to Protect IoT Networks

Yasmine Labiod[1] · Abdelaziz Amara Korba[1] · Nacira Ghoualmi[1]

Accepted: 7 January 2022 / Published online: 21 February 2022

Abstract
With the deployment of billions of Internet of Things (IoT) devices, more and more cyber attacks involving or even targeting such devices are rife. Cyberattack vectors are in constant evolvement in terms of diversity and complexity. Thus, to detect novel cyberattacks, we use anomaly-based techniques which model the expected behavior of the IoT device to identify occurrences of attacks. In this paper, we propose a new distributed and lightweight intrusion detection system (IDS). To provide efficient and accurate intrusion detection, the proposed IDS combines variational AutoEncoder and multilayer perceptron. The IDS operates within a two-layered fog architecture, an anomaly detector within fog node, and attack identification module within the cloud. The proposed approach is evaluated on two recent cyber attack datasets. The experimental results showed that the proposed system is able to characterize accurately the normal behavior within fog nodes, and detect different attack types such as DDoS attacks with high detection rate (99.98%) and low false alarms rate (less than 0.01%). The proposed system outperforms other existing techniques in terms of detection and false positive rates.

Keywords Internet of things · Fog computing · Intrusion detection · Security · Cyberattack · Anomaly detection · Deep neural network · Artificial neural network

1 Introduction

The Internet of Things (IoT) has been in the spotlight for the past decade. It is regarded as one of the innovations which has the potential to provide unlimited benefits to our society [1]. The development of IoT has enabled to heterogeneous devices to exchange, collect, store and share data with each other without human intervention. Nowadays, IoT devices

✉ Yasmine Labiod
labiod.yassmine@gmail.com

Abdelaziz Amara Korba
abdelaziz.amara.korba@univ-annaba.org

Nacira Ghoualmi
nacira.ghoualmi@univ-annaba.org

[1] Networks and Systems Laboratory, Badji Mokhtar Annaba University, Annaba, Algeria

are used in several and different fields of application such as smart home, smart grids, transportation, environment, infrastructure and public services, etc [2, 3].

1.1 Background

As the IoT is characterized by limited computations in terms of storage and processing power, it suffers from many issues such as reliability, security, privacy and performance [4]. The integration of the IoT with the cloud, known as the Cloud of Things (CoT), is the right way to overcome most of these issues [5]. The cloud computing is being recognized as a success factor for IoT, providing reliability, ubiquity, scalability in addition to the high-performance. However, because of its communication implications and geographically centralized nature, cloud computing based IoT fails in applications that require very low and predictable latency, computational power, lack of mobility support, which are geographically distributed, as well as large scale distributed control systems [6]. Fog computing with pervasive and cost-effective services is capable of providing a promising technology to tackle the low-latency, considerable computation resources and geographical distribution required by IoT devices [7]. Figure 1 shows how Fog computing paradigm extends cloud computing and its services to the edge of the network. The general structure of the fog computing is composed of three main layers: (i) End devices; (ii) Fog nodes; and (iii) Cloud infrastructure [8].

As shown in Fig. 1, IoT devices are organized into clusters, and each devices is connected to one of the fog nodes. Meanwhile, fog nodes could be interconnected with each other and are linked to the Cloud. The close proximity of the fog nodes to the end devices helps in resolving the latency problems and provides the option of reducing unnecessary multiple communication between the cloud computing center and the mobile users in addition to processing the data coming from the end devices in real time. The fog computing layer embodies software modules in the form of fog services and embedded operating

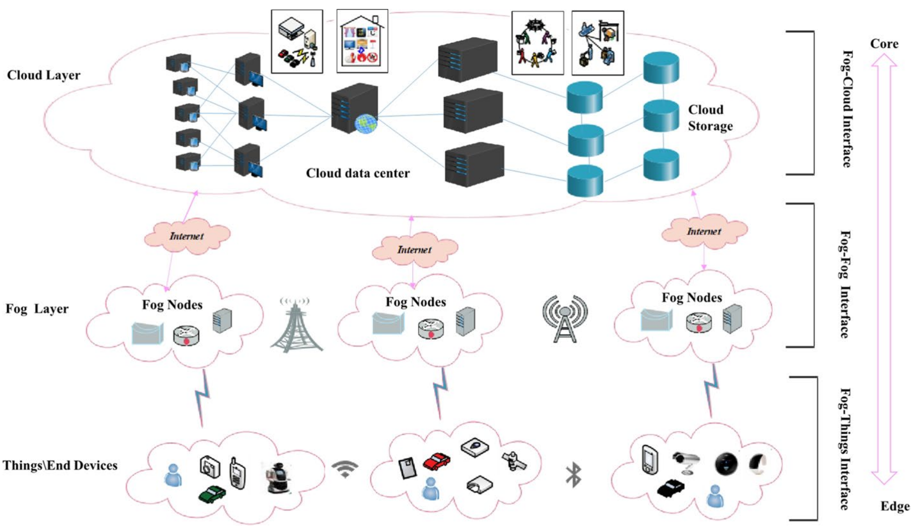

Fig. 1 The hierarchical architecture of fog computing

systems. It is also possible to analyze gathered data obtained from the sensor layer and thus make decisions locally [9].

Fog computing, as an emerging new technology, is facing many challenges related to security and privacy since fog devices are heterogeneous in nature and they are deployed in places where protection is minimal. Fog devices are vulnerable to many cyber-attacks such as Distributed Denial of Service (DDoS) attacks, rogue gateway attacks, man-inmiddle (MitM) attacks, privacy leakage, privilege escalation attacks, service manipulation attacks, and injection of information which compromise the data privacy. A lightweight network intrusion detection is a key technique to tackle this problem. This technique has been applied to fog computing to effectively reduce the latency and the security threats in the fog infrastructure.

Network intrusion detection system (NIDS) is one of several security mechanisms to manage security intrusions [10]. It monitors network traffic for abnormal or suspicious activity and issues alerts when such activity is discovered. Anomaly-based intrusion detection techniques include probabilistic-based detection, boundary-based detection, machine learning-based detection and deep learning-based detection.

Deep learning techniques have achieved a great success in many fields of artificial intelligence including language identification, computer vision, image processing and pharmaceutical research. Deep learning techniques have been applied for intrusion detection in fog architecture [11]. However, there are still many problems with fog computing intrusion detection systems, very high latency, a considerable computation resources in addition to low detection rate for unknown attacks and high false positive rate for minority attacks.

1.2 Research Motivations

The main research problem in this article is the lack of lightweight and robust intrusion detection techniques that can guarantee a secure and suitable environment for IoT based application. This article presents a lightweight intrusion detection architecture that operates in the fog-computing layer. As mentioned above, this layer has devices with more advanced features than the IoT device layer. Detecting the specific type of attack is extremely important for the countermeasures module to be able to carry out control measures and to inform the network manager of the vulnerability in question. However, existing techniques for multi class detection are not sufficiently accurate. This is precisely the focus of the proposed detection method. It has two steps and aims to classify the network traffic in specific types of attacks or normal behavior, for the execution of countermeasures. In the first stage, a binary classification method is applied to classify the network traffic as normal or malicious. For the second stage, a multi class classification method is applied to identify the attack type.

Our work makes a relevant contribution to state of the art in this regard. Most studies were based on two approaches. Some applied different machine learning algorithms and chose the one of them with the best performance [12]. Others sought to adjust the hyperparameters of a model until it achieved accuracy considered good [13, 14]. However, these classification techniques still have detection defects, low detection rate and high false positive rate for different attacks. In order to overcome these issues and improve the detection rate of wide range of attacks, this paper combines variational AutoEncoder (VAE) with a multi-layer perceptron (MLP) algorithm.

We propose a hybrid method of binary and multiclass classification with a high accuracy and precision rate to compose the first and second level of the proposed two-stage detection method. The first level detection method provides high rate of accuracy and precision, that is, that the most significant number of abnormal normal traffic attacks is classified as abnormal. At the second stage, it provides attack classification that helps to trigger the suited countermeasures. We conclude by analyzing the results to traditional machine learning approaches and state of the art approaches that the proposed method achieves this goal. The approach is capable of achieving an accuracy rate of 99.88%,for the IoT-23 dataset and 99.88%, for the IoT network intrusion dataset. Promising results were obtained in several subsets of data, demonstrating its efficacity. Although previous works have proposed methods based on deep neural networks VAE and Multi-layer perceptron (MLP) to detect intrusions, as far as we know, no work has combined these techniques in a hybrid binary classifier to protect an IoT environment.

The main contributions of this paper can be summarized as follows:

- A lightweight intrusion detection system was proposed for detecting a wide range of IoT attacks. The proposed IDS combines Variational AutoEncoder and multi-layer neural networks.
- A fog computing-based intrusion detection architecture to protect IoT networks is proposed. It reduces considerably the latency of the IDS by performing anomaly detection at a first stage. In a second stage, it provides attack classification that helps to determine precise prevention measures.
- A comprehensive analysis and comparison of the proposed IDS with existing machine learning classification techniques on two recent datasets including different attack types.

The rest of the paper is organized as follows. Section 2 presents related works. Section 3 presents a detailed description of the proposed approach. Section 4 presents the experimental details, obtained results and discussion. Section 5provides conclusion and, finally, Sect. 6 draws some lines for future works.

2 Related Work

Most of the existing works in the literature are related to threats and malicious fog node detection in fog-computing architecture.

