Journal of Information Security and Applications 77 (2023) 103569

Contents lists available at ScienceDirect

Journal of Information Security and Applications

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

Secure framework for IoT applications using Deep Learning in fog Computing

Ananya Chakraborty, Mohit Kumar [*], Nisha Chaurasia

Department of Information Technology, Dr B R Ambedkar NIT Jalandhar, India

ARTICLE INFO

Keywords:
Internet of Things
Fog computing
Cyber-attacks
Deep Learning
Cloud computing

ABSTRACT

An efficient offloading strategy provides a promising solution to latency-sensitive applications in a cloud fog environment, but the fog computing environment is not fully protected and fascinates the attackers due to its dynamic and hazardous environments. Several security-based frameworks have been proposed by the authors in an integrated cloud fog environment for IoT applications to mitigate, prevent and defend against vulnerabilities, but failed to address the mentioned issues. We have proposed a lightweight secure framework using Deep Learning (DL) techniques to detect security attacks and monitor the network traffic for IoT applications in this article. The proposed Artificial Neural Network (ANN) based model is trained and deployed over a cloud platform while the detection mechanism is implemented on the fog Nodes to reduce the possibilities of any vulnerabilities and delayed execution. The proposed lightweight framework performance is analyzed and tested over NSL-KDD datasets and compared with the respective baseline techniques Support Vector Machine (SVM), Logistic Regression and Decision Trees to validate the results. We observed from the experimental results that the proposed framework provides the guarantee to defend against vulnerabilities and surpass the baseline algorithms in terms of QoS parameters. Our framework achieved 99.43% accuracy, 99.26% precision and least (0.7396%) False Alarm rate.

1. Introduction

There has been a constant growth in the Internet of Things (IoT) layer devices every year. The advent of IoT has brought the whole world under its umbrella, revolutionizing various domains such as smart cities, smart grids, smart industries, Smart Vehicular Networks, healthcare, and the list goes endless. These autonomous systems are embedded with smart sensors, which perform sensing, processing, and data analytics. These devices generate data which are either needed to be executed or stored for the long term. The devices in the layer do consist of computing resources, although in a very limited amount. Hence, to handle such a huge amount of data, cloud, fog, and similar paradigms have been studied. The inclusion of such computing paradigms into the picture increases storage and computation power drastically. The data generated at the IoT layer are sensitive and prone to security attacks. The fact that the data it generates is heterogeneous in nature, makes it challenging to secure the framework [1].

Cloud computing is a computing paradigm designed to provide computing resources in a flexible, on-demand manner, to the users. Resources are pooled from third-party services, and presented in a virtualized manner to the user. When the IoT layer is used in inclusion with cloud, the resultant framework is called Cloud of Things (CoT). This framework is quite famous for bringing cloud's exhaustive resources to the benefit of requests generated at the IoT layer. As a result, the workload is reduced by a considerable amount. Although, the resources that the cloud consists of, are geographically scattered. Hence, if a request is required to be executed within real or near-real time, the distance between the end layer and cloud makes it unrealistic for the latency-sensitive processes [2]. Also, cloud employs third-party resources, increasing the risk of cyber-attacks on the data. The security measures at the cloud level generally fail at taking data and network heterogeneity into consideration [3]. To compensate for the distance of the cloud from IoT, there have been studies conducted on all types of frameworks, like fog, edge, mist, etc. These paradigms are amalgamated with IoT for better QoS attributes on task execution, along with improved security measures. One such paradigm namely Fog computing, has been used quite a lot along with IoT.

The idea behind fog is to bring cloud closer to the end layer. A fog

* Corresponding author.
E-mail address: kumarmohit@nitj.ac.in (M. Kumar).

https://doi.org/10.1016/j.jisa.2023.103569

Available online 24 July 2023
2214-2126/© 2023 Elsevier Ltd. All rights reserved.

layer consists of fog nodes, where each node is capable of computation. Although the computation power of the fog nodes is way less than that of cloud. Resources used in the layer are isolated, i.e., unlike cloud, cog employs its resources for data handling. This ensures that the sensitive data is secured. Another advantage of the paradigm is that the heterogeneity of data is considered while implementing security measures. Now, in a framework where fog is placed in between IoT and cloud, the flow of requests is from IoT to fog or cloud, as depicted in Fig. 1 [11]. Cloud can be preferred in this scenario for requests requiring much larger computation or storage resources, and fog can be used to execute tasks which need to be executed in real or near real-time. The placement of fog improves the security of the framework [2,3,8,10].

The data in this cloud-fog-IoT framework is efficiently managed, but there are security threats it faces [12,16]. As discussed above, cloud has its own disadvantages for security measures used. It has not yet been successful implementing these algorithms over heterogeneous data and networks [19,20]. This issue is resolved to an extent with the inclusion of fog. The security measures implemented on this layer can secure the different types of data. But the drawback it faces is that the algorithms are not standardized [11,20]. Hence in a framework, as shown in Fig 1, there are security attacks which need to be acknowledged and detected. The previous studies have usually not considered the integrated cloud-fog framework. This framework has its own security concerns. Furthermore, the existing solutions are not lightweight in nature and do not consider the latency of the requests. The terminologies used in this paper are depicted in Table 1.

The Distributed Denial of Service (DDoS) attacks is used as an umbrella term for a variety of security attacks as depicted in Fig 2. In a cloud-fog-IoT framework, the presence of IoT devices makes DDoS attacks probable. We propose an Intrusion Detection System (IDS) to secure our framework against Ping of Death, Teardrop, Land Attack and UDP Flood. Ping of Death attack is a security attack where the attacker uses abnormally large data packets to freeze or disrupt a server. In a Teardrop attack, the user transfers malformed or fragmented data packets to the server. In this case, the server is not able to reassemble the data packets, crashing the device. Land Attack refers to the situation when the user set the source and destination in the TCP segment as the same. Here the server/device might crash because of repeatedly processing the data. And UDP Flood attack refers to a DDoS attack where a huge number of UDP packets are transferred to the server. In response to this, the server becomes unresponsive. The proposed framework will secure fog network from all the mentioned attacks using DL based model and ensure to the users for safe services.

1.1. Motivation for the paper

There have been quite a few studies to ensure security in cloud, fog and IoT frameworks [15,18]. In such frameworks, the attacks generate from the IoT layer, affecting the network and resources of fog-cloud. Machine Learning has been implemented to detect attacks in the form of anomalies, such as Random Forest (RF), Support vector machine and Decision Tree (DT) [4]. From the following studies discusses in Section 2, it can be observed that the ML techniques are implemented to detect irregular and suspicious requests from the dataset. Following that, the various types of detected anomalous data are classified into different types of attacks. These techniques are proven to be effective when the dataset is small, compared to the real-time huge datasets. Pertaining to the multi-layer structure, the Deep Learning methods perform better in detecting anomalies for real datasets [5–7]. The reason attributing to the effective performance of DL is that these algorithms need the least in terms of human interaction to detect the attacks. Along with that, DL techniques perform better when dealing with real-time data.

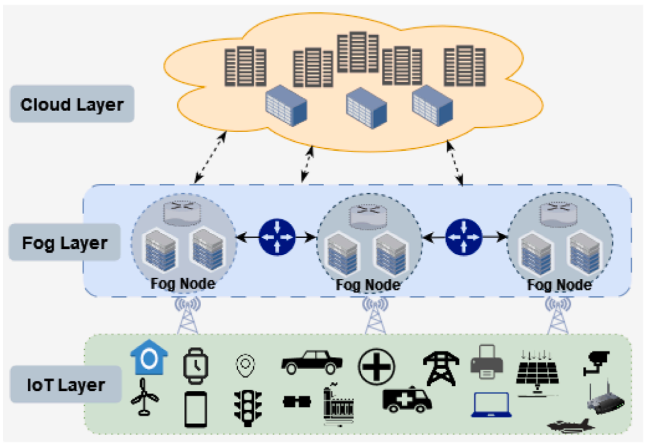

Fig. 1. Structure of the integrated Cloud, Fog and IoT framework.

Table 1
Terminologies and their acronyms used in the study.

