Intrusion Detection Based on Autoencoder and Isolation Forest in Fog Computing

KISHWAR SADAF AND JABEEN SULTANA
Department of Computer Science, College of Computer and Information Sciences, Majmaah University, Al Majma’ah 11952, Saudi Arabia
Corresponding author: Kishwar Sadaf (k.sadaf@mu.edu.sa)
This work was supported by the Majmaah University’s Deanship of Scientific Research under Project 1439-64.

ABSTRACT Fog Computing has emerged as an extension to cloud computing by providing an efficient infrastructure to support IoT. Fog computing acting as a mediator provides local processing of the end-users’ requests and reduced delays in communication between the end-users and the cloud via fog devices. Therefore, the authenticity of incoming network traffic on the fog devices is of immense importance. These devices are vulnerable to malicious attacks. All kinds of information, especially financial and health information travel through these devices. Attackers target these devices by sending malicious data packets. It is imperative to detect these intrusions to provide secure and reliable service to the user. So, an effective Intrusion Detection System (IDS) is essential for the secure functioning of fog without compromising efficiency. In this paper, we propose a method (Auto-IF) for intrusion detection based on deep learning approach using Autoencoder (AE) and Isolation Forest (IF) for the fog environment. This approach targets only binary classification of the incoming packets as fog devices are more concerned about differentiating attack from normal packets in real-time. We validate the proposed method on the benchmark NSL-KDD dataset. Our technique of intrusion detection achieves a high accuracy rate of 95.4% as compared to many other state-of-art intrusion detection methods.

INDEX TERMS Autoencoder, fog computing, intrusion detection, isolation forest.

I. INTRODUCTION
IoT devices like health monitoring systems, gaming, banking, home alarm systems, smart vehicles, etc. require a proper milieu. With the proliferation of IoT devices for real-time applications, high latency, high energy consumption, intrusion detection, secure communication, etc. have become pertinent issues. Fog computing has emerged as a solution to address the high latency and high energy consumption problems of cloud computing. Fog computing, a word given by Cisco which refers to comprehensive cloud computing also known as edge computing or fogging, facilitates the operations involving computers, data storage, and services related to networking amid fog devices, devices storing data at cloud centers and end devices [1], [2]. Fog computing represents a layer (fog) between cloud and end-users, the layer being near the edge i.e. near the end-user devices [3] (figure 1). This proximity between the fog layer and the end-users lowers the risk of en route attacks. Fog computing uses devices like access points, gateways, and temporary storage devices that reduce the connection delay and power consumption of big cloud data centers. These devices are called fog devices. Instead of communicating with cloud data centers, end-users interact with fog devices. In [4], the authors present a detailed comparison between cloud and fog computing in terms of latency (communication delay) and energy consumption. The cloud data centers and other cloud processing units are well-equipped with effective and efficient security features. But the fog devices have low resources like processors, memory, etc. and are vulnerable to attacks by malicious entities on the network. An attacker or intruder can secretly enter the network to harm the user data. An intrusion detection system (IDS) is a powerful approach to detect the presence of intruders in the network. There are two forms of detection: signature-based and anomaly-based. In signature-based IDS, incoming network traffic is compared with pre-installed rules. If the packet matches the rules, then they are dropped and appropriate actions are taken. In anomaly-based IDS, the normal behaviour of the system is considered as a model. The system inspects the behaviour of the incoming
traffic. And if the current behaviour deviates from the normal, the system classifies this as an anomaly and takes corrective measures.

FIGURE 1. "Fog Computing" by Cisco [1].

In a world where new threats and attacks surface every other day, updating the database of signature-based IDS is not feasible. The anomaly-based detection of attacks works well for network security. They are also applicable in the areas of fraud detection, medical applications, etc. Therefore, our method is based on the anomaly-based detection technique. An effective anomaly-based IDS is characterised by accurate segregation of normal and attack network traffic with efficiency.

The shift of major online communication from conventional devices to IoT devices has changed the way security has been perceived for years. Moreover, with fog computing, the security dynamics have changed considerably. The importance and significance of security in fog computing environment can be found in these studies [5], [6] and [7].

Various studies on intrusion detection in fog environment have been done. They are generally based on supervised and unsupervised machine learning. Lately, detection methods based on ensemble methods and deep learning approaches have emerged as prevalent detection techniques. In [8], authors provided a comprehensive study on the various machine learning techniques like Decision Tree, KNN, Random Forest, etc. and how machine learning based intrusion detection techniques at fog layer are capable of detecting abnormalities or attack. Authors [9] proposed a smart data approach for intrusion detection in fog computing. Smart data is formatted data on which immediate actions can be taken. The fog layer uses this data to raise intrusion alerts. To handle Distributed Denial of Service (DDoS) attack on fog devices, An et al. [10] proposed a method based hypergraph clustering using Apriori algorithm of association rule mining. Using this, the association among the fog nodes, which are suffering from the DDoS attack, can be found effectively. In [11], the authors employed decision tree method for intrusion detection. First, they digitized and pre-processed the massive data generated by fog devices and then applied decision tree to this data. In [12], the authors proposed an approach to detect and prevent intrusion like MiTM-Man in the middle attack. Every node in the IDS probes for other nodes of fog and examines the reaction by calculating their arrival time. Intrusion prevention system makes use of lightweight encryption and decryption to avoid attacks of man in the middle like eaves dropping, wormhole attacks, and packet alteration. In [13], the authors suggested a framework to detect DDoS attacks in fog environment by implementing rules at the fog layer.