Hosseinpour et al. [15] introduced a distributed and lightweight IDS based on an Artificial Immune System (AIS). The IDS is distributed in a three-layered IoT structure including the cloud, fog and edge layers. The AIS-IDS approach was tested on the KDD-Cup 99 dataset [16] and was proven to be efficient against low-frequent attacks, i.e., R2L and U2R attacks. In addition, they have tested their model on SSH Brute Force from ISCX dataset [17] . One major drawback of the work in lightweight IDS is using old datasets. On the other hand, IDS models that used modern datasets were designed for cloud platforms. An et al. [18] proposed a lightweight IDS named Sample Selected Extreme Learning Machine (SS-ELM) to overcome the space limitation of fog nodes. The KDD-Cup 99 dataset was used and SS-ELM was shown to outperform the classical back propagation algorithm in terms of detection accuracy and training time. Diro et al. [19] proposed a new distributed approach based on deep learning to detect intrusions into the IoT/Fog network environment. The authors uses a fog device as a master to perform the collaborative sharing and optimization of model parameters. This primary node can be considered as a Single Point of Failure (SPOF) that is easier to compromise than a cloud-based parameter update approach. The performance of the deep model is compared against traditional machine learning approach, and distributed attack detection is evaluated against the centralized detection system. The experimental results showed the effectiveness of the proposed model in detecting cyberattacks (high accuracy and recall). Peng et al. [20] proposed a lightweight IDS system based on decision tree for fog computing environment. The proposed IDS overcomes the limitation of the fog node. The KDD-cup 99 dataset was used to test the performance of the proposed IDS. The performance of the proposed IDS in terms of accuracy was 98.67% and 96.65% for normal and abnormal traffic respectively, which is better compared to the performance of the Naïve Bayes and KNN classifiers that compose it. In addition, the proposed IDS compared the detection time for each method on both binary and multiclass classification. One major drawback of the proposed system is its detection delay. An et al. [21] proposed a fog computing intrusion detection system framework (FC-IDS) using hyper graph clustering model. The FC-IDS can effectively describes the association between fog nodes under DDoS attacks. The experimental results show that FC-IDS can effectively detect DDoS attacks. Illy et al. [22] proposed a lightweight IDS for Fog-to-Things environment. The proposed solution employed diverse base learners using different known algorithms and built different ensemble classifiers for anomaly detection and attack classification. The experiments on NSL-KDD dataset [23]show that the IDS model is more suitable than other recently proposed intrusion detection systems in fog computing environment. The proposed system achieves high accuracy on binary and multiclass classification.

Pacheco et al. [24] proposed an artificial Neural Networks-Based Intrusion Detection System for Internet of Things Fog Nodes. The proposed approach detects compromised fog node, and then it takes the required actions to ensure communication availability. The experimental results showed that the proposed approach is able to detect anomalies, whether are related to system failures or cyberattacks. Khater et al. [7] proposed a lightweight intrusion detection system based on single hidden layer Multilayer Perceptron (MLP) model for Fog computing. The proposed IDS was designed to detect different cyber-attacks including Hydra-FTP, Hydra-SSH, Adduser, Java-Meterpreter,

Meter-preter, and Webshell attacks in the fog node layer. In order to make the IDS lightweight, authors used a feature extraction technique by modifying vector space representation via n-gram transformation. Sparse matrix is also applied to compress the matrix formatting. Furthermore, the linear correlation coefficient (LCC) is used to compensate the zero values, and mutual information feature selection to reduce the number of features. The proposed method was evaluated against the Australian Defense Force Academy Linux Dataset (ADFA-LD) and Australian Defense Force

Academy Windows Dataset (ADFA-WD) [25], which are new generation system calls datasets that contain exploits and attacks on various applications. The experimental results show that by using a single hidden layer and a small number of nodes, a low computational complexity for feature extraction and selection is achieved.

Most of existing IDSs are based on binary classification, while identifying the attack type is essential to trigger an adapted countermeasure. In addition, most of them have used the NSL-KDD and KDD-cup datasets, which are outdated and of very limited practical value for a modern IDS. In other hand, the above intrusion detection evaluation results are very encouraging but these classification techniques still have detection defects, low detection of unknown attacks, time overhead, high false positive rate for minority attacks and

Table 1 Related works on intrusion detection systems for Fog Computing

Year	Work	Machine/depp learning and data mining methods	B\M	Dataset used	Advantage	Limitation
2016	Hosseinpour et al. [15]	Artificial immune system (AIS)	B	KDD-Cup 99 and SSH Brute Force from ISCX dataset	Securing fog and cloud layer in fog computing	Training and testing samples is not mentioned
2017	An et al. [18]	Extreme learning machine	B	KDD-Cup 99	Building an efficient IDS	High processing cost.
2017	Diro et al. [19]	Depp neural network	B\M	NSL-KDD	Fog-based distributed detection scheme	Master node can be a SPOF, fog training can overload device.
2018	Peng et al. [20]	Decision tree	B	KDD-Cup 99	DT-based approach for light detection	High processing cost.
2018	An et al. [21]	Hyper graph clusteringmodel based on apriori algorithm	B	Own dataset.	Describe the association between fog nodes which are suffering from the threat of DDoS	Low number of features, the absence of most evaluation metrics.
2019	Illy et al. [22]	Ensemble learners	B	NSL-KDD dataset	Fog-based distributed detection scheme	Focused only on DDoS attack.
2019	Pacheco et al. [24]	Artificial neural network.	B	Own dataset.	secure the end devices in fog computing	Lack of Evaluation metrics.
2019	Khater et al. [7]	Perceptron-multi layer neural network	B	ADFA-LD and ADFA-WD Dataset	lightweight intrusion detection model for Fog computing	Single classification. Absence of Multi-Class classification. Low detection rate.
2019	Almiani et al. [26]	Deep recurrent neural network (RNN).	B	NSL-KDD dataset	Provide scalability and the Fog computin IDS is proved over different performance metrics.	Lower detection rate of impersonation Attacks. High false positive rate.
2019	Krzyszton, et al. [27]	Several machine learning techniques.	B	Own dataset.	Good performance and flexible.	No testing for other IoT datasets.
2020	Rahman et al. [28]	A single multi-layer perceptron.	B\M	AWID dataset	Fog-based Distributed detection scheme	The absence of most evaluation metrics that can further improve detection

Table 1 (continued)

Year	Work	Machine/depp learning and data mining methods	B\M	Dataset used	Advantage	Limitation
2020	Cristiano et al. [29]	DNN+ kNN.	B	NSL-KDD and CICIDS2017	Hybrid IDS in the fog, high accuracy and recall rate.	Training and testing samples is not mentioned Single classification.
Our approach		VAE + MLP	B\M	IoT-23 and IoT-network intrusion datasets.	Detection distributed in the fog, high accuracy and recall rate.	Processing performed by the MLP algorithm

B Binary classification, *M* multiclass classification latency issues. To overcome these shortcomings, we propose a fog computing-based intrusion detection architecture to monitor IoT networks and detect intrusions.

Table 1 summaries recent state-of-the-art IDSs for IoT and fog environments, with emphasis on approaches based on artificial intelligence techniques. It also presents the security issue that each one of these methods tries to address, along with the dataset that was used in order to evaluate their performances.

3 Proposed Approach

This section presents the architecture concept and design principles of our proposed approach. Figure 2 shows the general architecture of our detection system within fog to things environment. The IoT networks secured by the detection architecture takes advantage of the storage capacity and computing capability of fog nodes and cloud servers layers. Besides, the lightweight IDS architecture was designed to operate at three fog-computing layer namely: IoT nodes, fog nodes, and cloud layers. Each IoT device has a detection module situated in the fog node that analyses and classify the network traffic. The detection module operates at the foggy node layer without interaction with the cloud layer, thus avoiding latency. Each fog device is responsible for monitoring and securing its linked IoT network. All traffic on each IoT network is captured by its specific fog node, which operates in promiscuous mode. In order to have a reliable architecture, it is necessary to have different types of fog devices with lightweight IDS.

The fog layer is responsible for training the model and hosting the different types of modules, including feature preprocessing and detection modules.The coordinating fog node should be in place for collaborative parameter sharing. When a fog node receives a network traffic, it processes it on different modules including preprocessing module, detection module and countermeasure module. The data preprocessing module digitizes the strings in the given dataset and then it normalizes the whole data, to ensure the quality of the input data so as to improve the efficiency of detection. Detection module is taking charge of analyzing the attributes of the captured traffic and classifying them in legitimate or anomalous traffic. The detection module essentially requires very low latency in order to allow fast response for reducing the potential damage that an anomalous traffic can cause. We built an anomaly detection model using the VAE-MLP technique.

This model is deployed within fog nodes as a first level classifier. When an anomaly is detected, the countermeasure module raises an alert to the security administrator in order to take the suited measures. Then, the anomalous traffic is sent to the cloud for the second task. The countermeasure module is responsible for performing alerts, blocking actions and sending a report to security administration. Also, information about the detected traffic is sent to the cloud for the classification attacks and results summary module. Within cloud layer, once a traffic is detected as anomalous, the information from the given traffic are sent to the cloud for attack classification (denial of service, MitM and scanning attacks). The attack classification requires a more complex model, demanding more resources. Moreover, this task is less latency sensitive than the first one. For this task, we built an attack classification model based on MLP algorithm deployed in the cloud as a second level classifier. When the attack category is predicted, the information is sent to the security administrator in order to apply complementary and more precise prevention mechanisms (Fig. 3).

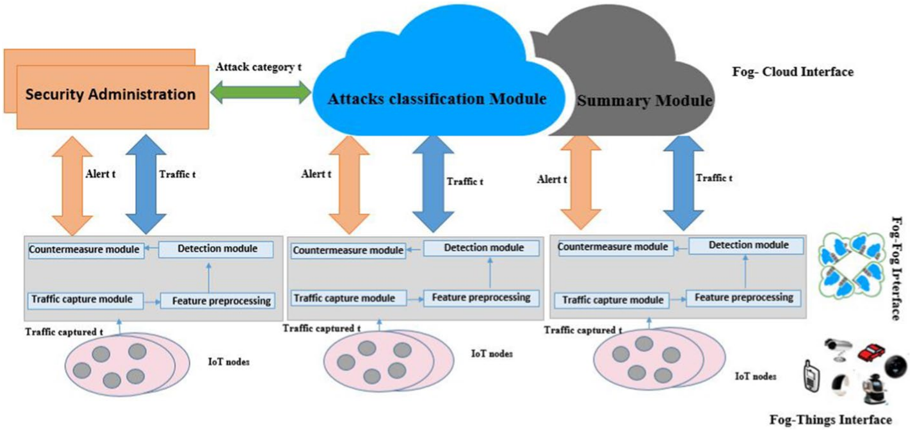

Fig. 2 Proposed approach architecture

3.1 Data Pre-Processing

We have used IoT-23 and IoT network intrusion datasets for the sake of model training, testing, and validation. Data features that represent input traffic of networking system are naturally inconsistent. Thus, traffic data preprocessing is a necessary gate for the classification engine. Traffic data preprocessing is a key step because it can reduce the experiment time and increase productivity. Traffic preprocessor engine applies two preprocessing steps on raw traffic data: (1) 1-to-N Numerical encoding, and (2) normalization.