Acronyms	Terminology
ML	Machine Learning
DL	Deep Learning
ANN	Artificial Neural Network
SVM	Support Vector Machine
QoS	Quality of Service
CoT	Cloud of Things
IDS	Intrusion Detection System
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
DDoS	Distributed Denial of Service
SVM	Support Vector Machine
ReLU	Rectified Linear Unit

The next area of focus is the layer of the framework where these security measures are to be employed. Generally, studies focus on implementing security measures on IoT, to be able to detect malicious data at the source. Just as this idea has its merits, it has its own challenges. This is because IoT devices are heterogeneous in nature, consisting of very less to none of the resources required for efficiently implementing security measures. Such a framework will face more delay [9]. This can be solved by employing DL algorithms on the fog layer instead of IoT. The training of the DL algorithms can be conducted on cloud because of the presence of a tremendous number of resources. While the DL detection can be implemented on fog nodes, to secure the framework. To validate the efficiency of DL, ML techniques have also been used as baseline. Four algorithms have been implemented on the proposed framework, namely, ANN (Artificial Neural Network), SVM (Support Vector Machine), Logistic Regression (LR) and Decision Trees. The results achieved by each of the algorithms are compared to validate the performance of proposed approach.

The existing solutions are discussed in related work section along with their advantages and limitations. Observing these studies, ML techniques provide precise solutions, but their efficiency reduces with increasing amounts of data. Hence, authors have devised a secure cloud-fog-IoT framework using the DL method that can be used for several IoT applications especially healthcare. In cases of healthcare, one of the main vulnerabilities is to consume the resources of fog servers by suspicious users and fog server will not respond for IoT services to legitimate users. There could be the possibility of a healthcare emergency being overlooked. For this purpose, authors have focused at DDoS attacks and address the issues in the proposed framework. The dataset used in our work, NSL-KDD [36,37], is quite widely for detecting such attacks. This is because the dataset has features which enable the detection of DDoS attacks.

1.2. Contribution of the article

IoT devices are quite prone to security attacks, given the data and network heterogeneity. In a cloud and fog integrated network, the

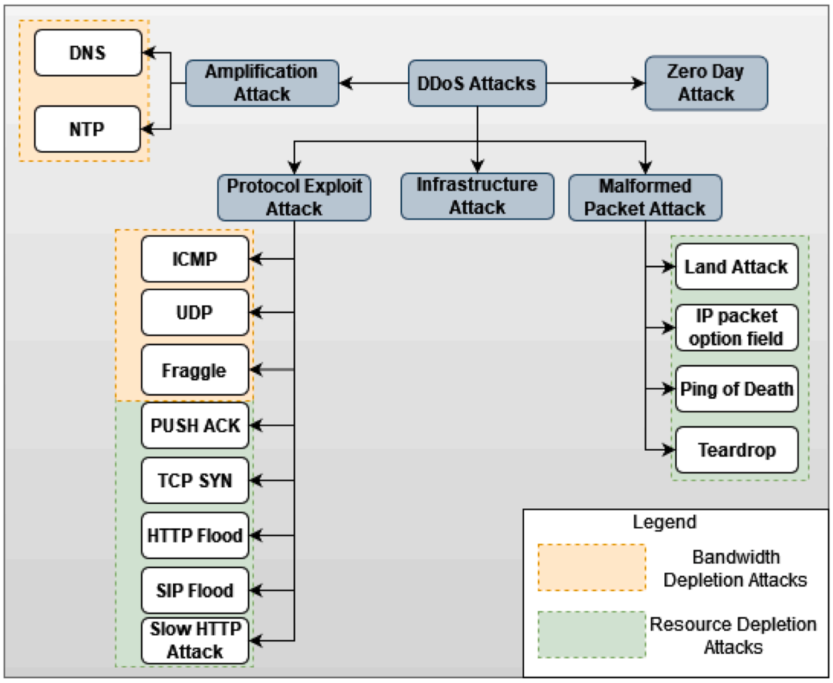

Fig. 2. Distributed Denial of Service Security attacks for Cloud-Fog-IoT framework.

attacks generated from IoT can hamper with the resources and databases. Hence, studies have been conducted to secure the IoT devices. The major contribution of the article is given below:

- The recent state of art trends to detect security attacks and anomalies using machine and deep learning-based approaches have been studied and analyzed along with their limitations.
- We propose a framework to detect the illegitimate traffic, and anomalies to secure the IoT layer using learning techniques in the integrated framework.
- The proposed framework is integrated with ANN that has learning as well as decision-making capability. Subsequently, it is compared with baseline SVM, LR and DT on the NSL-KDD dataset, to detect and monitor the anomalies.
- The performance metrics are computed for the proposed framework and compared with the other approaches to validate the results.

1.3. *Structure of paper*

The remaining paper is presented as: Section 2 describes the related work in the field along with their respective advantages and disadvantages. The following Section 3 presents the framework used for the presented algorithm. Section 4 presents the DL techniques to be used. In Section 5, the mechanisms that help calculate the performance of the proposed algorithm are discussed. In Section 6, the implementation and results are discussed and it is followed with Conclusion and Future Work in Section 7.

2. Related work

Study in securing a cloud-fog-IoT framework has garnered a lot of attention in recent times, because of the need to secure IoT information. Security measures have been applied to fog and cloud layers individually while handling IoT datasets. The need is to apply security algorithms on the whole framework. There have been quite a few studies in the field, basically focussing on ML and DL. The algorithms are used as detection methods, along with various classifiers. The detection methods are trained and modelled to detect the anomaly in incoming data. Whereas, the classifiers help in creating classes for different types of anomalies. Some of these studies have been discussed in Table 2. The Table depicts the aim, attacks focused on, and algorithms used to achieve the aims. Following that, the datasets used are also noted, along with accuracy, advantages of the proposed solution and their limitations. This gives a clearer view of the recent trends in securing a framework.

Many Learning algorithms have been used to detect intrusion in the frameworks. A hybrid ant bee colony optimization algorithm (HABCO) is proposed to differentiate malicious from legitimate requests in CoT paradigm [3]. The algorithm was designed using Ant-colony optimization and Artificial bee colony optimization and the problem of feature selection is turned into an optimization problem. HABCO is used along with classifiers: Naïve Bayes, KNN, etc., to classify the records. JRIP is the best classifier found when used along with HABCO on KDD CUP 99 dataset. The proposed framework is able to detect intrusion attacks at 90.7348%. One downside to this framework needs higher training time. In another work, Support Vector Machine is used to detect Intrusion [22]. The SVM is optimised using PSO and PCA to achieve an improved feature selection for KDD CUP 99 dataset. The only drawback of the framework is the increased rate of false alarm rate. Game Theory is used along with Nash Equilibrium to detect Intrusion [23]. Although the algorithm is lightweight and scalable, no dataset has been specified for the simulation and the following accuracy. In another work, a Secure and Privacy Preserving Distributed Deep Learning (SPDLL) is proposed for Cloud-Fog-IoT framework for detecting intrusion [5]. SPDLL is used along with RSA and FCNN (Fully connected neural network) to authenticate and detect users dropping out. Although the algorithm is able to achieve better accuracy using MNIST data, the complexity of the algorithm is not mentioned.

An attack-detection framework is proposed on Fog layer using Various Deep Learning (DL) models [9]. DL has been used recently in this field. The reason that it is quite efficient is because DL needs the least human interaction to select features. Five different datasets (UNSW-NB15, CICIDS-2017, RPL-NIDS17, N_BaIoT, NSL-KDD) were used for each DL model used (LSTM, RNN, BiLSTM, CNN-LSTM, GRU). Out of all the DL models LSTM is proven to be performing better at feature selection. Although DL can efficiently select features in distributed environments such as fog and cloud, this algorithm did label data

Table 2
State of art comparison of cyber-attacks detection at Internet of Things layer.