In this paper, we propose an intrusion detection method based on deep learning approach using autoencoder and isolation forest. It is a two-staged detection technique that performs binary classification. The analysis of the attacks can be done by the cloud service provider once it receives the historical data from the fog node. This historical data may span from minutes to days. Therefore, our method concentrates on segregating attacks from the normal network traffic data at the fog layer. Our method shows promising results. As far as we know, [14] is the only study based on the autoencoder and isolation forest. Authors employed stacked autoencoder for feature extraction and then use isolation forest for classification of attacks.

The rest of the paper is organized as follows: Section 2 introduces some methods related to autoencoder and isolation forest in the context of intrusion detection. Some basic concepts are explained in section 3. Our proposed method is explained in section 4. In section 5, experiment, results, performance, and comparisons with other methods are discussed. In section 6, a conclusion based on the proposed work is drawn.

II. RELATED WORK

The security concern in fog computing is drawing many studies and researches. In this section, we discuss some similar works carried out in the area of intrusion detection systems including works dedicated to intrusion detection in the fog environment. Fog computing emerged since cloud computing was not able to manage more load of data at the cloud because of the increase in IoT devices, resulting in latency problems which are not acceptable in many environments. In the Fog environment, fog nodes like switches, routers, video cameras, and controllers are set up throughout the system and are installed at target setup zones. As soon as any of the devices generate data, immediately the nearest fog node processes it without needing to send data straight away to the cloud center. Fog nodes send periodic data summaries to the cloud for analysis purposes so that the cloud service provider can enhance its future performance.

Generally, Intrusion detection techniques are divided into two categories namely, signature-based and anomaly detection (AD) based. Signatures present in the database are matched with the traffic arriving from the network to identify signature-based intrusion patterns. Known attacks can be detected by using signature-based Intrusion techniques, but as the network access goes on and new data arrives, it has to be continuously updated to detect new intrusion patterns [15]. Detection methods based on anomaly identification are capable of sensing the anonymous zero-day attacks. Anomaly detection methods based on machine learning techniques are tough to unknown attacks [16], [17]. To detect unknown or strange attacks, Intrusion detection techniques...
based on anomaly detection are extensively researched compared with Intrusion detection techniques based on signatures. The unknown attack is identified by finding out the change in the functioning of the system as compared to its normal functioning [15]. Techniques to detect anomaly can be classified into three approaches; supervised learning, unsupervised learning, and semi-supervised learning. Supervised learning deals with known class labels for the traffic regularly arriving from the network and already known intrusions that are supplied to the network. This results in predicting the network attacks with high accuracy but proves costly and consumes much time for labeling the class. In the unsupervised learning process, data is unlabeled and therefore, clusters are formed from the known data based on certain similarities using algorithms like k-Mean, k-medoid. In the semi-supervised approach, supervised and unsupervised learning concepts are used together, which consist of both the labeled data and unlabeled data to cut the clumsy process of assigning labels to packet flows, particularly when the systems get huge and diverse data samples.

To detect the intruders inside the intrusion detection systems, algorithms like Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Multi-Layer Perceptron (MLP), and etc. are used for the supervised learning process. Tavallaee et al. [18] provided a significant study on how KDD-CUP’99 dataset is flawed and how NSL-KDD dataset overcomes these flaws. They performed a comparative analysis of the performance of various machine learning methods; J48, Naïve Bayes, NB Tree, Decision Tree, Random Forest, MLP and SVM, on these datasets. A stacked SVM was proposed by Chand et al. [19] to detect the attacks and was effective in identifying the intrusion. A stacked SVM is considered as an effective classifier for intrusion detection when matched with the rest of other classifiers. Genetic Algorithm was combined with SVM to increase the accuracy and lessen the model building time by picking the most relevant features by Tao et al. [20]. Furthermore, hybrid classifiers were suggested by combining various algorithms like Random Tree, REP Tree, J48, and Naïve Bayes and by using both the supervised learning and unsupervised learning process. A novel hybrid method to reduce the time consuming process in selecting the features by associating one feature with another was suggested by Aljawarneh et al. [21]. An approach based on theoretical game for lightweight Intrusion Detection System was developed in combination with signature based intrusion detection and anomaly detection to mainly achieve maximum detection rate in terms of accuracy and minimum rate for false-positive [22]. One more light-weight Intrusion Detection System was developed to get rid of space limit at fog nodes, named as SSELM in order to get rid of the space limit at nodes of fog. This proposed method used the KDDCUP-99 dataset to attain high classification accuracy, and to improve time in building the model [23]. Also, past research mainly considered SVM to safe smart grid systems and to detect Intruders and it was observed that the machine learning classifiers like SVM, KNN, Random Forest, and KNN outperformed in intrusion detection [24]. Ling and Wu [25] proposed an ensemble method of intrusion detection. Random Forest was used as feature selection method. Then selected features were used to train the multi-classifier based ensemble method comprising SVM, KNN, Naïve Bayes, and Decision Tree. Later, deep learning is employed to ensemble the output of multi-classifier, which effectively improves the accuracy of intrusion detection. In [26], authors proposed a fuzzy approach based on semi-supervised machine learning performing binary classification. A single hidden neural network is trained on unlabeled samples. The vectors produced are then classified into fuzzy categories (high, mid and low). These categories are the incorporated with the original training and the network is retrained.