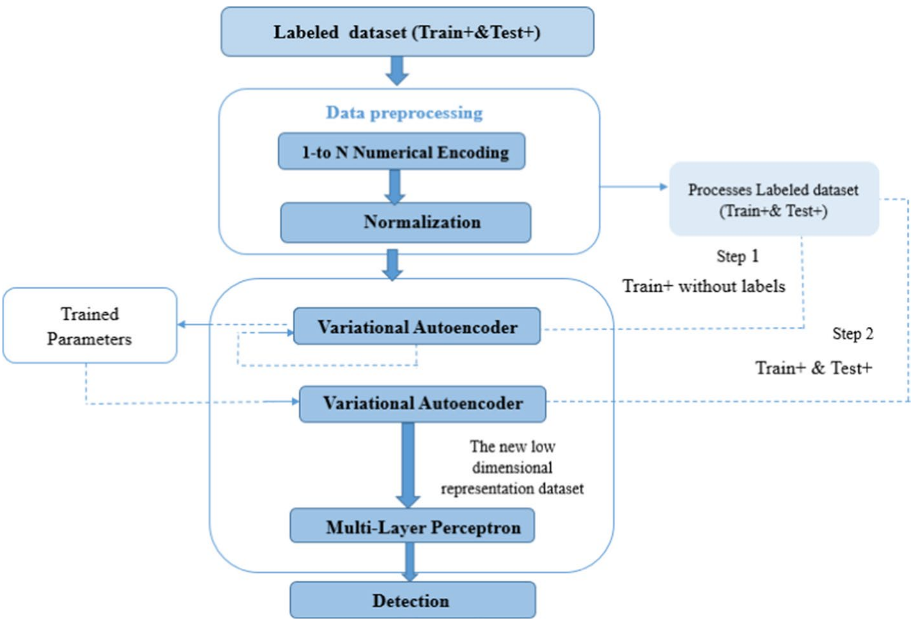

Fig. 3 Block diagram of the proposed lightweight IDS

1-to-N Numerical encoding: the detection module used in our approach cannot directly process the two datasets in their original format. Therefore, we use a 1-n encoding system to convert non-numeric features into numeric features. The IoT-network-intrusion-dataset has one non-numeric feature and 82 numeric features. Hence, we apply an encoding system to the non-numeric features, for instance “timestamp” feature has two distinct attributes namely, AM, and PM, and these can be encoded as (0, 0) (and 1, 0), in binary vectors, respectively.

Normalization: several features in both datasets have very large ranges between the maximum and minimum values, such as the difference between the maximum and minimum values in “flow duration” [0, 785673], where the minimum is 0 and the maximum value is 785673. This large difference also exists in other feature values, such as Stime (Record start time), drate (Destination-to-source packets per second), and srate feature (Total packets per second in transaction). Hence, these features are normalized by using max- min standardization data process based on calculating the mean absolute difference for mapping all feature values to the range [0, 1] according to Eq. 1.

$$X_i = \frac{xi - Min}{Max - Min}. \tag{1}$$

Where xi denotes each data point, Min denotes the minimum value from all data points, and Max denotes the maximum value from all data points for each feature.

3.2 Model Training Process

3.2.1 Overview

We propose the VAE-MLP method, a binary detection approach based on VAE and the MLP algorithm. At the first detection level, the proposed approach performs a binary classification, the traffic is either classified normal or anomalous, as shown in Fig. 4. Only traffic detected as anomalous by the first level, is then sent to the second detection level. The second level detection module operates within the cloud layer and classifies the anomalous traffic in a specific attack category. Besides identifying the attack type, the second detection level allows for correcting false positives of the first detection level.

The structure of VAE is composed of an encoder and a decoder. The distribution of the likelihood $p_\theta(x|z)$ decoder depends on the nature of the data. As we convert the two

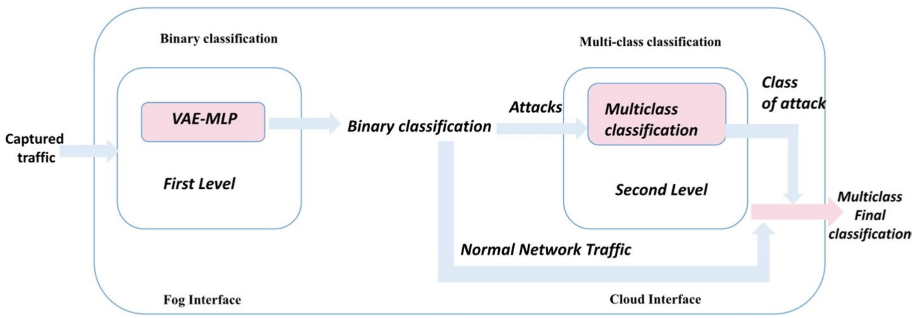

Fig. 4 Two-step classification method

datasets in binary nature, we use as encoder $q\phi(z|x(i))$, a multivariate Bernoulli distribution as the $q(z|x)$ distribution. Also for the decoder $p_\theta(x|z, Y)$, we use a multivariate Bernoulli distribution to fit $P(x|zY)$. For these parameters (θ, Φ), we use fully connected neural network to estimate them. Parameters (θ, ϕ) were updated using Adam algorithm [30]. The output of the decoder network is reconstructed data, which is the predicted probability. Finally, we use the gradient descent method called stochastic gradient variational Bayes or back propagation stochastic to train the proposed VAE [31]. However we calculate Monte Carlo methods [32] to optimize variational lower bound because it suffers from very high variance.

The MLP architecture contains a six layer feed forward deep neural network. The activation function of all hidden layers in MLP is ReLu6. The activation function of the output layer in MLP is sigmoid, which generates a rating value between 0 and 1 for each neuron. The softmax function transforms the outputs for each class to values between 0 and 1 and divides it by the sum of the outputs. This essentially gives the probability that the entry is in a particular class. The network structure of MLP hidden layers is exactly the same as that of Variational AutoEncoder. VAE can automatically extract high-level samples, so the trained parameters of VAE hidden layers is used to initialize the trained parameters of the MLP. Then the obtained training dataset is used to fine tune MLP classifier, and the Adam algorithm is used to optimize the MLP classifier. Finally, test features are introduced into the trained MLP classifier to detect attacks.

3.2.2 Problem Emulation

Given a labeled training set of m samples $\{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), \dots, (x_l^{(m)}, y^{(m)})\}$, where input feature vector $x_l(i) \in \mathbb{R}^n$ (The subscript "l" indicates that it is a labeled sample), $y(i) \in \{+1, -1\}$ are the corresponding labels for binary classification, $y(i) \in \{1, 2, \dots, N\}$ are corresponding labels for multiclass classification. Additionally, we assume there are m unlabeled samples $x_u^{(1)}, x_u^{(2)}, \dots, x_u^{(m)} \in \mathbb{R}^n$ produced by removing the labels from the labeled training set. For a better representation and less dimensionality of the input training set $x_l^{(1)}, x_l^{(2)}, \dots, x_l^{(m)} \in \mathbb{R}^n$, as in 6b.

- *Step 1*: we feed the unlabeled sample $x_u^{(1)}, x_u^{(2)}, \dots, x_u^{(m)} \in \mathbb{R}^n$ (IoT-23\IoT network intrusion Train+) to the Variational AutoEncoder algorithm. It can be used to reconstruct and learn the input training dataset $x_l^{(1)}, x_l^{(2)}, \dots, x_l^{(m)} \in \mathbb{R}^n$. After learning the optimal values for w, b, θ and Φ (trained parameter set in 6b) by applying Variational AutoEncoder on unlabeled data xu (IoT-23\IoT network intrusion Train+). As in 6b.
- *Step 2*: we feed $x_l^{(1)}, x_l^{(2)}, \dots, x_l^{(m)}$ (IoT-23\IoT network intrusion Train+ and IoT-23\IoT network intrusion Test+ dataset) as an input to a Variational AutoEncoder which attempts to reconstruct and learn its output values $\hat{x}_l^{(1)}, \hat{x}_l^{(2)}, \dots, \hat{x}_l^{(m)} \in \mathbb{R}^n$ to be equal to its inputs $x_l^{(1)}, x_l^{(2)}, \dots, x_l^{(m)} \in \mathbb{R}^n$ getting a new and good representation $\{(z_l^{(1)}, y^{(1)}),$ $(z_l^{(2)}, y^{(2)}), \dots, (z_l^{(m)}, y^{(m)})\}$, where the original input data is replaced with corresponding latent space vector as in Fig. 2. Thus, our training set becomes $\{(z_l^{(1)}, y^{(1)}), (z_l^{(2)}, y^{(2)}), \dots, (z_l^{(m)}, y^{(m)})\}$.
- *Step 3*: finally, we train the MLP classifier using the new training set to obtain a function that performs predictions of the intrusion on the y values. For the given testing set x test, we follow the same scenario for the training set: feeding it to Variational AutoEncoder to get z test. Then, we feed z test to the trained MLP classifier to get a prediction.

3.2.3 Combining VAE and MLP

Variational AutoEncoder provides good data representation because of its simple and straightforward implementation and its capability to learn the original expressions and structures of data. We believe Combining robust classifiers such as MLP and VAE provides high detection performances. Feature extraction and dimensionality reduction process in Variational AutoEncoder involves two steps: encoding and decoding. Input data is firstly projected to a stochastic distribution of the latent variable through encoder, then the latent variable is sampled from the distribution, and the decoder will reconstruct the input data based on the latent variable.