Paper	Aim	Attack	Algorithm	Accuracy	Advantages	Limitations
[3]	Feature Selection for Intrusion Detection System	Intrusion	Hybrid Ant Bee Colony Optimization	90.7348	Robust solution	Increased training time
[5]	Secure and privacy preserving Distributed Deep Learning	Intrusion	Fully Connected neural network	NA	Encryption used for authentication	Accuracy not provided
[7]	Object Detection	malware	Deep Learning and Aneka	NA	Two scenarios were provided: for high accuracy and low latency; QoS considered	Requests with deadline not considered
[9]	Deep Learning for attack detection	DDoS	Long short-term memory model	99.96	High detection rate	Wrong labelling of data while classifying
[13]	Secure communication system	Brute Force and Statistical Attack	Chaos-based security protocols	NA	Decision Tree gave best results with varying input, good accuracy	Processing speed might lag
[14]	Two-layer Intrusion detection	Intrusion	Feed Forward Neural Network	99.0	Reduced latency, better detection accuracy	–
[15]	Attack Detection	DDoS	Random Forest and Decision Trees	96.5	Scalable solution, better accuracy	Feature selection not conducted, larger processing time
[17]	Cyber-attack detection in IoMT	Intrusion	Ensemble learning using LSTM	98.56	Real-world dataset used	Training time is larger
[21]	Secure mobile fog-cloud	Malicious data	energy-efficient and secure hybrid (EESH) algorithm	NA	Lower energy consumption	–
[22]	Support vector Machine Intrusion Detection	Intrusion	PCA and PCO optimized SVM	NA	Efficient feature selection	False alarm rate increased
[23]	Intrusion detection using Game Theory	Intrusion	Game Theory and Nash equilibrium solution used	NA	Lower complexity of algorithm	Static IoT
[24]	LSTM Networks	Morphing attacks	LSTM	98.2	Lesser need for manual help during feature selection	–

wrong while classifying. Another work used DL and Aneka in the cloud-fog framework for object detection [7]. COCO dataset is used to provide two scenarios for detection: one when the task is latency sensitive and when it is not so. The framework does provide improved accuracy, jitter, response time, power, and network bandwidth but only if the task is not latency sensitive. As soon as an incoming activity is needed to be executed within a deadline, the accuracy and detection rate is reduced. Focusing on DL, in another study, LSTM networks are used to detect attacks in Fog-to-Things framework [24]. The attacks focused are morphing attacks and it is simulated on the dataset ISCX and AWID while proving the efficiency in improved detection rate (98.2%). Similarly in another framework, ensemble learning is used to detect Intrusion for IoMT [17]. Ensemble learning refers to the method where various Learning algorithms are used simultaneously to achieve the best performance. This algorithm uses various LSTM models for the purpose, on the ToN-IoT dataset, achieving 98.56% accuracy. It is an effective model for intrusion detection except for the downside that the training time is larger and feature selection is not conducted.

An adaptive sliding mode security scheme is developed using Chua's chaotic in CoFA (Class of Fog Applications) framework where the applications are mapped to five levels of chaos-based security protocols [13]. The proposed algorithm is used along with Decision Tree and eventually used on Google Trace dataset. The framework achieved accuracy while detecting brute force and statistical attacks. Although the processing speed is quite large. A two-layer intrusion detection system is proposed in another study using a stacked autoencoder and Feed Forward Neural Networks, namely Fog-Layer Binary Neural Network (FBiNN) [14]. The algorithm is implemented on two datasets: NSL-KDD, and CICIDS2017, achieving reduced latency and better detection accuracy. The only downside the system faces is that the possibility of nodes' failure is not considered.

Using Machine Learning, a framework iFogLearn++ is proposed to process data streams on the Fog layer [15]. The framework when used along with Random Forest and Decision Trees, can detect DDoS attacks efficiently and classify them. For this purpose, BAIOT and BoTIOT datasets and a scalable solution was achieved with 96.5% accuracy. A secure algorithm with energy-efficient solution is proposed using a scheduling module and blockchain [21]. The algorithm is able to secure the mobile Fog-Cloud framework against malicious data, although no real data testing has been conducted.

As it can be observed from Table 2, the type of solutions is mapped with their respective drawbacks. The Swarm-based optimization techniques provide better output for multiple metrics, but there is an increase in training time. The Machine Learning models instead provide accurate results in larger processing times. Also, ML models are not adaptive to changes or irregularities in the dataset. Instead, Deep Learning methods are capable of handling huge datasets as well as anomalies. Hence, authors have used DL based ANN model to tackle the above-mentioned issues in integrated cloud fog environment and offer the secure services for IoT applications. There are existing solutions on the integrated model, although low in frequency where DL has been deployed. Although each of these frameworks has its own limitations which can be rectified to achieve a more efficient model. For the Intrusion detection model, in existing solutions, training and testing are both deployed on the fog layer, increasing the time and resources required by the secure framework [17,24]. For this purpose, authors aim for a lightweight secure integrated model and propose such a model in this paper.

3. Proposed deep learning and baseline techniques for attack detection

The need to create intelligent and learning algorithms led to the creation of Machine and Deep Learning models. Machine Learning enables the system to improve through experiences. The algorithms are taught from the training set and training set to be able to predict the results when new data is fed. Deep Learning techniques are learning systems inspired by neurons, which may be supervised or unsupervised and capable to creating relations among various types of data [9,25,26]. ML and DL have been used quite a lot for securing data for quite some time now. The need to use DL on the framework has already been discussed in Section 1. There are quite a lot of Machine Learning and Deep Learning algorithms and among them, a few have been selected to implement. It is of utmost importance to select an appropriate technique for the detection of security attacks. DL is gaining attention recently as it required the least human interaction, while ML provides more accuracy. To implement such algorithms, the first step is to preprocess the data to filter only the required attributes. Later, specific ML and DL techniques are deployed on the environment simulated. This section elaborates on the cloud-fog-IoT framework used to implement various methods. Following that, the used techniques are discussed.

3.1. Proposed framework

Our proposed framework consists of cloud, fog and IoT environment as shown in Fig 3. The IoT or end layer consists of sensors and user devices, where the requests are generated from. As discussed in Section 1, these devices are not resource-extensive. To overcome this challenge, cloud computing provides a promising solution by alleviating the resource limitations imposed by the layer. However, this computing paradigm is intrinsically unsuitable for real-time applications which are latency-sensitive in nature. An emerging paradigm, called Fog computing is capable of providing a substantial number of resources to assist the working of the proposed cloud-fog-IoT framework. Now, IoT devices are connected via the internet, which makes them exposed to potential attackers to sniff into their connections and steal crucial information. Moreover, these devices can be further exploited to launch various attacks by malicious users.

Nevertheless, the data generated by IoT-enabled applications is always on the radar of attackers causing a threat to privacy and security concerns. Hence, it becomes vital to preserve the security of voluminous data generated at the IoT layer by incorporating apt security mechanisms. AI-based techniques are on the rise to detect anomalies in requests. This can be hence used in enforcing security to the data emerging from IoT. But, due to the resource-constrained nature of these devices, AI-based solutions are difficult to incorporate at this layer as they thrive for space and computational capabilities for model training. Therefore, fog nodes assist IoT devices by providing a considerable amount of computation, communication and caching capabilities. Henceforth, our work proposes a Deep Learning enabled IoT attack detection framework, which is implemented and executed at fog layer.

The proposed DL model has two phases: training and detecting. The training of the model is carried out at the cloud layer because of the presence of a large number of resources. Whereas the detection mechanism is deployed at the fog layer. The smart gateway devices are equipped to tap the incoming network traffic, learn the features by processing it and then further sending the features to upper layer. The data travels from IoT to the fog nodes through gateways. The DL detection mechanism is implemented on this layer. The ML and DL algorithms implemented on the framework are described in the following sections.

3.2. Artificial neural network

The human brain is made up of neural networks, consisting of billions of neurons. These neurons carry information and extract patterns in information. Artificial Neural Networks are networks made from learning cells named artificial neurons, which mimic the way a human brain behaves as shown in Fig. 4.

Each neuron is named a processing element (PE). These processing elements receive input signals. Following that, there is a weight assigned to each of these PEs. The weight can be adjusted, as required, and then the information received is multiplied with the weight and combined for output. Each PEs has their activation function which is the sum of all the weighted information a neuron receives [27]. These models can be trained through experiences to be able to detect patterns in data and if required, various relations. ANN has been used to detect anomalies in cloud and fog frameworks [28].