To detect U2R and R2L attacks, a model was suggested to improve accuracy by reducing the feature dimensions. Dual features were applied to a 2-tier model for classifying the attacks using KNN and NB classifiers [27]. Based on space representation using vectors, a lightweight Intrusion Detection System was suggested by Khater et al. [28]. The suggested model used a single hidden layer of MLP and a small number of nodes to evaluate two different datasets containing information of intrusion attacks and they belong to different domains on different applications of a particular region. The model is shown to have achieved the accuracy of 94%. It was found that using deep learning, highly accurate results are achieved in terms of classification compared to conventional methods of machine learning [29].

A. DEEP LEARNING BASED IDS

Deep learning as a subset of machine learning is being widely applied in the area of cybersecurity due to its capability to learn new, unknown patterns in the raw data. It uses multiple layers of transformations to find higher-level features. Deep learning is being used to solve the problems of classification, image recognition, self-driving cars and speech recognition on huge datasets. It makes use of several unknown layers to automatically select the features or mine attributes and then perform training and testing on the given data to get the classification results. Deep learning involves feature selection and training through the same process, where as in conventional machine learning: first features need to be extracted then training and testing is done. There are several variants of deep learning. Autoencoder is one of them.

In [30], an STL-IDS based approach using the concept of deep learning was proposed for learning the features based on stack auto-encoders and to reduce the dimensions, the proposed framework uses a support vector machine instead of Soft-max function. The obtained results from this proposed approach on the NSL-KDD dataset reveals that SVM attained high classification accuracy and enhanced training and testing times for two-class and five-class classification when compared with other classification techniques like naïve Bayes, random forest, and J48. Yin et al. [31] proposed a method based on the Recurrent Neural Network (RNN)
for intrusion detection. Their method achieved an accuracy rate of 83.28%. In [32], the authors implemented different models of deep learning including autoencoder, RNN, and convolutional neural network. A Network AD system was suggested in real-time to minimize the amount of work done by humans by pairing up dual stages of learning in [33]. A shallow autoencoder was used at the initial stage to carry out adaptive unsupervised AD. In the next stage, classification was done manually to clean the false positives by using traditional nearest-neighbor classifier. The authors developed a prototype to assess nationwide traffic. Their method is shown to achieve an accuracy of 98.5%.

An approach using a Stacked Autoencoder was suggested by Farahnakian and Heikkonen [34] in which four autoencoders were used and the resulting output of each autoencoder layer as input to the next layer of autoencoder. The Softmax layer was the activation function used to classify the arriving input flows of the network into normal or anomaly ones. An NDAE-Non-Symmetric Deep Auto Encoder was suggested by Shone et al. [35] to perform feature selection using unsupervised concept and classification using random forest. NDAE combines shallow and deep learning to derive benefits and minimize the error rate. Another IDS based on autoencoder was proposed by Choi et al. [36]. They used heuristics to arrive at a threshold value for reconstruction error which defines a data point normal or abnormal. Yan and Han [37] used stacked autoencoder as feature extraction tool to extract low-level features from the NSL-KDD dataset and then applied different classifiers to find the attacks. In [38], the authors asserted that deep learning methods are superior to other conventional classification methods in identifying attacks. They used autoencoder to map high-level features of the NSL-KDD dataset to low-level features and then apply the Softmax classifier. It is evident from the literature that deep learning methods are proving instrumental in intrusion detection especially in the distributed environment of IoT.