VAE applies back propagation algorithm to obtain the optimal values for its weight matrices $W \in R^{K\times N}$ and $V \in R^{N\times K}$ bias vectors $b_1 \in R^{K\times 1}$ and $b_2 \in R^{N\times 1}$, which attempts to learn and reconstruct its output values $(\widehat{xi})$ to be equal to its inputs xi. In other words, an approximation to the identity function is learned to make the output values similar to the input values; that is, it uses $y^{(i)} = x^{(i)}$. VAE uses also back propagation algorithm to optimize the parameters of decoder and encoder $\{\theta, \phi\}$. In other words, VAE applies back propagation algorithm to minimize the loss function, which is represented by Eq. 3.

$$\begin{aligned} \log\left(p_\theta(x)\right) = & \Big[D_{KL}\left(q\phi\left(z|x^{(i)}\right)\|p_\theta(z|x)\right) \\ & + L(\theta, \boldsymbol{\Phi}, x))\Big] \end{aligned} \tag{2}$$

$$\begin{aligned} L(\theta, \boldsymbol{\Phi}, x) = & - D_{KL}\left(q\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right) \\ & + Eq\phi\left(z|x^{(i)}\right)\left[\log p_\theta\left(x^{(i)}|z\right)\right] \end{aligned} \tag{3}$$

The VAE loss function is composed of a reconstruction loss and a KL loss. The KL divergence part is a similarity measure between two distributions: the approximate posterior distribution and the real posterior distribution. To estimate this maximum likelihood, VAE needs to maximize the evidence variational lower bound (ELBO) $L(x)$. To optimize the KLD between $q\phi(z|x)$ and $p_\theta(z)$, the encoder estimates the parameters vectors of the Gaussian distribution $q\phi(z|x)$: mean μ and standard derivation σ. There is an analytical expression for their KLD, because both $q\phi(z|x)$: and $p_\theta(z)$ are Gaussian. To optimize the second term of Eq. 2, VAE minimize the reconstruction errors between the input and outputs. Given a labeled training set of m samples $x \in \mathbb{R}^d$, the objective function can be defined as:

$$\begin{aligned} L_{VAE} &= L_{MSE}(x, G0, (z)) \\ L_{VAE} &= L_{MSE}(x, x_t) + \lambda L_{KLD}(\mu, \sigma) \end{aligned} \tag{4}$$

$$L_{MSE}(x, x_t) = \|x - x_t\|^2 \tag{5}$$

$$\begin{aligned} L_{KLD}(\mu, \sigma) &= KL(q\phi(z|x^{(i)})\|p_\theta(z)) \\ &= KL(q\phi(z;\mu, \sigma)\|N(z;0, I) \\ &= \int N\left(z;\mu, \sigma^2\right)\log\frac{N(z;\mu, \sigma)}{N(z;0, I)}dz \\ &= \frac{1}{2}\left(1 + \log\left(\sigma^2\right) - \mu^2 - \sigma^2\right) \end{aligned} \tag{6}$$

The first term $L_{MSE}(x, x_t)$ represents the mean squared error (MSE) between the input and their output (reconstruction) for all m input data. The second term $L_{KLD}(\mu, \sigma)$ regularizes the encoder by encouraging the approximate posterior $q\phi(z|x)$ to match the prior $p(z)$. To hold the tradeoff between these two targets, each KLD target term is multiplied by a scaling hyper parameter λ. KLD at train a zero value when, $q\phi(z|x)$ is equal to $p_\theta(x|z)$, that is $p_\theta\left(x^{(i)}|z\right) = q\phi(z|x)$. We will get the true posterior distribution (KL is minimized). After learning the optimal values for w and $b1$ by applying Variational AutoEncoder on unlabeled data xu, we evaluate the feature representation $a = z$ for labeled data (x_l, y). We use this new feature representation, z, with the label vector, y, in MLP for the classification task. The pseudocodes of the proposed detection approach VAE-MLP is provided in Algorithm 1

Algorithm 1: VAE-MLP

Input: Dataset: Dataset $x(i), \ldots, x(N)$, learning rate lr, L2 regularization β, the batch size b, and number of hidden layer L

Output: The final classification results.

for $(1 \in [1, L])$ **do**
- initialize $\theta_1 = 0, \acute{\theta}_1 = 0, \Phi_1 = 0, \acute{\Phi}_1 = 0$;
- define the l-th hidden layer representation vector $h_1 = [q_\Phi(x) = (q_\Phi(z|x^{(i)}))p_\theta(z)dz]$
- define the l-th hidden layer output $y_1 = [p_\theta(x) = \int(p_\theta(x^{(i)}|z))]$;
- **while** *(θ, Φ, β, have not converged)* **do**
  - compute h_1 from $h_{1\text{-}1}$;
  - Draw L sample from $\epsilon \sim N(0, \acute{\varepsilon}, I)$
  - $z(I, l) = h\Phi(\acute{\varepsilon}(i), x(i)) \quad , i = 1, \ldots, N$;
  - compute y_1 ;
  - compute the loss function ;
  - Update layer parameters
  - $\varepsilon_1 = (\theta_1, \Phi_1)$ and $\acute{\varepsilon}_1 = (\acute{\theta}_1, \acute{\Phi}_1)$;
- **end**

end

initialize $(\theta_{1+1}, \phi_{1+1})$ at the new variationnel autoencoder;

obtain the new low dimensional representation of the training dataset S ;

initialize the weight of the MLP hidden layers using the weights of the trained VAE ;

adjust the parameters of the output layer by back propagation, and the new training data set is used to fine tune MLP classifier ;

Test samples are fed into the trained MLP classifier to detect attacks ;

Return the classification result.

4 Experimental Result and Analysis

In this section, first, we describe the datasets used in the experiment. Then, we evaluate the performance of the proposed approach. Finally, we discuss results and provide a comparative study between our approach and existing techniques.

4.1 Description of Datasets

Currently, the most common data sets used to evaluate the performance of IoT network intrusion detection systems in the literature are IoT-23 dataset and IoT network intrusion dataset.

4.1.1 IoT-23 Dataset

The IoT-23 dataset [33] consists of twenty three captures of different IoT network traffic. These captures are divided into twenty network captures (labeled files) from infected IoT devices and three network captures of real IoT devices network traffic. The three real IoT devices are Philips HUE smart LED lamp, Amazon Echo home intelligent personal assistant, and a Somfy smart doorlock. The IoT network traffic was captured in the Stratosphere Laboratory, AIC group, FEL, CTU University, Czech Republic and it is published on January 2020. The dataset contains diverse types of malware's traffic such as: Mirai, Torii, Okiru, Trojan, Kenjiro, Gagfyt, Hakai, and IRC Bot botnet malware. And different types of attacks: distributed denial of service (DDoS), C&C, C&C-HeartBeat, C&C-File Download, file download, and port scan.

We split the dataset into two parts, Train+_20 Percent.txt (a 20% subset of the full training set), Test+.txt, and IoT-23 Test-21.txt (A subset of the full test set, excluding records of difficulty level 21). In our experiments, IoT-23 Train+_20Percent is used as a training set, and IoT-23 Test+ and IoT-23 Test-21 are used as test sets, which has different normal records and different types of attack records, as shown in Table 2. Each traffic record in the IoT-23 dataset contains 21 features, 1 class label and 1 class detailed-label.

4.1.2 IoT Network Intrusion Dataset

The IoT-network-intrusion-dataset [34] is a collection of various types of network attacks in IoT environment. It reflects real modern normal activities and incorporates both normal IoT-related and other network traffic, along with various types of attack traffic commonly used by botnets. This dataset consists of 42 raw network packet files (pcap) captured at different time points. The raw network packet file was captured using monitor mode of wireless network adapter. In addition, the wireless headers are removed by Aircrack-ng. All attack traffic except botnet traffic are the packets captured while simulating attacks using tools such as Nmap. Hence in the Mirai botnet category, the attack packets were generated on a laptop and then manipulated to make it appear as if it is originated from the IoT device.

From raw network traffic, we have extracted a set of network features using CICflowMeter Tool [35]. CICflowMeter Tool is a network traffic flow generator distributed by the Canadian Institute for Cybersecurity CIC to generate network traffic features. The IoT

Table 2 The class distribution detail of IoT-23 dataset

IoT malware	Training dataset		Testing dataset			
	IoT-23 dataset train+_20Percent		IoT-23 dataset test+		IoT-23 dataset test-21	
	Category	Count	Category	Count	Category	Count
	Benign	1153	Benign	385	Benign	385
CTU-IoT-Malware	C&C	4024	C&C	1341	C&C	1341
-Capture-34-1 (Mirai)	DDoS	8636	DDoS	2879	DDoS	2878
	Port scan	73	Port Scan	24	Port Scan	25
CTU-IoT-malware	Benign	1915	Benign	639	Benign	639
-Capture-20-1 (Torii)	C&C-Torii	10	C&C-Torii	3	C&C-Torii	3
CTU-IoT-malware	Benign	1963	Benign	668	Benign	668
-Capture-20-1 (Torii)	C&C-Torii	10	C&C-Torii	2	C&C-Torii	2
	Benign	2420	Benign	1000	Benign	1000
CTU-IoT-Malware	C&C	0	C&C	0	C&C	0
-Capture-42-1 (Troja)	C&C-FileDownload	2	C&C-FileDownload	1	C&C-FileDownload	-
	FileDownload	2	FileDownload	1	FileDownload	-
CTU-IoT-Malware	Benign	1485	Benign	495	Benign	496
-Capture-60-1 (Gagfyt)	C&C-HeartBeat	50	C&C-HeartBeat	25	C&C-HeartBeat	20
	DDoS	2147075	DDoS	715691	DDoS	715691
	Benign	1597	Benign	533	Benign	533
CTU-IoT-Malware	C&C-HeartBeat	9412	C&C-HeartBeat	3138	C&C-HeartBeat	3138
-Capture-36-1 (Okiru)	Okiru	8176046	Okiru	2725349	Okiru	2725349
	Okiru-Attack	2	Okiru-Attack	1	Okiru-Attack	-
CTU-IoT-Malware	Benign	1308	Benign	436	Benign	437
-Capture-8-1 (Hakai)	C&C	4933	C&C	1644	C&C	1645