3.3. Support vector machine

Support Vector Machines are a supervised Machine Learning technique used to solve classification and even regression problems. This helps in classifying the dataset. SVM can be used for both linear and non-linear classifications. When a problem is provided, SVM plots the data and chooses the extreme data points to chart a hyperplane. The

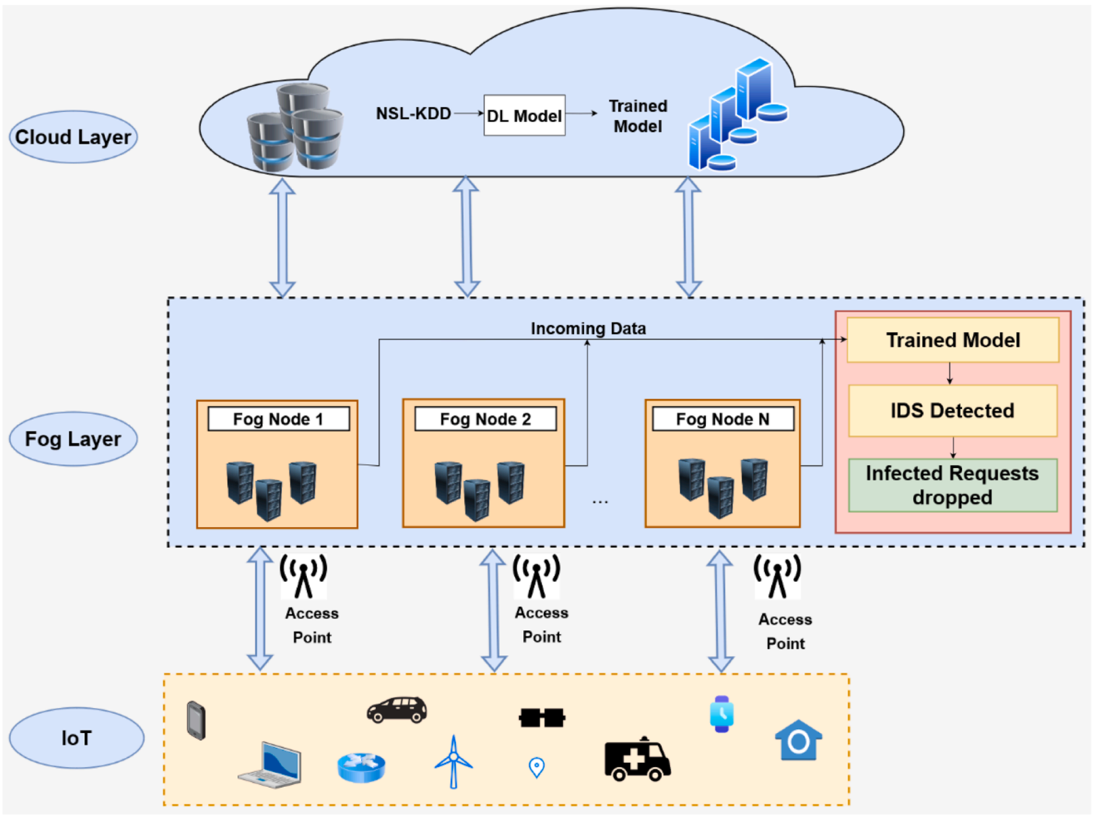

Fig. 3. Proposed Cloud-Fog framework for IoT applications.

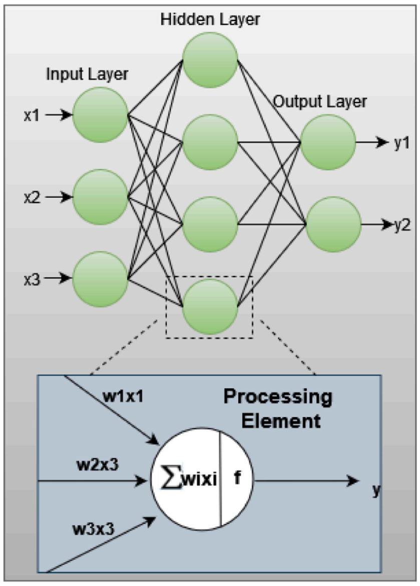

Fig. 4. Structure of Artificial Neural Network.

hyperplane is created by taking care of extreme data points, which is the decision boundary. The SVM is trained using a training set to decide on a hyperplane which optimally divides the different types of data. Following this, the SVM is tested, reducing the error with trials as shown in Fig. 5. SVM has been used for cloud and fog security in recent works with fair success when the dataset provided is huge [29,30].

3.4. Logistic regression

Logistic Regression is a Supervised Machine Learning Technique used to analyze the relationship between independent and dependent variables and predict output when a dataset of independent variables is provided. It uses a Sigmoid function to plot the way the variables are dependent on each other. It is expressed in Eq. (6). The sigmoid function expresses the independent variable into an expression of probability, within a range of 0 and 1, with respect to the dependent variable.

$$Sigmoid\ function = \frac{1}{1+e^{-x}} \tag{1}$$

Here, x is the independent variable. The Sigmoid function generates an ‘S’ curve between 0 and 1. Here, it is important to be provided with a threshold called Decision Boundary to classify the output. This value decides the way the variables will be mapped to each other. Logistic Regression solves classification problems and has been used for securing cloud and fog frameworks [31–33].

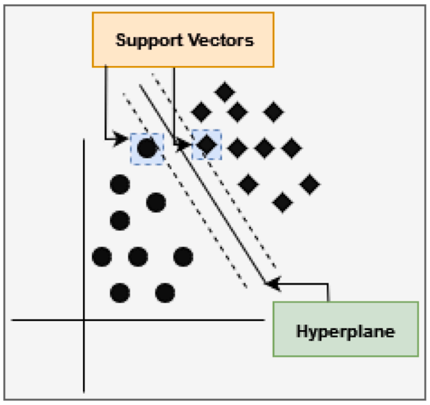

Fig. 5. Plot of Support vector Machine and hyperplane.

3.5. Decision trees

Another supervised Machine Learning method is Decision Tree, which is primarily used for classification. It is a tree structure where the decision rules provided are used to classify data. The nodes of the tree are the features of the provided dataset. The branches, which generate from a decision node are called the decision rules. Decision nodes are where the decision rules are implemented. The outcomes are represented by the leaf nodes. The structure is created on the basis of a provided dataset, which then can predict classes. Decision Trees have been used for enforcing security on cloud and similar frameworks. The features in the dataset are provided to the decision tree, which is then classified according to decision rules, which in turn can detect anomalies [34,35].

The flowchart of the proposed framework based upon integrated algorithm based upon ML and DL is shown in Fig. 6. After the framework is designed, the dataset NSL-KDD is selected, and preprocessed. The process for each of the ML and DL algorithms have been discussed in Section 3. Subsequently, each of the learning methods is implemented and compared using the performance metrics.

4. Performance metrics

An algorithm for security can be evaluated about its performance using a few parameters. These parameters describe the efficiency of the algorithm, in various ways. There are a few terms to note before discussing the performance metrics. These are: True Positive (T_P), True Negative (T_N), False Positive (F_P) and False Negative (F_N). True Positive refers to the attacks detected correctly and True Negative means that the data not detected by the algorithm are legitimate. Whereas, False positive refer to the data incorrectly marked as an anomaly, while it is not. And False negative means that the data the algorithm classified as non-anomalous is an attack. Several QoS parameters are exist to measure the performance of proposed scheme like accuracy, network failures, F1 score, Precision, fault tolerance, communication overhead etc. but authors have considered only significant parameters for the performance evaluation such as Accuracy, Precision, Recall, F1 Score and False Alarm rate. Each of the metrics’ importance is presented in this section. Accuracy refers to the ability of the model to detect correctly and False Alarm rate denotes the rate at which the model mistakenly detected attacks. Among the selected metrics, there exists a conflict between Recall and precision. That is, Recall and Precision are inversely related as Precision. This is because a high Precision value means that the framework will wrongly label normal traffic as an attack, in rare instances. And a high Recall value determines that the model will seldom let a suspicious data as a normal request. The F1 score is useful if the data is imbalanced, and a high F1 score means that the model can attain high values for both Precision and Recall, instead of the tradeoff. The performance metrics used in this paper to analyze the algorithms are as follows:

4.1. Accuracy

After learning, a DL algorithm is implemented on a dataset. The accuracy of an algorithm denotes how accurately the attacks are detected correctly in the given framework with an unseen dataset. It can be calculated using True Positive, True Negative, False Positive and False Negative values as shown in Eq. (2). Hence accuracy not only takes into consideration the number of attacks detected but also the attacks wrongly detected.

$$Accuracy = \frac{(T_P + T_N)}{T_P + T_N + F_P + F_N} \tag{2}$$

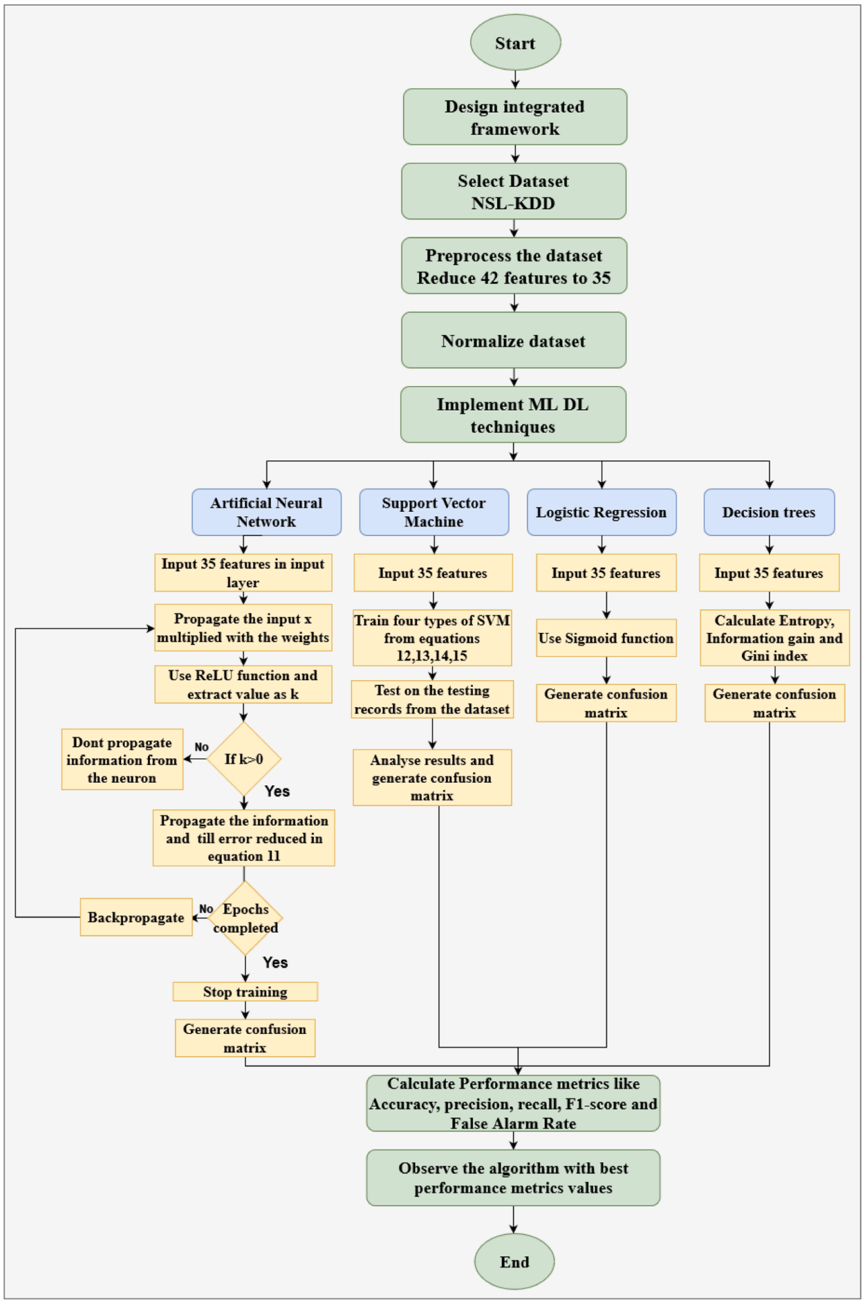
Start
Design integrated framework
Select Dataset NSL-KDD
Preprocess the dataset Reduce 42 features to 35
Normalize dataset
Implement ML DL techniques
Artificial Neural Network
Support Vector Machine
Logistic Regression
Decision trees
Input 35 features in input layer
Propagate the input x multiplied with the weights
Use ReLU function and extract value as k
If k>0
No
Dont propagate information from the neuron
Yes
Propagate the information and till error reduced in equation 11
Epochs completed
No
Backpropagate
Yes
Stop training
Generate confusion matrix
Input 35 features
Train four types of SVM from equations 12,13,14,15
Test on the testing records from the dataset
Analyse results and generate confusion matrix
Input 35 features
Use Sigmoid function
Generate confusion matrix
Input 35 features
Calculate Entropy, Information gain and Gini index
Generate confusion matrix
Calculate Performance metrics like Accuracy, precision, recall, F1-score and False Alarm Rate
Observe the algorithm with best performance metrics values
End

Fig. 6. Flowchart of the proposed framework.

4.2. *Precision*

This measure is used to calculate the ratio of correct positive detections made to the total positive detections made. It calculates how much precise is the algorithm among all the data classified as malicious. The Precision of an algorithm is described in Eq. (3).

$$Precision = \frac{T_P}{T_P + F_P} \tag{3}$$

4.3. *Recall*

As shown in Eq. (4), the Recall of an algorithm presents the ratio of correctly detected malicious data as opposed to the actual total number of attacks, which might have been wrongfully marked as False.

$$Recall = \frac{T_P}{T_P + F_N} \tag{4}$$

4.4. *F1 score*

In some instances, F1 Score is a better way to analyze an algorithm. This is when the False Negative and False Positive are not the same. It is the weighted average of Precision and Recall.

$$F1_S = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

4.5. *False alarm rate*

False Alarm Rate ($F_{(AR)}$) is the rate of wrongly classified attacks to the actual number of legitimate data requests.

$$F_{(AR)} = \frac{F_P}{T_N + F_P} \tag{6}$$

5. Experimental results and discussion

5.1. Dataset

The dataset used is NSL-KDD [36,37], which is an improvement above KDD. There are no redundant records in the more efficient newer dataset. It has been used quite a lot in research to evaluate IDs. It is available in a few formats, such as CVV, ARFF and TXT. The CSV format file has been used here. It consists of 42 features such as protocol type, service, and dst_host_server_rate. The dataset consists of five categories of attacks, namely DoS, user to root, remote to local, probe attacks and normal category. We aim to detect Ping of Death, Teardrop, Land Attacks and UDP Flood attacks (DDoS). With that intention, NSL-KDD is a situation-appropriate dataset. This is because the dataset has the features to detect the DDoS attacks. The features of the dataset are presented in Table 3.

The dataset comprises four sub-datasets: KDDTest+, KDDTrain+, KDDTest-21 and KDDTrain+_20Percent. The KDDTest+ and KDDTrain+ are testing and training sets respectively. The KDDTrain+_20Percent is the subset of KDDTrain+ comprising 20% of the original set. On the other hand, KDDTest-21 is the subset of KDDTest+ and it consists of all the traffic which is not difficult in nature.

5.2. Preprocessing

The dataset consists of 125,973 records for training and 22,544 records for testing. These records are further differentiated based on the five categories. For training, the first category, DoS has 45,927 records, 67,343 records are normal, 995 remotes to local, 11,656 probes and 52 in the user to root category. Similarly, the testing set consists of 7458 records for DoS, 9711 records for normal, 2421 records for remote to local, 2754 records for probe attacks and 200 records in the user-to-root category. It has first been preprocessed to extract useful features. As a result of preprocessing, out of 42, only 35 features are received in the final dataset. This was conducted using a correlation matrix. Following the preprocessing, the dataset has been normalized to achieve a mean of 0 and a standard deviation in the range of −1 to 1.

Table 3
NSL-KDD dataset features.