Semi-supervised learning approach was suggested for Intrusion Detection Systems known as AutoIDS which can differentiate anomalous flows of packets from regular packets by considering dual effective finders. These finders are dual encoder-decoder neural networks and are required to offer a sparse and compacted depiction from the regular flows. If these neural networks fail to offer compacted or sparse representation from an arriving flow of packet implies as Intruders. To achieve high accuracy with low computational cost, a huge number of packet flows are considered. By maintaining the classification accuracy, a large number of flows are sorted out by the first detector. Mostly the first detector is used and when the first one fails in achieving the desired classification then the second detector is used for hectic samples. The suggested approach AutoIDS was evaluated on the NSL-KDD data set as this dataset is widely used by most of the researchers. This NSL-KDD data set is regarded as a benchmark data set and is the widely-used and well-known dataset. AutoIDS when applied on this data dominated with a maximum accuracy of 90.17%, when compared with the rest of the methods [39]. In [40], authors proposed an autoencoder based feature learning model. The model is validated using different machine learning methods like SVM, Gaussian Naïve Bayes, KNN etc. Their autoencoder model with Gaussian Naïve Bayes as classifier achieves 83.3% accuracy on NSL-KDD dataset. Niyaz et al. [41] proposed a network IDS based on sparse autoencoder and soft-max regression. Feature learning is performed by the autoencoder and soft-max regression method is employed as a classifier.

Anomalies are nothing but outliers or the unusual data points which fall out of the group. As the unusual points are few and different from the others in the group they are more liable to isolate early in the process. Isolation implies separating a sample from other samples. A single instance is being measured to find out whether the instance is suspected for isolation and those instances with more suspect rate fall under the category of anomalies. The most common approaches based on models to detect anomalies, prefer constructing a profile to find normal instances. Some of the distinguished models to detect anomalies are the standard methods based on classification, clustering, which make use of statistical methods [42], [43] and [44]. But this kind of approach to detect anomaly has drawbacks like maximum profiling is done to normal instances but not further enhanced to detect anomalies resulting in the classification of normal instances as anomalies which leads to more false positives. The majority of the existing methods are limited to small datasets for the reason that they possess high computational difficulty.

Isolation Forest shows promise, as a fast and robust anomaly detection tool. Proposed by Tony Liu et al. [45], isolation forest is based on the premise that outliers tend to isolate early if data space is split randomly. This method is discussed in detail in the next section. A novel framework iForestASD was suggested by Ding and Fei [46] to detect anomalies basing on adapted streaming data anomaly detection algorithm for iForest using the sliding window frames. Experimentation considered 4 real-time datasets taken from UCI source validate that the suggested procedure can efficiently detect anomalies for the considered data. Song et al. [47] employed isolation forest method to detect attacks on power systems which are very vulnerable to cyber-attacks. Authors [48] find isolation forest useful in finding anomalous data patterns in sensory data generated by IoT environment. Tao et al. [49] proposed a method of anomaly detection based on isolation forest and Spark. Instead of using a single machine for anomaly detection, the authors took advantage of the multithread environment of Spark and executed isolation forest in parallel.

Based on the above studies, we observed the importance and ubiquitous application of deep learning in intrusion detection. We propose a novel approach for IDS based on autoencoder and isolation forest for fog computing.

III. PRELIMINARIES

Before we discuss our approach, we succinctly explain the concept of the autoencoder and isolation forest.
**A. AUTOENCODER**

An autoencoder is a kind of neural network with multiple layers whose target output is the same as the input with some amount of reconstruction error as the output is similar to the input with reduced changes. Autoencoder makes use of the unsupervised learning to decode or reconstruct the output by encoding the input. To reduce the dimensionality of features, extract relevant features, compress and remove noise from the images, predict sequences, detect anomalies and in recommender systems, autoencoders are widely used.

Without going into specifics, and for the sake of brevity, we explain the general structure of an autoencoder.

![Autoencoder Diagram](image)

**FIGURE 2.** Autoencoder.

A general autoencoder comprises of 4 major components namely; encoder, bottleneck, decoder and reconstruction loss. Encoder helps the model in reducing the features from the given input and further compresses the input data into encoded representation. The layer which contains the compressed input data with minimum features is known as Bottleneck. Decoder helps the model to reconstruct the output from the encoded representation and sees that output is as similar as input. At last, evaluating the performance of decoder and measuring the similarity between the output obtained and the original input is called Reconstruction Loss. Furthermore, back propagation is incorporated to perform training and to further minimize reconstruction loss. This minimum loss depicts the goal of AE which it tries to reach. The encoder function \( E \) will compress the input \( x \) into \( z = E(x) \). The Decoder \( D \) will try to recreate the input as \( x' = D(E(x)) \). Here, the reconstruction loss is the difference between the encoded and decoded vectors.

Mean Square Error (MSE) is one of the methods to measure the reconstruction loss. It is given by:

\[
\text{Loss}(E, D) = \frac{1}{n} \sum_{i=1}^{n} (x_i - D(E(x_i)))^2
\]

Kullback-Leibler (KL) Divergence is another method for calculating reconstruction loss, used in variational autoencoders (VAEs). KL Divergence is a non-negative value and it measures the dissimilarities between two probability distributions; the probability distribution of data in the latent space and the probability distribution of data being projected into the latent space.