Table 3 The class distribution detail of IoT network intrusion dataset

Category	Training dataset: IoT network intrusion dataset train		Testing dataset: IoT network intru-siondataset test	
	Attack	Count	Attack	Count
Normal	Normal	15440648	Normal	175628
Scanning	Host discovery	2 208	Host discovery	256
	Port scanning	18845	Port scanning	2094
	OS/version detection	1635	OS/version detection	182
Man in the	ARP spoofing	91696	ARP spoofing	10189
Middle (MITM)	SYN flooding	58181	SYN flooding	6465
Mirai botnet	Host discovery	606	Host discovery	67.3
	Telnet brute force	1732	Telnet brute force	192
	UDP flooding	854356	UDP flooding	94948
	ACK flooding	68069	ACK flooding	7563
	HTTP flooding	9417	HTTP flooding	1065

network intrusion dataset contains a total of 25, 400, 443 records with 83 features. The partition of the full dataset are divided into a training set and a test set according to the hierarchical sampling method, namely, IoT_network_intrusion_training-set.csv and IoT_network_intrusion_testing-set.csv. The training dataset consists of 175, 341 records whereas the testing dataset contains 82, 332 records. The IoT network intrusion dataset contains ten categories of traffic, one normal and nine malicious (port Sscanning OS/Version detection, ARP spoofing, host discoveryV1, etc). Table 3 shows in detail the class distribution of the IoT network intrusion dataset.

Table 4 Model and training information

Model and training information		
	Model Type	Multi-Layer Network
Deep learning4j parameters	Learning rate	0.0001
	L2	$1 \times 10 - 5$
	Optimization algorithm	Adam
	Total parameters	25, 400, 443
	Last update	$05 - 07\,23 : 57 : 39 - 2020$
	Total parameter updates	938
	Updates/sec	3, 47
	Examples/sec	107, 38
	Optimal network structures (IoT-23)	$200 - 160 - 80 - 40 - 20 - 10$
	Optimal network structures (IoT-network)	$160 - 160 - 40 - 20 - 10 - 05$

4.2 Performances Evaluation

The data transformation and model training were implemented using java programming language, Weka data mining tools [36], and Deeplearning4j tool [37], which is the first commercial-grade, open-source, distributed deep-learning library written for Java and Scala. The dependency management for the built project was based on Apache Maven [38]. Both Deeplearning4j (DL4J) and MLlib Apache Spark's libraries were integrated with IntelliJ IDEA IDE. The main model parameters and training information of the proposed system VAE-MLP are described in Table 4.

In deep learing4j the default learning rate of the Adam optimizer is 0.0001, a stochastic gradient descent is used with a momentum value of 0.9, and a low learning rate is fixed at 0.0001 to train the neural network. We use Grid search to find the optimal hyper parameters of the model. For each group of hyper parameters, 10-fold cross-validation is used to evaluate the model. We consider the following metrics to evaluate the model:

Accuracy: it estimates the ratio of the correctly recognized connection records to the total number of samples for a given test data set. If the accuracy is higher, the machine learning model is better ($Accuracy \in [0, 1]$). Accuracy serves as a good measure for the test data set that contains balanced classes and defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{7}$$

Precision: it estimates the ratio of the correctly identified attack connection records to the number of all identified attack connection records. If the Precision is higher, the machine learning model is better ($Precision \in [0, 1]$). Precision is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Recall: also known as detection rate or sensitivity. It estimates the ratio of correctly predicted attack cases to the actual size of the attack class. Recall is defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

F1-Score: also called as F1-Measure. It is the harmonic mean of Precision and Recall. If the F1-Score is higher, the machine learning model is better (F1-Score$\in [0, 1]$). Compared with the accuracy, F1-score is more suitable for evaluating the detection performance of imbalanced samples. It can be defined as follows:

$$\text{F1-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

The false positive rate (FPR): it estimates the ratio of the Normal connection records flagged as Attacks to the total number of Normal connection records. If the FPR is lower, the machine learning model is better ($FPR \in [0, 1]$). FPR is defined as follows:

$$FPR = \frac{FP}{TN + FP} \tag{11}$$

The true positive rate (TPR): It estimates the ratio of the correctly classified Attack connection records to the total number of Attack connection records. If the TPR is higher, the machine learning model is better ($TPR \in [0, 1]$). TPR is defined as follows:

$$TPR = \frac{TP}{TP + FN} \tag{12}$$

To evaluate the proposed system, we compare it with some well-known classifiers and some recent ones namely J48, KNN, Naïve Bayes, REP Tree, Random Forest, LIBSVM. In this comparative study we use the different metrics detailed in 4. 2. Table 5 summarizes the performance of IDS system compared to the other classifiers for different attacks and benign traffic on IoT network intrusion dataset. It shows that the proposed detection system gives the highest average true positive rate (TPR) with 98.324% and the highest accuracy for eleven attacks type namely Host Discovery with 99.828%, port scanning with 99, 778%, OS/Version detection with 100%, ARP spoofing with 97, 230%, SYN flooding 97, 132%, host discovery 98.841, Telnet brute force 95.833%, UDP flooding with 98.118%, ACK flooding with 99.342, and HTTP flooding with 99.112. Moreover, the proposed IDS is very close to the highest detection rate for two types of attacks namely host discovery with 99.828%, and port scanning with 99.778%. Overall, VAE-MLP provides the best performances on the different attack types. The VAE-MLP shows high detection rate and a low positive rate, with a 42% reduction in memory and CPU overhead. For devices where resources are restricted, the VAE-MLP approach presents clear advantage.

In order to assess the robustness of the proposed approach, we compare VAE-MLP with existing detection approaches, considering the IoT-23 dataset. Table 6 summarizes the performance of VAE-MLP compared to other classifiers regarding the detection of different attack and benign network traffics. The results shows that VAE-MLP shows the highest accuracy with 99.984% and the highest true positive rate (TPR) for fourteen attacks type namely C&C-Mirai with 99.91%, DDoS Mirai with 99.92%, Port Scan-Mirai with 98.99%, C&C Torii1with 100%, C&C-Torii2 with 99.98%, C&C-Troja with 99.99%, C&C-File Download -Troja 98.99%, File Download-Troja with 99.97%, C&C HeartBeat Gagfyt with 99.818%, Okiru 99.92%, Okiru-Attack with 99.99% and C&C-Hakai with 99.90%. Moreover, the proposed IDS system is very close to the highest detection rate for two types of attacks namely C&C-Torii1with 100%, DDoS-Gagfyt 100%, and C&C HeartBeat Okiru with 100%. For the rest of the attack types, VAE-MLP gives an average performance compared to the other models. Overall, VAE-MLP provides excellent detection performances for the different attack types. Figure 5 present the overall performance of the proposed IDS and other classifiers in terms of false alarm rate, VAE-MLP presents the lowest false alarm rate with 0, 00066 and 0, 00053 in IoT-23 dataset and IoT-network intrusion dataset, respectively.

Table 5 Comparison of detection performance for different classification methods on the IoT network intrusion dataset

Algorithms	Category	Accuracy	Recall	Precision	F1-score	TPR
Lib SVM	Normal	76.561 %	68,500%	70.001%	71.990%	56,333%
	Host discovery	97.990%	77,044%	87,330%	97,230%	74,230%
	Port scanning	44,230%	91,990%	98,600%	98,770%	77,000%
	OS/version detection	78,100%	99.718%	90.568%	99.718%	90.718%
	ARP spoofing	99.023%	99.118%	67.945%	55.918%	45.118%
	SYN flooding	70.001%	99.999%	99.934%	90.000%	66.999%
	Host discovery	97.990%	99.778%	70.777%	90.807%	90.778%
	Telnet brute force	81.234%	99.778%	61.700%	23.323%	74.778%
	UDP flooding	45.228%	45,234	81.230%	100%	56%
	ACK flooding	67.999%	97,230%	55,560%	77,103%	67,255%
	HTTP flooding	74,210%	98,600%	98,000%	88,760%	88,000%
REP tree	Normal	95.165%	87.999%	99.0348%	61.111%	98.00 %
	Host discovery	99.636%	99.909%	73.465%	30.425%	99.999%
	Port scanning	50.000%	100%	98.158%	99.778%	99000%
	OS/version detection	99.47%	99.778%	95.84%	98.655%	77.474%
	ARP spoofing	99.778%	99.999%	55.007%	100%	99.111%
	SYN flooding	100%	97,230%	73.469%	97,230%	97,230%
	Host discovery	97,230%	98.050%	51.407%	76.561%	99.727%
	Telnet brute force	98,600%	99.345%	67.600%	23.001%	79.000%
	UDP flooding	99.718%	99.332%	88.888%	59.971%	99.118%
	ACK flooding	99.118%	72.499%	99.999%	100%	79.999%
	HTTP flooding	99.999%	99.778%	99.800%	99.227%	19.221%
J48	Normal	83.333%	100%	77,880%	51.282%	70.200%
	Host discovery	100%	29.968%	95,200%	97,990%	97,230%
	Port scanning	77,880%	61.236%	83.333%	83.333%	94,441%
	OS/version detection	66,600%	99.718%	99.455%	99.718%	92.918%
	ARP spoofing	92.444%	93.172%	99.182%	90.99%	98.675%
	SYN flooding	98.117%	95.999%	99.999%	39.999%	67.000%
	Host discovery	99.999%	99.778%	19.778%	99.778%	20.665%
	Telnet brute force	99.000%	99.998%	64.778%	99.455%	72.083%
	UDP flooding	98.775%	100%	99.778%	100%	69.778%
	ACK flooding	100%	97,230%	87,230%	97,230%	97,230%
	HTTP flooding	97,230%	98,600%	58,400%	97.680%	99,333%
Random forest	Normal	99.00 %	77.756%	63.243%	99.110%	99.000%
	Host discovery	99.333%	92.650%	66.489%	23.999%	48.521%
	Port scanning	90.000%	70.060%	99.276%	99.187%	98.778%
	OS/version detection	77.000%	99.778%	99.778%	99.778%	99.778%
	ARP spoofing	67.571%	100%	100%	99.222%	100%
	SYN flooding	99.727%	86.599%	97,230%	97,230%	65.429%
	Host discovery	99.818%	98,600%	99.718	78,000%	88.537%
	Telnet brute force	32.500 %	61.837%	99.718%	99.798%	78.485%
	UDP flooding	83.333%	99.44 8%	92.865%	59.118%	91.212%
	ACK flooding	93.100%	99.999%	99.999%	99.999%	92.65%
	HTTP flooding	99.188%	99.888%	89.33%	100%	70.816%