Sr No	Feature	Sr No	Feature	Sr No	Feature
F1	Duration	**F15**	Su attempted	**F29**	Same srv rate
F2	Protocol type	**F16**	Num root	**F30**	Diff srv rate
F3	Service	**F17**	Num file creations	**F31**	Srv diff host rate
F4	Flag	**F18**	Num shells	**F32**	Dst host count
F5	Source Bytes	**F19**	Num access files	**F33**	Dst host srv count
F6	Destination Bytes	**F20**	Num outbound cmds	**F34**	Dst host same srv rate
F7	Land	**F21**	Is host login	**F35**	Dst hist diff srv rate
F8	Wrong fragment	**F22**	Is guest login	**F36**	Dst host same src port rate
F9	Urgent	**F23**	Count	**F37**	Dst host srv diff host rate
F10	Hot	**F24**	Srv count	**F38**	Dst host serror rate
F11	Number failed logins	**F25**	Serror rate	**F39**	Dst host srv serror rate
F12	Logged In	**F26**	Srv Serror rate	**F40**	Dst host rerror rate
F13	Num compromised	**F27**	Rerror rate	**F41**	Dst host srv rerror rate
F14	Root Shell	**F28**	Srv rerror rate	**F42**	Class label

5.3. ANN based proposed algorithm

Artificial Neural Network is used with the help of numpy, pandas and tensorflow libraries. There are three layers in the designed ANN with input layers with 35 nodes, the first hidden layer with 12 nodes, the second hidden layer with 8 nodes and the output layer with 1 node. The two hidden layers are using ReLU (Rectified Linear Unit) as their activation function, and the output layer is using sigmoid as its activation function. The ReLU function is a linear activation function. The function is present in the library numpy. After 50 Epochs, the ANN is trained and tested and the performance metrics are noted using a confusion matrix.

The pseudocode for ANN used is presented in Algorithm 1.

Line 1–3: To start with the algorithm of the secure model, the NSL-KDD dataset has been used. This dataset is first preprocessed to create a new dataset with correlating features. As a result, a new dataset is named here as *D'*, which is the result dataset. The features in *D'* are not only corelated but also possess values for the attacks we aim to detect. Line 4–5 initialized with *D'* as the input. The error as provided in Eq. (11) is initialized as zero. Error at a node is obtained by using desired value and the actual value (Eq. (11)). The algorithm will be run for 50 epochs as shown in line 6. In an epoch, all the three layers of the model are visited. The input layer consists of 42 nodes. Following that, the first hidden layer consists of 35 nodes; and the second hidden layer comprises of 12 nodes. Consecutively, the output layer has 1 node. Line 10–22 represent the data transferred from a node *i* in layer *m* to node *r* in layer (*m* + *1*), is assigned a weight W_{ri}. The value in the node r is represented as A_r. It is measured as the summation of all the weighted data from the nodes in previous layer. This value obtained at a neuron node is then used to calculate the ReLU value using Eq. (10). If the value obtained is larger than 0, it is propagated ahead to the next node. The error is updated at the end of each epoch, which will be used for next iteration. The resultant model is trained and hence deployed on fog future traffic.

The input layer receives the dataset as X_i and multiplies it by weight W_i to be propagated to the succeeding nodes. The weights W_i is decided prior to the implementation and then improved each iteration through backpropagation. The structure is shown in Fig 6.1. Hence node 1 in the first hidden layer would receive data from the first layer as:

$$A_1 = X_1W_{11} + X_2W_{12} + X_3W_{13} + X_4W_{14} + \ldots + X_j\ W_{1j} + B \tag{7}$$

A_1 is the neuron in the first hidden layer and B is the bias of a neuron. The weight of the data forwarded from node 1 in the input layer to node 1 in the first hidden layer is W_{11}. Similarly, the weight of the data

Algorithm 1
Secure Model with ANN.

```
Input: $NSL-KDD$ dataset with features and weights
Output: Trained model for QoS parameters
1.  NSL-KDD imported
2.  Preprocess (NSL-KDD)
3.    $D'$ ← Dataset trimmed to 42 features
4.  ANN ($D'$)
5.  Error $E$ initialized to zero: (Eq. (11))
6.  for epoch $e = 1$ to 50 do
7.    for layer $k = 1$ to 3 do
8.      $t$ ← number of input nodes in layer $(m-1)$
9.      for neuron $r = 1$ to $n$ do
10.     $W_{rt}$ ← weight for input from node $t$ to $r$
11.     $A_r$ ← value in node $r$
12.     $A_r = \sum x_i * W_{ri}$: where $i = 1$ to $t$
13.     Calculate ReLU function value (Eq. (10))
14.     if ReLU value > 0
15.       Propagate value; edit $E$ (Eq. (11))
16.     else
17.       continue
18.     end for
19.   end for
20.   Backpropagate $E$
21.   end for
22. end
```

propagated from a node in the input layer to node j would be W_{jl}. Here, W_{jl} represents the weight of the data in node j of layer l. In this way, the propagation of data among the layers can be observed. The data from the input layer when multiplied with the weights can be expressed in matrices to achieve the input to individual neurons on the first hidden layer. Let matrix M consists of all the data, and matrix N consists of the weight. j is the number of input nodes and i is the number of nodes in the first hidden layer.

$$M * N = Neurons\ in\ first\ hidden\ layer = A \tag{8}$$

$$\begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1j} \\ W_{21} & W_{22} & \cdots & W_{2j} \\ \vdots & \vdots & \cdots & \vdots \\ W_{i1} & W_{i2} & \cdots & W_{ji} \end{bmatrix} * \cdot \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_i \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_j \end{bmatrix} = A \tag{9}$$

Each row in the resultant matrix A is the input value each of the first hidden layer neurons receives. The two hidden layers run the ReLU as their activation function. The activation function determines the output of the individual PEs. It is used as it helps in overcoming the gradient problem and trains the model faster. It is defined as in Eq. (10). And the output of the hidden neurons can be expressed as in Eqs. (8) and 9.

$$f(x) = max(0, x) \tag{10}$$

The proposed model of ANN uses backpropagation to make the weights more efficient. That is, if there is an error in the prediction, the weights are improved over time to reduce the error as much as possible. The error can be calculated from Eq. (11). Here x_i is the information received and T_i is the desired value.

$$Error = (1/2)\sum_i (x_i - T_i)^2 \tag{11}$$

The confusion matrix is achieved for ANN by calculating the difference in the actual positive and negative data records against the predicted positive and negative records. This gives a view into the capability of ANN to detect the anomalies correctly and not raise False Alarm rates. The values can be observed in Table 4. Using these values, the performance metrics can be calculated.

5.4. SVM

A Support Vector Machine is implemented to classify the data. The framework used is the cloud-fog-IoT and the data is generated at the IoT layer, to be detected at the Fog layer. Hence the algorithms are implemented on the fog nodes. It has been taken care of that the data might not be efficiently classified using SVM. Hence, 4 types of SVM are used: Linear kernel, polynomial kernel, sigmoid kernel and RBF kernel. These are expressed in Eqs. (12), 13, 14 and 15 simultaneously. In all these equations, x_i and x_j are the data to be classified and $f(x_i,x_j)$ is the decision function. Out of all these, the RBF kernel performs the best with 99.1903% efficiency in the training set and 99.1586% efficiency in the testing set, as shown in Table 5. The confusion matrix of SVM is represented in Table 6.

$$f(x_i, x_j) = sum(x_i * x_j) \tag{12}$$

$$f(x_i, x_j) = (1 + x_i * x_j)^d \tag{13}$$

$$f(x_i, x_j) = tanh(c + \alpha x_i^T x_j) \tag{14}$$

Table 4
Confusion Matrix for ANN.

	Actually positive	Actually Negative
Predicted Positive	17,363	150
Predicted Negative	68	20,211

Table 5
Comparison of various Support Vector Machine algorithms on the NSL-KDD dataset.

SVM kernel type	Training set efficiency	Testing set efficiency
Linear Kernel	95.9697%	96.0150%
Polynomial kernel	99.1472%	99.1083%
Sigmoid Kernel	85.4334%	85.3196%
RBF kernel	99.1903%	99.1586%

Table 6
SVM based confusion matrix.