Sparse autoencoder, denoising autoencoder, varational autoencoder, convolution autoencoder etc. are some of the autoencoder variants.

**B. ISOLATION FOREST**

Isolation Forest is an unsupervised machine learning method of anomaly detection, which finds anomaly by randomly partitioning the data points. The best part of this method is the isolation of anomalous instances without profiling the normal points.

Isolation Forest, a novel and proficient method introduced by [45] to detect anomalies, assumes that the instances which fall away from the data center are anomalies. It forms like binary trees and ensembles iTrees by sampling randomly for a given dataset. Isolation tree’s key role is to make use of unusual samples, also called anomalies in detecting the unknown attacks which are strange from the normal attacks. Random selection of a subset from the training set is done to build iTrees and it was found that realistic amount is 256 after subsampling and this is the first step in creating iForest. It does not make use of distance measures, hence reduction found in the cost required for computing. Secondly, iForest utilizes no distance or density measures to detect anomaly, this second step, therefore, eliminates computational cost compared with distance measures involved in clustering and takes time complexity in linear fashion. Lastly, iForest requires a low amount of memory and uses the idea of ensemble and does not bother if some iTrees does not yield efficient results as the ensemble algorithms convert the weak trees into efficient ones. Due to all these benefits, using iForest is strongly recommended to detect anomalies on huge datasets involving complex features.

It calculates anomaly score \( S \) as:

\[
S = 2^{-\frac{E(h(x))}{C(n)}},
\]

where \( h(x) \) is the number of edges in a tree for a certain point \( x \) and \( C(n) \) is normalization constant for a dataset of size \( n \). Binary class gets separated based on threshold value on anomaly score in supervised classification and without threshold value in unsupervised classification.

**IV. PROPOSED WORK: AUTO-IF**

In our research work, we have proposed an intrusion detection system for the Fog environment as this is crucial for network security to identify malicious traffic. In this section, we present our proposed method (Auto-IF) for intrusion detection which is based on autoencoder (AE) and isolation forest (IF) methods. Although, AE achieves good accuracy on its own, but to further increase the accuracy, we employed Isolation Forest (IF).

Our proposed method involves two stages of anomaly detection. The output of the first stage acts as the input to second stage. As depicted in the flowchart (Figure 3), test dataset is supplied to the autoencoder in stage 1. AE identifies the attack and segregates the attack and normal network traffic data into two sets. However, the resultant sets contain data points which ideally don’t belong to them. Isolation forest in stage 2, attempt to identify these misfit (outlier) data points, which improves the overall accuracy.
For a given dataset of data packets $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$, where $x_i$ represents a packet information and $i = 1, 2, \ldots, N$. Data packets are labelled into binary classification $y$, where $y_i \in \{0, 1\}$. Normal packets are labelled as 0 and malicious packets as 1.

**A. PREPROCESS**

The training and testing datasets are normalized as the datasets contain numerical and nominal values. The goal of normalizing values is to make every feature have the same scale. Our approach considers all the features of the dataset. Therefore, each feature is equally important.

**B. TRAINING THE AUTOENCODER**

We train the autoencoder only on normal data packets\(^1\) (Figure 4). There are several advantages of this approach. First, training the AE only on normal data traffic overcomes the issue of class imbalance of the NSL-KDD dataset. Second, it allows the model to capture data traffic of normal type and discard the attacks. Third, it makes our approach more viable for real-time applications such as fog devices where decision over normal and attack data traffic must be made in real-time.

---

\(^1\)Improving Network Intrusion Detection using a Denoising Autoencoder with Dropout. Radwan Diab, Mahmoud Aslan and Eiad Soufan.(Github)

---

To train the autoencoder, we split the training dataset $D$ into normal and attack datasets using the available label or class of each data packet sample (Figure 4).

$$D_0, D_1 \leftarrow \text{Split}(D),$$

where, $D_0 \leftarrow (x_i, y_0)$ and $i = 1, 2, \ldots, k$

$$D_1 \leftarrow (x_i, y_1)$$

where $i = 1, 2, \ldots, N - k$

$D_0$ represents the “normal” dataset and $D_1$ represents “attack” dataset. We train the AE on $D_0$.

$$x_k \rightarrow \text{AE} \rightarrow x'_k$$

Let’s consider a set of data packets $T = \{x_1, x_2, \ldots, x_m\}$ where $x_i$ represents network traffic data and $i = 1, 2, \ldots, m$ and AE as the autoencoder.

$$x_i \rightarrow \text{AE} \rightarrow \hat{x}_i$$

Autoencoder AE creates the same number of output as input, but with a reconstruction loss for each $x_i$. Since the AE is only trained on “normal” data, the reconstruction loss for the attack data is much higher than the normal data. We came to a threshold of reconstruction loss value by experiment. If the reconstruction loss value of a data point is higher than the threshold value, then data point is classified as “attack”, otherwise, it will be classified as “normal”.

$$\hat{x}_i \rightarrow \{\hat{x}_p, \hat{x}_q\}$$

where $\hat{x}_p$ represents “normal” data packets having reconstruction loss error less than the threshold and $\hat{x}_q$ contains data points having reconstruction error greater than the threshold and are considered as “attacks”. Since the result of AE is not hundred percent accurate, both $\hat{x}_p$ and $\hat{x}_q$ contain attack and normal data respectively.