Table 5 (continued)

Algorithms	Category	Accuracy	Recall	Precision	F1-score	TPR
Naïve Bayes	Normal	77.001%	89,990%	99,310	100%	75,000%
	Host discovery	100%	75,000%	97,444%	97,440%	77,990%
	Port scanning	23,000%	98,300%	92,330%	77,200%	68,333%
	OS/version detection	97,670%	77.718%	93.001%	69.1118%	89.145%
	ARP spoofing	99.998%	99.001 %	99.999%	46.666%	89.000%
	SYN flooding	46.666%	99.003%	99.999%	93.000%	91.999%
	Host discovery	97.009%	99.114%	23.777%	95.888%	96.004 %
	Telnet brute force	96.238%	99.333%	97,000%	99.222%	93.800%
	UDP flooding	95.118%	100%	77.001%	93,000%	91.000%
	ACK flooding	100%	93,000%	95,990%	97,999%	77,000%
	HTTP flooding	90,330%	76,100%	89,700%	90,699%	88,99 0%
KNN	Normal	76.461 %	68,500%	77.001%	77.090%	99,000%
	Host discovery	77.390%	67,044%	97,200%	97,230%	74,000%
	Port scanning	44,130%	99,000%	88,800%	98,770%	77,110%
	OS/version detection	78,100%	78.018%	60.500%	99.718%	81.008%
	ARP spoofing	99.999%	98.118%	77.443%	55.918%	55.188%
	SYN flooding	70.000%	99.999%	99.999%	90.000%	56.999%
	Host discovery	97.00%	92.00%	100%	90.807%	91.778%
	Telnet brute force	44.432%	91.333%	81.650%	23.323%	99.778%
	UDP flooding	100%	33,004	31.000%	100%	98,003%
	ACK flooding	87.999%	95,330%	70,020%	77,103%	77,255%
	HTTP flooding	81,210%	98,900%	98,000%	88,760%	78,000%
VAE MLP	Normal	98.355%	100%	99,068%	100%	98,23%
	Host discovery	99.828%	98,432%	99,007%	95,930%	97,280%
	Port scanning	99.778%	98,999%	95,612%	97,500%	100 %
	OS/version detection	100%	99.999%	94.790%	99.888%	100 %
	ARP spoofing	97,230%	99.118%	98.234%	99.118%	99.657%
	SYN flooding	97,132%	99.999%	99.999%	99.901 %	99.999%
	Host discovery	98.841%	97.483%	99.000%	90.756%	95.098%
	Telnet brute force	95.833%	98.700%	90.156%	97.348%	99.778%
	UDP flooding	98.118%	100%	100%	97,230%	99.118%
	ACK flooding	99.342%	100%	97,343%	98,230%	99.222 %
	HTTP flooding	99.112%	99.998%	97,250%	100%	98,403%

4.3 Discussion

We have assessed the processing overhead and the memory required for the operation of the detection method. For the experimental setup, we have used a computer with an Intel Core i7-7500U@ 3.40GHz with 8 GB RAM running Windows10, 64 bits. 6a and 6b represents the measurements regarding the memory allocated and CPU expenditure made

Table 6 Comparison of detection performance for different classification methods on the IoT-23 dataset

Algorithms	Category	Accuracy	Recall	Precision	F1-score	TPR
Lib SVM	Benign	75.355%	92.355%	99,068%	99,099%	78,230%
	C&C-Mirai	65.828%	98,432%	99,007%	95,930%	87,280%
	DDoS- Mirai	90.778%	98,999%	95,612%	97,500%	85.888%
	Port Scan - Mirai	100%	99.999%	94.790%	99.888%	100 %
	C&C-Torii1	97,132%	99.999%	99.999%	33.901 %	99.999%
	C&C-Torii2	95.833%	98.700%	90.156%	97.348%	79.778%
	C&C-Troja	99.100 %	85.888%	97,343%	28,230%	59.992 %
	C&C-FileDownload -Troja	99.112%	99.998%	97,250%	100%	98,403%
	FileDownload-Troja	98.355%	95.828%	95.828%	95.828%	95.828%
	C&C-HeartBeat-Gagfyt	75.828%	98,999%	95,612%	97,500%	100 %
	DDoS-Gagfyt	100%	99.999%	94.790%	99.888%	100 %
	C&C-HeartBeat-Okiru	77,132%	91.999%	99.999%	99.901 %	99.999%
	Okiru	98.002 %	23.483%	99.000%	53.483%	70.456%
	Okiru-Attack	95.345 %	98.700%	90.156%	97.348%	99.778%
	C&C-Hakai	99.666%	95.828%	97,343%	98,230%	99.000 %
Random forest	Benign	98.112%	98.112%	98.999 %	100%	98,403%
	C&C-Mirai	92.355%	90.968%	60.968%	90.968%	90.999%
	DDoS- Mirai	94.555%	98,432%	99,007%	95,930%	97,280%
	Port Scan - Mirai	99.778%	98,999%	95,612%	97,500%	90.968%
	C&C-Torii1	100%	99.999%	74.790%	99.888%	100 %
	C&C-Torii2	97,230%	93.333%	92.234%	99.118%	55.657%
	C&C-Troja	97,132%	99.999%	99.999%	99.901 %	99.999%
	C&C-FileDownload -Troja	98.841%	97.483%	19.000%	90.756%	75.098%
	FileDownload-Troja	95.833%	98.700%	90.156%	90.348%	99.778%
	C&C-HeartBeat-Gagfyt	98.118%	100%	100%	97,230%	89.888%
	DDoS-Gagfyt	99.342%	100%	97,343%	98,230%	92.222 %
	C&C-HeartBeat-Okiru	99.112%	99.998%	97,250%	100%	77,403%
	Okiru	98.355%	100%	99,068%	100%	98,553%
	Okiru-Attack	99.828%	98,432%	89,007%	95,930%	97,760%
	C&C-Hakai	99.778%	98,999%	85,612%	97,500%	100 %
KNN	Benign	88.888%	88.562%	99.002 %	100%	99,403%
	C&C-Mirai	97.999%	88.999%	70.999%	88.998%	89.999%
	DDoS- Mirai	98.075%	96,990%	99,009%	94,967%	97,239%
	Port Scan - Mirai	99.008%	94,999%	97,666%	97,666%	97.666%
	C&C-Torii1	100%	99.999%	44.999%	99.888%	100 %
	C&C-Torii2	96,230%	99.456%	99.999%	80.998%	60.777%
	C&C-Troja	95,332%	99.999%	95.999%	95.999 %	99.999%

Table 6 (continued)

Algorithms	Category	Accuracy	Recall	Precision	F1-score	TPR
	C&C-FileDownload -Troja	66.111%	76.567%	44657%	70.777%	75.098%
	FileDownload-Troja	94.563%	98.700%	99.000%	99.99%	99.778%
	C&C-HeartBeat-Gagfyt	96.333%	100%	100%	97,230%	99.888%
	DDoS-Gagfyt	99.990%	100%	97,343%	99,990%	90.000 %
	C&C-HeartBeat-Okiru	90.111%	99.999%	90,990%	100%	77,403%
	Okiru	95.355%	100%	99,068%	100%	98,553%
	Okiru-Attack	23.008%	91,430%	99,000%	96,999%	97,760%
	C&C-Hakai	69.888%	99,999%	85,612%	92,000%	100 %
Naïve bayes	Benign	98.999 %	98.999 %	100%	97,230%	98.999%
	C&C-Mirai	99.564%	92.999 %	95,333%	98,999%	99.111 %
	DDoS- Mirai	90.222%	99.998%	97,250%	100%	90,999%
	Port Scan - Mirai	90.655%	100%	99,068%	99.999 %	98,293%
	C&C-Torii1	19.888%	68,432%	79,007%	79,007%	79,007%
	C&C-Torii2	98.777%	98,999%	95.111%	97,500%	100 %
	C&C-Troja	100%	99.999%	90.799%	99.888%	100 %
	C&C-FileDownload -Troja	97,008 %	99.118%	98.000 %	99.118%	97.657%
	FileDownload-Troja	97,999 %	99.999%	99.999%	99.901 %	88.999%
	C&C-HeartBeat-Gagfyt	97.999%	95.483%	99.000%	90.756%	95.098%
	DDoS-Gagfyt	98.333%	96.700%	90.156%	97.348%	99.778%
	C&C-HeartBeat-Okiru	88.118%	100%	100%	95,870%	98.111%
	Okiru	79.999%	100%	77,333%	78,230%	70.999 %
	Okiru-Attack	89.111%	97.998%	97.998%	97.998%	77.998%
	C&C-Hakai	90.777%	100%	99,068%	100%	98,23%
VAE-MLP	Benign	99.993%	99.999%	99.999%	99.948%	99.988%
	C&C-Mirai	99.967%	100%	98.967%	100%	99.918%
	DDoS- Mirai	99.990 %	100%	99.990 %	99,730%	99.922 %
	Port Scan - Mirai	99.998%	99.698%	99.998%	100%	98,990%
	C&C-Torii1	100%	99.999%	100%	99.888%	100 %
	C&C-Torii2	99,230%	99.118%	99,230%	99.118%	99.987%
	C&C-Troja	99,132%	99.076%	99,132%	100%	99.999%
	C&C-FileDownload -Troja	99.841%	98.483%	99.841%	99.756%	98.998%
	FileDownload-Troja	100%	99.700%	100%	99.990%	99.978%
	C&C-HeartBeat-Gagfyt	99.118%	100%	98.118%	98.888%	99.818%
	DDoS-Gagfyt	99.342%	99.122%	99.342%	99.999%	100 %
	C&C-HeartBeat-Okiru	99.912%	99.998%	99.912%	100%	100 %
	Okiru	99.955%	100%	99.955%	100%	99,923%
	Okiru-Attack	99.833%	99.900%	99.833%	99.948%	99.997%
	C&C-Hakai	100%	100%	100%	99,930%	99.900%

during the execution of VAE-MLP in the fog node. 6a and 6b show that VAE-MLP did not generate a significant overhead in the fog node memory. Thus, the proposed approach can operate exhaustively without generating memory overload and processing in the fog node.