	Actually positive	Actually Negative
Predicted Positive	17,345	168
Predicted Negative	150	20,129

$$f(x_i, x_j) = exp\left(\left(-\gamma\left|\left|x_i - x_j\right|\right|\right)^2\right) \tag{15}$$

Here d is the degree of the function and the γ can be calculated as $\gamma = \frac{1}{2}\sigma^2$ and the value of γ varies from 0 to 1.

5.5. Logistic regression

The next algorithm used on the refined dataset is Logistic Regression, using the sklearn library. The function for Logistic regression is represented in Eq. (16). Here, x is the input value, b is the bias, and c is the coefficient for the input value x. The performance metrics are extracted from the confusion matrix for the algorithm. This can be observed in Table 7 and the values for T_P, T_N, F_P, and F_N can be used for calculating the respective performance metrics.

$$y = \frac{e^{(b+c*x)}}{1 + e^{(b+c*x)}} \tag{16}$$

5.6. Decision tree

The final algorithm used here is Decision Tree. The sklearn library is used for the Decision Tree and a resultant tree is achieved by classifying the features, along with the confusion matrix. The equations for this classifying structure can be expressed as in Eq. (17) for Entropy, Eq. (18) for information gain and Eq. (19) for the Gini index. The confusion matrix of the Decision Tree is represented in Table 8.

$$Entropy(x) = -p_y log_2 p_y - p_n log_2 p_n \tag{17}$$

$$Information\ Gain = Entropy(x) - \{E_A * Avg_W\} \tag{18}$$

$$Gini\ Index = 1 - \sum_i p_i^2 \tag{19}$$

Here p_y = probability of yes and p_n= probability of no. E_A is the entropy of all the features and Avg_W is the Weighted Average. Gini Index is the measurement of the purity of the information as classified in the Decision tree.

5.7. Comparison of the algorithms

Table 9 presents the performance metrics of all the four algorithms. The accuracy, precision, recall and F1-score of all four algorithms are

Table 7
LR based Confusion Matrix.

	Actually positive	Actually Negative
Predicted Positive	16,432	1081
Predicted Negative	729	19,550

Table 8
Confusion Matrix for Decision Tree.

	Actually positive	Actually Negative
Predicted Positive	17,234	279
Predicted Negative	487	19,792

Table 9
Comparison of algorithms according to performance metrics.

Algorithm	Accuracy	Precision	Recall	F1-score	False Alarm rate
ANN	0.994232	0.992633	0.996647	0.994636	0.007396
SVM	0.991586	0.991723	0.992603	0.992163	0.008284
Logistic Regression	0.952106	0.947603	0.964051	0.955757	0.053306
Decision Tree	0.979731	0.986099	0.975985	0.981016	0.013758

compared with each other as shown in Fig 7, Fig 8, Fig 9, Fig 10 and Fig 11. Finally, all the algorithms are compared according to all the performance metrics in Fig 12. From the figures and tables, it is observed that ANN performs the best at detecting attacks in the cloud-fog-IoT framework with 99.4232% accuracy, 99.2633% precision, 99.6647% recall, 99.4636% F1-score and 0.7396% False Alarm Rate, which is least among all the four algorithms. To validate the robustness of the model, we observe the variation in the data fed. For our algorithm, 25% of the dataset is used to train ANN iteratively, until the dataset as a whole has been used to train the model. The accuracy for all these cases is found to be in the range of 99.28% to 99.42%. Hence, the accuracy of the model is consistent in nature even when different sections of the dataset have been to test our model. Subsequently, it can be deduced that our proposed model is robust in nature.

It can be interpreted that our model is capable of detecting DDoS attacks with the more accuracy as shown in Fig. 7. The proposed framework predicts the True Positive and True Negative most accurately, when compared with other models. It would be useful in a scenario where it would be important to detect legitimate and illegitimate requests correctly. It would be a an advantage of proposed model to apply in IoT applications specially healthcare sector. From Fig 8, it can be deduced that the proposed model is most precise at detecting legitimate attacks and that a very small amount of the suspicious requests was counted as legitimate. This would be an important metric in healthcare where the aim would be to drop suspicious requests so that each medical information is untouched. The next metric, for Recall (99.66%) too our model has a good score compared to the other models. This denotes that the framework is discarding legitimate requests as an attack in a rarest of instances. The F1 score (99.46%) denotes that the ANN model is capable predicting the attacks accurately. It also means that the model tries its best so that neither normal requests are discarded as suspicious, nor illegitimate data are passed as legitimate. The tradeoff between the two

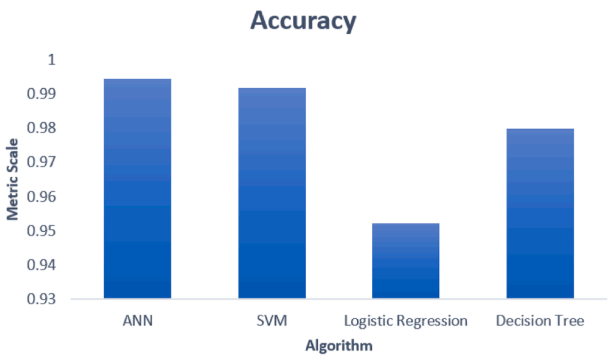

Fig. 7. Accuracy of the algorithms.

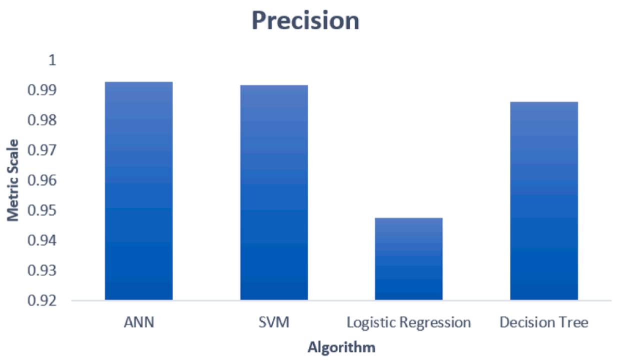

Fig. 8. Precision of the algorithms.

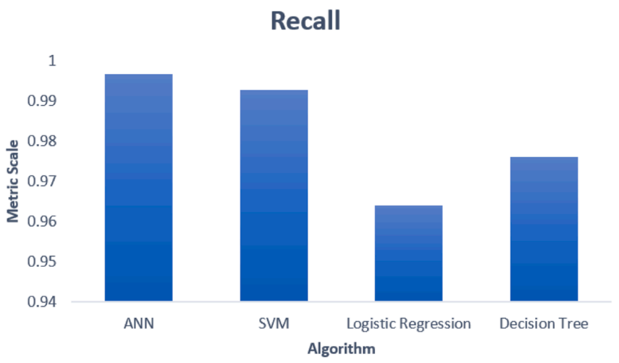

Fig. 9. Recall of the algorithms.

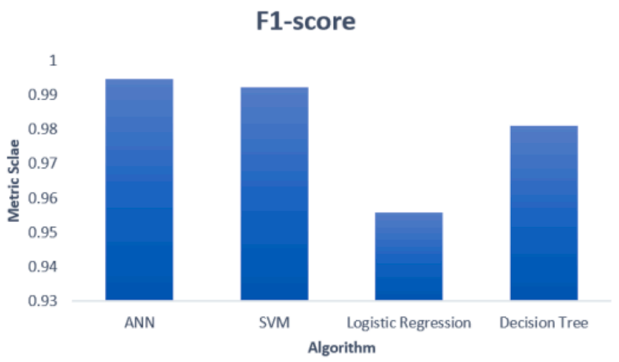

Fig. 10. F1-score of the algorithms.

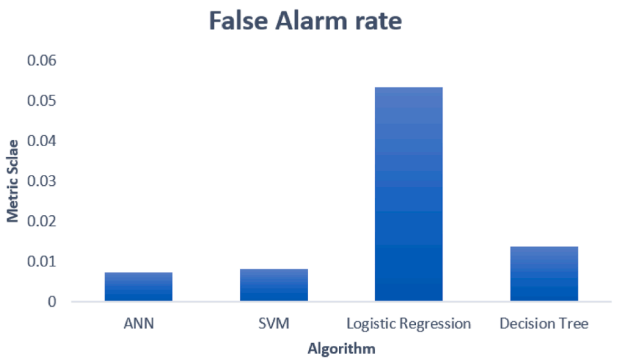

Fig. 11. False Alarm Rate of the algorithms.