To achieve more accuracy i.e. detecting more intrusions, these 2 sets are then supplied as inputs to two Isolation Forest modules (Figure 3). The first module (Isolation Forest 1) gets the “attack” output of the AE and search for the anomalies, in our case – normal data points. Similarly, the second module (Isolation forest 2) takes the “normal” output of the AE and search for anomalies, in this case - attack data points. The attack data in “normal” set and normal data in “attack”
set are nothing but outliers or anomalies. Isolation Forest 2 takes the “normal” \( \hat{x}_p \) and searches for attack data. Since AE has already identified the most the of normal and attack packets in the stage 1, the set \( \hat{x}_p \) contains fewer number of attack packets. The set \( \hat{x}_q \) containing “attack” data is fed to Isolation Forest 1.

\[
\hat{x}_n^a, O^a_n \leftarrow \text{Isolation Forest } 1(\hat{x}_n^q) \\
\hat{x}_n^b, O^b_n \leftarrow \text{Isolation Forest } 2(\hat{x}_p^q)
\]

At the end, data in sets \( \hat{x}_n^a \) and \( O^a_n \) (the blue boxes in Figure 3) are accepted as valid communication and allowed to pass the fog device. The results related to the attack and normal traffic can be sent to the cloud for analysis. Based on the decision of the cloud, autoencoder and isolation methods can be properly tuned.

V. EXPERIMENT AND RESULT

A. EXPERIMENT

The KDD CUP99 and NSL-KDD datasets are two most famous benchmark datasets for evaluating intrusion detection techniques. However, KDD CUP99 dataset suffers from various drawbacks [18]. Therefore, NSL-KDD dataset is being widely used by researchers. Recent works on intrusion detection in fog computing environment have been evaluated using this dataset [50]–[52] and [53]. But this dataset also suffers from class imbalance problem. This problem does not affect our method as we train the autoencoder only on the normal instances of the NSL-KDD datasets. For training, NSL-KDDTrain+ is used. NSL-KDDTest+ is used for validating our approach.

The NSL-KDD dataset has 41 features. These features can be categorized as: nominal, binary and numeric.

The whole experiment was performed on Anaconda Machine using Jupyter Notebook. Python has a rich set of libraries for almost all the learning methods. For autoencoder, we used the Keras and Tensorflow python packages. For isolation forest method, we used the python package from sklearn.\(^2\)

An autoencoder cannot operate on nominal data directly. It requires all the input instances to be numeric. To preprocess the nominal or categorical features, we used One Hot Encoding method. Remaining features are preprocessed using MinMaxScaler functions. This action converted the 41 features into 122 features. These features are then fed to the autoencoder. We kept the autoencoder’s parameters relatively simple. We employed the sparse autoencoder for initial detection task. To overcome the problem of overfitting, we added a “dropout” layer at the autoencoder’s input. This layer acts as a regularization constraint. It prevents autoencoder from copying the input to create the output. As the name suggests, dropout layer drops some random neurons from the input during training. This action encourages other neurons

| No | Feature                  | Type    |
|----|--------------------------|---------|
| 1  | duration                 | Numeric |
| 2  | protocol_type            | Nominal |
| 3  | service                  | Nominal |
| 4  | flag                     | Nominal |
| 5  | src_bytes                | Numeric |
| 6  | dst_bytes                | Numeric |
| 7  | land                     | Binary  |
| 8  | wrong_fragment           | Numeric |
| 9  | urgent                   | Numeric |
| 10 | hot                      | Numeric |
| 11 | num_failed_logins        | Numeric |
| 12 | logged_in                | Binary  |
| 13 | num_compromised          | Numeric |
| 14 | root_shell               | Binary  |
| 15 | sa_attempts              | Binary  |
| 16 | num_root                 | Numeric |
| 17 | num_file_creations       | Numeric |
| 18 | num_shells               | Numeric |
| 19 | num_access_files         | Numeric |
| 20 | num_outbound_cmds        | Numeric |
| 21 | is_host_login            | Binary  |
| 22 | is_guest_login           | Binary  |
| 23 | count                    | Numeric |
| 24 | srv_count                | Numeric |
| 25 | srv_serror_rate          | Numeric |
| 26 | srv_serror_rate          | Numeric |
| 27 | rerror_rate              | Numeric |
| 28 | srv_rerror_rate          | Numeric |
| 29 | same_srv_rate            | Numeric |
| 30 | diff_srv_rate            | Numeric |
| 31 | srv_diff_host_rate       | Numeric |
| 32 | dst_host_count           | Numeric |
| 33 | dst_host_serror_rate     | Numeric |
| 34 | dst_host_same_srv_rate   | Numeric |
| 35 | dst_host_diff_srv_rate   | Numeric |
| 36 | dst_host_same_srv_port_rate | Numeric |
| 37 | dst_host_srv_diff_host_rate | Numeric |
| 38 | dst_host_srv_serror_rate | Numeric |
| 39 | dst_host_srv_serror_rate | Numeric |
| 40 | dst_host_rerror_rate     | Numeric |
| 41 | dst_host_srv_rerror_rate | Numeric |