The experimental results show that VAE-MLP can detect a wide variety of attacks initiated by infected IoT devices. The proposed approach was able to detect 99.99% of DDoS attacks; as can be seen in Table 5 and Table 6, VAE-MLP shows excellent performance in detecting DDoS attacks. VAE-MLP provides a better accuracy rate (99.98%) in comparison with existing work [40] (99.62%). The approach proposed in [41] provides an accuracy of 90,3%, no other metrics of this work were available for comparison. Regarding recall, the VAE-MLP approach presented an excellent recall rate of 99.98%. The results are superior to the results obtained by the related works. Observing Table 7, it is evident that the experimental results with IoT-23 dataset show that VAE-MLP provides better precision and accuracy (see figure 7). In terms of recall, VAE-MLP outperforms the existing approaches. Few approaches have made available the F-score rates achieved by their methods. Among the studies that provided this information, the closest to the VAE-MLP approach is the work proposed in [43] which achieved an F-score of 99.853%. Another approach that presented a good F-score rate was [44], which obtained 93.40% . However, the precision of the aforementioned approach is not as good as its F-score (90.2%), it also shows a higher false positives rate. Figure 8 shows that VAE-MLP provides better overall precision of detection accuracy. In terms of recall rate, the proposed approach also outperformed the others, with a rate of 99.56%. The second-best result belongs to the Tim et al. [43] approach with 98.71% accuracy, which uses MLP to detect abnormal traffic. The Zhipeng [46] proposed an anomaly detector approach based on logistic Regression algorithm. This approach achieved 86% accuracy.

5 Conclusion

Security is one of the biggest challenges in IoT environment. Traditional intrusion detection systems need to be adapted to the specific characteristics of IoT environment. In this paper, we have proposed a novel, robust and lightweight intrusion detection model based on combining Variational AutoEncoder (VAE) with a multi-layer perceptron algorithm to detect cyber attacks in IoT environment. The proposed IDS architecture operates on the computing fog layer. The fog computing layer has processing nodes closest to the physical system which provides processing mechanisms and edge storage so that it can detect threats at a faster rate. The proposed approach has been evaluated with two recent datasets. The experimental results show that IDS outperforms different well known and recent machine learning models in terms of detection performance, and is more effective in detecting sophisticated attacks. In addition, the proposed method consumes minimal memory and processing overhead at the fog node.

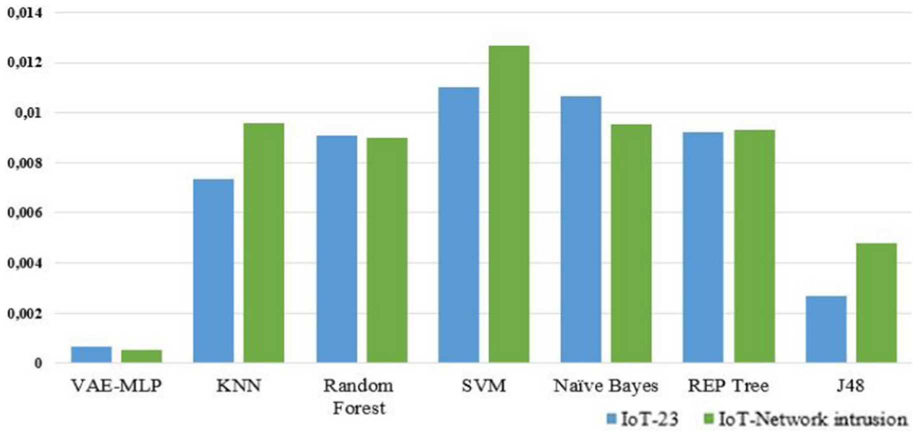

Fig. 5 Overall Performance of the proposed IDS and other classifiers in terms of false positive rate

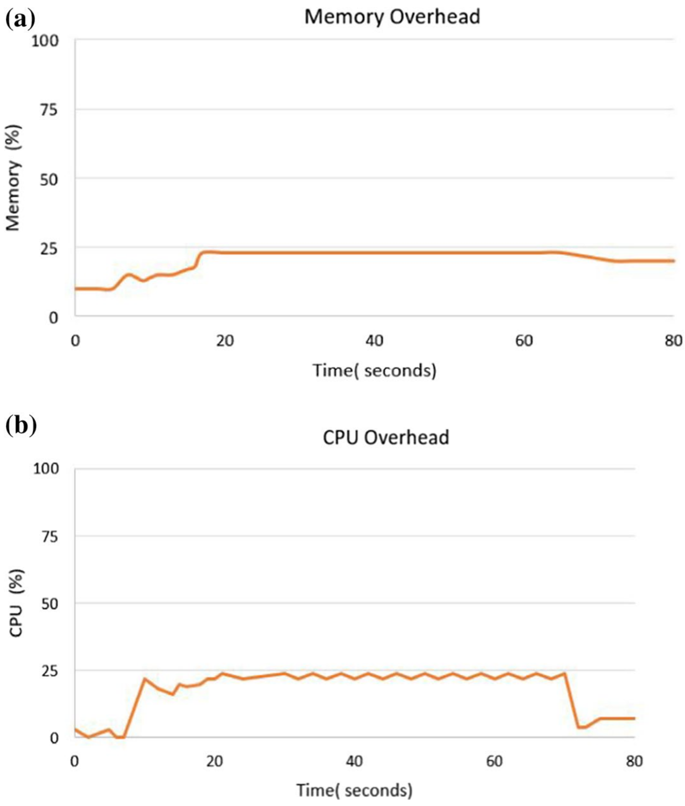

Fig. 6 **a** Performance of the VAE-MLP approach in relation to the memory cost in the fog node. **b** Performance of the VAE-MLP approach in relation to the CPU cost in the fog node

Table 7 Performance comparison of different recents approaches with IoT-23 dataset

Work	Machine/depp learning methods	Binary\ Multi-class	Accuracy	Others metrics
Nicolas-Alin Stoian (2020) [39]	Random Forest	B	99.50%	Recall = 85.00%
Imtiaz Ullah (2020) [40]	CNN	B\M	99.62%	Precision = 99.61% Recall = 99.62% F1-score = 99.61%
Vibekananda Dutta (2020) [41]	DAE, mLSTM and MLP	B	90.3%	MCC= 89.70%
Mandira [42]	Multi-Class Neural Network	B	85.2%	TPR =71.0% Probability of Detection = 71.0% FPR : 0.6%
Tim M. Booij (2021) [43]	MLP	B	99.42%	F1-score = 99.853% ACC =99.352%
Rafał (2021) [44]	a two-layer feed-forward neural network	B	90.2 %	Recall = 96.70% f1-score = 93.40
Hrushikesh (2021) [45]	KNN	B	89.21%	Kappa = 80.95%
Our work	Deep variationnel autocnoder (VAE) and Mlp	B\M	99.98%	Recall =99.87% f1-score =99.98% Precision = 99.67 % TPR=99.99% FPR= 0.0006%

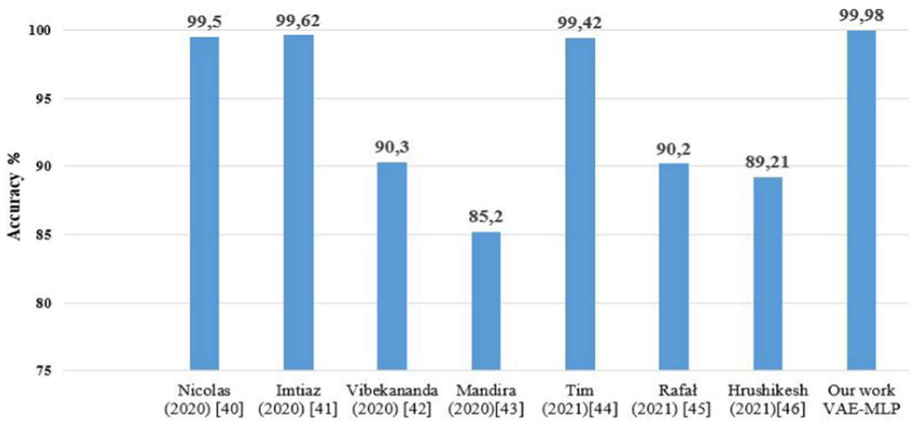

Fig. 7 Performance comparison with different recents approachs on IoT-23 dataset

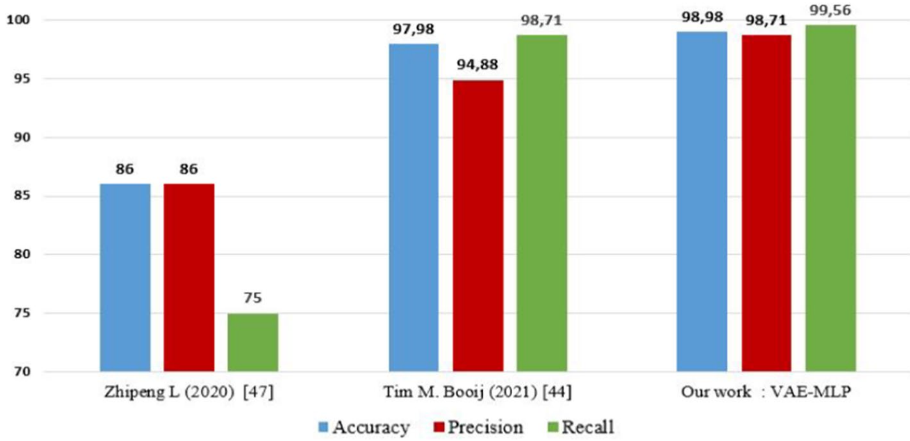

Fig. 8 Performance comparison with different recents approachs on IoT-Network intrusion dataset

6 Future Work

We believe there is still room for improvement in terms of computational and detection performance through exploring more network features and using other algorithms. As future work, we need to explore more cyberattacks and to propose a sophisticated response intrusions approach. Furthermore, testing the efficacy on different fog nodes could be a helpful in evaluating the performances of the IDS.