\(^2\)Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
For the isolation forest, most important parameter is the “contamination percentage”. Contamination refers to the probable existence of outliers in the dataset. For this, we observed through experiment that value from 10%-12% gives optimum result as the input dataset contains around these many outliers. It is interesting to know that the isolation forest does not perform well if the dataset contains no outliers.

To validate our approach, we calculated the ability of our method in identifying the attacks correctly using the following evaluation metrics.

### B. EVALUATION METRIC

We validated our proposed method of intrusion detection on NSL-KDD dataset. Our method achieves very promising and superior results. We evaluated the performance in terms of accuracy, precision, recall and f-measure. These metrics are defined as follows:

**Accuracy** of a method on a given test set is the percentage of test instances that it correctly identifies. It is calculated as:

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

**Precision** is the proportion of the positive instances that are correctly identified by the model to the ratio of all positively labeled instances. It is measured as:

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

**Recall** is the proportion of the positive instances that are correctly identified by the model. It is calculated as:

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

**F-measure** is a measure for testing the accuracy, calculated by taking harmonic mean of precision and recall:

\[
\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

where,

- TP (True Positive): Number of instances correctly classified as an attack.
- FP (False Positive): Number of instances wrongly classified as an attack.
- TN (True Negative): Number of instances correctly classified as normal
- FN (False Negative): Number of instances wrongly classified as an attack.

Using the above mentioned metrics, we measured the performance of our method. Figure 8 shows the superior result of our method (Autoencoder and Isolation Forest- Auto-IF) as compared to the autoencoder when used without isolation forest. Autoencoder on its own obtains good accuracy, but achieves better accuracy when combined with isolation forest.
C. COMPARISON WITH OTHER STATE-OF-ART METHODS

Intrusion detection is a heavily researched topic due to its necessity in the modern cyber world. Researchers have employed many strong and sophisticated machine learning methods for intrusion detection. In this section, we compare our method with other state-of-art intrusion detection methods based on conventional machine learning and deep learning techniques in terms of accuracy over NSL-KDD dataset. Table 3 shows, methods based on autoencoder have achieved better result as compared to other methods. The following table (Table 3) is adapted from [30]. The methods mentioned in Table 3 have been evaluated using NSL-KDDTrain+ and NSL-KDDTest+ datasets.

TABLE 3. Comparison with other ID methods.

| No. | Method                        | Accuracy (%) | Type                  |
|-----|-------------------------------|--------------|-----------------------|
| 1   | Multi-layer Perceptron [18]   | 77.41        | Deep Learning based methods |
| 2   | Recurrent Network [31]        | 83.28        | Naive Bayes [40]      |
| 3   | AE+Guassian                  | 83.34        | Naive Bayes [40]      |
| 4   | AE+SMR [41]                  | 88.39        | Naive Bayes [40]      |
| 5   | STL+SVM [30]                 | 84.96        | Naive Bayes [40]      |
| 6   | DNN+KNN[51]                  | 99.77        | Naive Bayes [40]      |
| 7   | DAE+DNN [53]                 | 98.6         | Naive Bayes [40]      |
| 8   | J48 [18]                     | 81.07        | Conventional classification methods |
| 9   | Naive Baysian [18]           | 75.56        | Conventional classification methods |
| 10  | Random Forest [18]           | 80.67        | Conventional classification methods |
| 11  | NB-Tree[18]                  | 82.02        | Conventional classification methods |
| 12  | Fuzzy approach [26]          | 84.12        | Conventional classification methods |
| 13  | Our Method                   | 95.4         | Conventional classification methods |

VI. CONCLUSION

In this study, we proposed a method (Auto-IF) of intrusion detection based on autoencoder and isolation forest for the fog computing. Our approach considers only binary classification of incoming traffic on a fog device where decision over attack and normal traffic must be made in real time. We trained the autoencoder to learn only the normal data. When it encounters attack data, it fails to encode them resulting in higher reconstruction loss. Based on a threshold value over loss, AE separates the normal and attack data. Although autoencoder in this manner achieved very good result but to further enhance the accuracy, isolation forest is used. We tested our method on NSL-KDDTrain+ and NSL-KDDTest+ datasets. Our method achieves high accuracy of 95.4%. We also validated our method by comparing it to other state-of-art methods of intrusion detection.