Acknowledgements This research is a result from PRFU project C00L07UN23 0120180009 funded in Algeria by La Direction Générale de la Recherche Scientifique et du Développement Technologique (DGRSDT).

References

1. Alaba, F. A., Othman, M., Hashem, I. A. T., & Alotaibi, F. (2017). Internet of things security: A survey. *Journal of Network and Computer Applications, 88,* 10–28.
2. Yang, Y., Wu, L., Yin, G., Li, L., & Zhao, H. (2017). A survey on security and privacy issues in internet-of things. *IEEE Internet of Things Journal, 4*(5), 1250–1258.
3. Sadeeq, M. A., Zeebaree, S. R., Qashi, R., Ahmed, S. H., & Jacksi, K. (2018, October). Internet of things security: A survey. In 2018 international conference on advanced science and engineering (ICOASE) (pp. 162-166). IEEE.
4. Atlam, H. F., Walters, R. J., & Wills, G. B. (2018). Fog computing and the internet of things: A review. *Big Data and Cognitive Computing, 2*(2), 10.
5. Atlam, H. F., Alenezi, A., Alharthi, A., Walters, R., & Wills, G. Integration of cloud computing with internet of things: Challenges and open issues. In Proceedings of the 2017 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE Cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData), Exeter, UK, 21-23 June 2017 (pp. 670-675).
6. Ai, Y., Peng, M., & Zhang, K. (2017). Edge cloud computing technologies for internet of things: A primer. Digital Communication Network in press.
7. Sudqi Khater, B., Abdul Wahab, A. W. B., Idris, M. Y. I. B., Abdulla Hussain, M., & Ahmed Ibrahim, A. (2019). A lightweight perceptron-based intrusion detection system for fog computing. *Applied Sciences, 9*(1), 178.
8. An, X., Lü, X., Yang, L., Zhou, X., & Lin, F. (2019). Node state monitoring scheme in fog radio access networks for intrusion detection. *IEEE Access, 7,* 21879–21888.
9. Hu, P., Dhelim, S., Ning, H., & Qiu, T. (2017). Survey on fog computing: Architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications, 98,* 27–42.
10. Ali, M. H., Al Mohammed, B. A. D., Ismail, A., & Zolkipli, M. F. (2018). A new intrusion detection system based on fast learning network and particle swarm optimization. IEEE Access, 6, 2025-5-20261
11. Abeshu, A., & Chilamkurti, N. (2018). Deep learning: The frontier for distributed attack detection in fog-to-things computing. *IEEE Communications Magazine, 56*(2), 169–175.
12. da Costa, K. A., Papa, J. P., Lisboa, C. O., Munoz, R., & de Albuquerque, V. H. C. (2019). Internet of things: A survey on machine learning-based intrusion detection approaches. *Computer Networks, 151,* 147–157.
13. Kim, J., Kim, J., Thu, H. L. T., & Kim, H. (2016, February). Long short term memory recurrent neural network classifier for intrusion detection. In 2016 International conference on platform technology and service (PlatCon) (pp. 1-5). IEEE.
14. Koroniotis, N., & Moustafa, N. (2020). Enhancing network forensics with particle swarm and deep learning: The particle deep framework. arXiv preprint arXiv:2005.00722.
15. Hosseinpour, F., Vahdani Amoli, P., Plosila, J., Hämäläinen, T., & Tenhunen, H. (2016). An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach. International Journal of Digital Content Technology and its Applications, 10.
16. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
17. https://www.unb.ca/cic/datasets/ids.html
18. An, X., Zhou, X., Lü, X., Lin, F., & Yang, L. (2018). Sample selected extreme learning machine based intrusion detection in fog computing and MEC. Wireless Communications and Mobile Computing, 2018.
19. Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for internet of things. *Future Generation Computer Systems, 82,* 761–768.

20. Peng, K., Leung, V., Zheng, L., Wang, S., Huang, C., & Lin, T. (2018). Intrusion detection system based on decision tree over big data in fog environment. Wireless Communications and Mobile Computing, 2018.
21. 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 Journal on Wireless Communications and Networking, 2018*(1), 1–9.
22. Illy, P., Kaddoum, G., Moreira, C. M., Kaur, K., & Garg, S. (2019, April). Securing fog-to-things environment using intrusion detection system based on ensemble learning. In 2019 IEEE Wireless Communications and Networking Conference (WCNC) (pp. 1-7). IEEE.
23. https://www.unb.ca/cic/datasets/nsl.html
24. Pacheco, J., Benitez, V. H., Félix-Herrán, L. C., & Satam, P. (2020). Artificial neural networks-based intrusion detection system for internet of things fog nodes. *IEEE Access, 8,* 73907–73918.
25. https://research.unsw.edu.au/projects/adfa-ids-datasets
26. Almiani, M., AbuGhazleh, A., Al-Rahayfeh, A., Atiewi, S., & Razaque, A. (2020). Deep recurrent neural network for IoT intrusion detection system. *Simulation Modelling Practice and Theory, 101,* 102031.
27. Krzysztoń, M., & Marks, M. (2020). Simulation of watchdog placement for cooperative anomaly detection in bluetooth mesh intrusion detection system. *Simulation Modelling Practice and Theory, 101,* 102041.
28. Rahman, M. A., Asyharia, A. T., Leong, L. S., Satrya, G. B., Tao, M. H., & Zolkipli, M. F. (2020). Scalable machine learning-based intrusion detection system for IoT-enabled smart cities. Sustainable Cities and Society, 102324.
29. de Souza, C. A., Westphall, C. B., Machado, R. B., Sobral, J. B. M., & dos Santos Vieira, G. (2020). Hybrid approach to intrusion detection in fog-based IoT environments. Computer Networks, 107417.
30. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
31. https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
32. https://towardsdatascience.com/an-overview-of-monte-carlo-methods-675384eb1694
33. https://www.stratosphereips.org/datasets-iot23
34. https://ieee-dataport.org/open-access/iot-network-intrusion-dataset
35. https://www.unb.ca/cic/research/applications.html
36. https://www.cs.waikato.ac.nz/ml/weka/
37. https://deeplearning4j.org/
38. https://maven.apache.org/
39. Stoian, N. A. (2020). Machine learning for anomaly detection in IoT networks: Malware analysis on the IoT-23 data set (Bachelor's thesis, University of Twente).
40. Ullah, I., & Mahmoud, Q. H. (2021). Design and development of a deep learning-based model for anomaly detection in IoT networks. *IEEE Access, 9,* 103906–103926.
41. Dutta, V., Choras, M., Pawlicki, M., & Kozik, R. (2020). Detection of cyberattacks traces in IoT data. *Journal of Universal Computer Science, 26*(11), 1422–1434.
42. Hegde, M., Kepnang, G., Al Mazroei, M., Chavis, J. S., & Watkins, L. (2020, October). Identification of botnet activity in IoT network traffic using machine learning. In 2020 International conference on intelligent data science technologies and applications (IDSTA) (pp. 21-27). IEEE.
43. Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N., & den Hartog, F. T. (2021). ToN.IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion datasets. IEEE Internet of Things Journal.
44. Kozik, R., Pawlicki, M., & Choraś, M. (2021). A new method of hybrid time window embedding with transformer-based traffic data classification in IoT-networked environment. Pattern Analysis and Applications, 1-9.
45. Chunduri, H., Kumar, T. G., & Charan, P. S. (2021, February). A multi class classification for detection of IoT botnet malware. In International conference on computing science, communication and security (pp. 17-29). Cham: Springer.
46. Liu, Z., Thapa, N., Shaver, A., Roy, K., Yuan, X., & Khorsandroo, S. (2020, August). Anomaly detection on iot network intrusion using machine learning. In 2020 International conference on artificial intelligence, big data, computing and data communication systems (icABCD) (pp. 1-5). IEEE.

Yasmine Labiod received the Bachelor's degree (June, 2015), and Master's degree (June, 2017) from Badji Mokhtar- Annaba University, Algeria, all in Computer Science. Since October 2017, she is a Ph.D. Student at the Department of Computer Science, Annaba University, Algeria. She is also affiliated as a Researcher member with Networks and Systems Laboratory - LRS, Badji Mokhtar - Annaba University, Algeria. Her research interests include intrusion detection, machine learning for anomaly detection, and wireless network security.

Abdelaziz Amara korba received his Ph.D. degree from Badji Mokhtar AnnabaUniversity, Algeria, in 2016. Currently, he is an Associate Professor in the Department ofComputer Science at the University of Badji Mokhtar Annaba, and he is member ofNetworks and Systems Laboratory (LRS), Badji Mokhtar Annaba University. His researchinterests include network traffic mining, intrusion detection, machine learning for anomalydetection, and wireless network security.

Nacira ghoualmi has Master in computer sciences in 1994 during cooperation programbetween Glassgow University-UK and Badji Mokhtar University-Algeria. She receivedher state doctorate in communication systems at computer sciences department from BadjiMokhtar University-Algeria in 2005. She is associate professor and researcher at computersciences department at Badji Mokhtar University–Annaba-Algeria.She was head ofcomputer sciences department at Badji Mokhtar University by 2000 to 2002 and memberof national pedagogic committee in Algeria by 2000-2001. Actually she is head of Masterand Doctoral entitled Networks and security. Also, she is head of research laboratoryentitled: Networks and systems.