ACKNOWLEDGMENT

Authors would like to thank Deanship of Scientific Research at Majmaah University for funding this work under the Project Number 1439-64.
[21] S. Aljawarneh, M. Aldwaairi, and M. B. Yassein, “Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model,” J. Comput. Sci., vol. 25, pp. 152–160, Mar. 2018, doi: 10.1016/j.jocs.2017.03.006.

[22] H. Sedjelmaci, S. Menouci, and M. Al-Bahi, “A lightweight anomaly detection technique for low-resource IoT devices: A game-theoretic methodology,” in Proc. IEEE Int. Conf. Commun. (ICC), May 2016, pp. 1–6, doi: 10.1109/ICC.2016.7510811.

[23] X. An, X. Zhou, X. Liu, F. Lin, and L. Yang, “Sample selected extreme learning machine based intrusion detection in fog computing and MCC,” Wireless Commun. Mobile Comput., vol. 2018, Jan. 2018, Article ID 7472095, doi: 10.1155/2018/7472095.

[24] M. Ozay, I. Esnaola, F. T. Yarman Vural, S. R. Kulkarni, and H. V. Poor, “Anomaly detection in IoT environments,” in Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2006, pp. 504–509, doi: 10.1145/1150402.1150459.

[25] F. Tony Liu, K. Ming Ting, and Z.-H. Zhou, “Isolation forest,” in Proc. ICDM, 2008, pp. 413–422. [Online]. Available: https://cs.nju.edu.cn/zhzhou/zhzhou.files/publication/icdm08b.pdf

[26] Z. Ding and M. Fei, “An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window,” IFAC Proc. Volumes, vol. 46, no. 20, pp. 12–17, 2013, doi: 10.3182/20130902-3-CN-3020.00044.

[27] Y. Song, Z. Yu, X. Liu, J. Tian, and M. Chen, “Isolation forest based detection for false data attacks in power systems,” in Proc. IEEE Innov. Smart Grid Technol.-Asia (ISGT Asia), May 2019, pp. 4170–4174, doi: 10.1109/ISGT-Asia.2019.8881319.

[28] H. Li, B. Yu, and T. Zhao, “An anomaly pattern detection method for sensor data,” in Web Information Systems and Applications—WISA (Lecture Notes in Computer Science), vol. 11817, W. Ni, X. Wang, W. Song, and Y. Wei, Eds. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-30952-7-8.

[29] X. Tao, Y. Peng, F. Zhao, P. Zhao, and Y. Wang, “A parallel algorithm for network traffic anomaly detection based on isolation forest,” Int. J. Distrib. Sensor Netw., vol. 14, no. 11, Nov. 2018, Art. no. 155017881881447, doi: 10.1155/2018/155014771881447.

[30] P. Illty, G. Kaddoum, C. Miranda Moreira, K. Kaur, and S. Garg, “Securing Fog-to-Things environment using intrusion detection system based on ensemble learning,” in Proc. IEEE Wireless Commun. Netw. Conf. (WCNC), Apr. 2019, pp. 1–7, doi: 10.1109/WCNC.2019.8885534.

[31] C. A. de Souza, C. B. Westphall, R. B. Machado, J. B. M. Sobral, and G. D. S. Vieira, “Hybrid approach to intrusion detection in fog-based IoT environments,” Comput. Netw., vol. 180, Oct. 2020, Art. no. 107417, doi: 10.1016/j.comnet.2020.107417.

[32] M. Almiani, A. AbuGhazleh, A. Al-Rahafeyeh, S. Atiewi, and A. Razaque, “Deep recurrent neural network for IoT intrusion detection system,” Simul. Model. Pract. Theory, vol. 101, May 2020, Art. no. 102031, doi: 10.1016/j.simpat.2019.102031.

[33] M. Al-Hawawreh, N. Moustafa, and E. Sitnikova, “Identification of malicious activities in industrial Internet of Things based on deep learning models,” J. Inf. Secur. Appl., vol. 41, pp. 1–11, Aug. 2018, doi: 10.1016/j.jisca.2018.05.002.

KISHWAR SADAF received the Ph.D. degree in computer science from Jamia Millia Islamia, New Delhi, India. She is currently working as an Assistant Professor with Majmaah University, Al Majma’ah, Saudi Arabia. She has several peer-reviewed publications in indexed journals and books. Her research interests include machine learning, information retrieval, and intrusion detection. She has memberships with the ACM and IAENG.

JABEEN SULTANA is currently working with Majmaah University, Riyadh, Saudi Arabia. She has several peer-reviewed publications in indexed journals and books. She has participated and presented research papers at international conferences and reviewed publications in indexed journals. Her research interests include machine learning, pattern recognition, and intrusion detection. She avails membership with ACM.

* * *