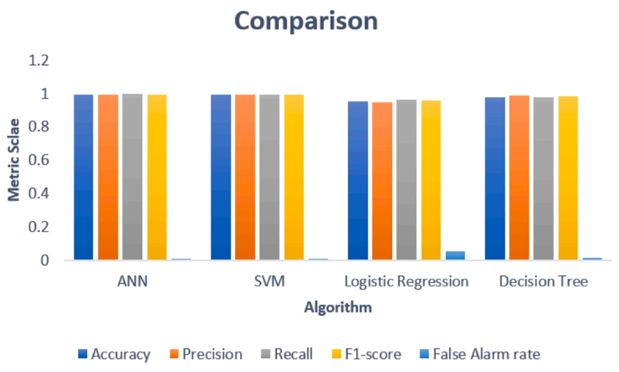

Fig. 12. Performance metrics comparison of the algorithms.

(Precision and Recall) has also been effectively taken care of. Finally, a lowest False Alarm rate (0.7396%) means that the model wrongly detects a request as an attack, in the smallest amount. Our model is followed by SVM according to the metrics.

Conclusion and future work

Security is a major issue in integrated cloud fog environment due to its dynamic and hazardous environments and leads to attracting illegitimate users. Artificial Intelligence-based several approaches have been developed by authors in the past to detect vulnerabilities, but fail to either secure the system or secure the framework while overloading the fog layer. Hence, we have proposed a lightweight cloud fog framework for secure IoT applications and it is integrated with deep learning-based techniques named Artificial Neural Network to detect attacks on IoT data. The algorithms were all implemented on the fog layer to reduce the workload on the IoT which already has the least computational power. The efficacy of the proposed approach is analyzed on the basis of five performance metrics and compared with state of art algorithms. ANN has been observed to be performing most efficiently on the NSL-KDD dataset with the highest accuracy of detecting accurately (99.42%), recall (99.66%), precision (99.26%), F1 score (99.46%) and least False Alarm rate (0.7396%). Hence ANN integrated model generated the least false attack alarms on the huge dataset, with recall and precision both better than the baseline methods as seen in Table 9. The experimental results proved that the proposed ANN surpasses the other ML-based techniques and provides the guarantee to reduce the anomalies. The model was implemented on a synthetic dataset. Thus, for future work, the framework can be implemented for IoT applications in a real-time scenario. Also, the proposed framework can be extended to test the other QoS parameters like network failures, fault tolerance, communication overhead etc. along with resource constraints of the environment.

Author statement

- The corresponding author is responsible for ensuring that the descriptions are accurate and agreed by all authors.
- Authors may have contributed in multiple roles like

Ananya Chakraborty: Writing - Original Draft, Conceptualization, and Methodology,

Mohit Kumar: Software, Experimental/Simulation work, Results and outcome, and Visualization

Nisha Chaurasia: Review & Editing

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The authors whose names are given in this article certify that they have no affiliations with or involvement in any organization or entity with any financial interest, or non-financial interest in the subject matter or materials discussed in this manuscript. Currently there is no concurrent submission of this article by any of authors.

Data availability

Data will be made available on request.

References

[1] Schiller E, Aidoo A, Fuhrer J, Stahl J, Ziörjen M, Stiller B. Landscape of IoT security. Comput Sci Rev 2022;44:100467. https://doi.org/10.1016/j.cosrev.2022.100467.4.

[2] Chakraborty A, Kumar M, Chaurasia N, Gill SS. Journey from cloud of things to fog of things: survey, new trends, and research directions. Softw - Prac. Exp 2022; (August). https://doi.org/10.1002/spe.3157.

[3] Sangaiah AK, Javadpour A, Ja'fari F, Pinto P, Zhang W, Balasubramanian S. A hybrid heuristics artificial intelligence feature selection for intrusion detection classifiers in cloud of things. Cluster Comput 2022;26(1):599–612. https://doi.org/10.1007/s10586-022-03629-9.

[4] Alli AA, Alam MM. SecOFF-FCIoT: machine learning based secure offloading in Fog-Cloud of things for smart city applications. Internet of Things 2019;7(2019):100070. https://doi.org/10.1016/j.iot.2019.100070.

[5] Li Y, Li H, Xu G, Xiang T, Huang X, Lu R. Toward secure and privacy-preserving distributed deep learning in fog-cloud computing. IEEE Internet Things J 2020;7(12):11460–72. https://doi.org/10.1109/JIOT.2020.3012480.

[6] Verma P, Tiwari R, Hong WC, Upadhyay S, Yeh YH. FETCH: a deep learning-based fog computing and iot integrated environment for healthcare monitoring and diagnosis. IEEE Access 2022;10:12548–63. https://doi.org/10.1109/ACCESS.2022.3143793.

[7] Tuli S, Basumatary N, Buyya R. EdgeLens: Deep Learning based object detection in integrated IoT, fog and cloud computing environments. In: 2019 4th Int. Conf. Inf. Syst. Comput. Networks, ISCON 2019; 2019. p. 496–502. https://doi.org/10.1109/ISCON47742.2019.9036216.

[8] Sarkar I, Kumar S. Deep learning-based energy-efficient computational offloading strategy in heterogeneous fog computing networks. J Supercomput 2022;78(13):15089–106. https://doi.org/10.1007/s11227-022-04461-z.

[9] Samy A, Yu H, Zhang H. Fog-based attack detection framework for internet of things using Deep Learning. IEEE Access 2020;8:74571–85. https://doi.org/10.1109/ACCESS.2020.2988854.

[10] Mathew A, Deepu NEI, Mohan M. Intelligent edge security with dynamic task offloading in fog environment. In: Proceeding of the 4th International Conference on Communication Electronic System ICCES 2019; 2019. p. 367–72. https://doi.org/10.1109/ICCES45898.2019.9002417.

[11] Ahanger TA, Tariq U, Ibrahim A, Ullah I, Bouteraa Y, Gebali F. Securing IoT-empowered fog computing systems. Mach Learn Perspect," Math 2022;10(8):1–20. https://doi.org/10.3390/math10081298.

[12] Chang V, et al. A survey on intrusion detection systems for fog and cloud computing. Futur Internet 2022;14(3). https://doi.org/10.3390/fi14030089.

[13] Sham EE, Vidyarthi DP. CoFA for QoS based secure communication using adaptive chaos dynamical system in fog-integrated cloud. Digit Signal Process A Rev J 2022;126:103523. https://doi.org/10.1016/j.dsp.2022.103523.

[14] Roy S, Li J, Bai Y. A two-layer fog-cloud intrusion detection model for IoT networks. Internet of Things (Netherlands) 2022;19(May):100557. https://doi.org/10.1016/j.iot.2022.100557.

[15] Sharifi A, Goli-Bidgoli S. IFogLearn++: a new platform for fog layer's IoT attack detection in critical infrastructure using machine learning and big data processing. Comput Electr Eng 2022;103(July 2020):108374. https://doi.org/10.1016/j.compeleceng.2022.108374.

[16] Hameed SS, et al. A hybrid lightweight system for early attack detection in the IoMT fog. Sensors 2021;21(24). https://doi.org/10.3390/s21248289.

[17] Khan F, Jan MA, Alturki R, Alshehri MD, Shah ST, ur Rehman A. A secure ensemble learning-based fog-cloud approach for cyberattack detection in IoMT. IEEE Trans Ind Informatics 2023:1–9. https://doi.org/10.1109/tii.2022.3231424.

[18] Butt UA, et al. A review of machine learning algorithms for cloud computing security. Electron 2020;9(9):1–25. https://doi.org/10.3390/electronics9091379.

[19] Jangjou M, Sohrabi MK. A comprehensive survey on security challenges in different network layers in cloud computing. Arch Comput Methods Eng 2022;29(6):3587–608. https://doi.org/10.1007/s11831-022-09708-9.

[20] Ometov A, Molua OL, Komarov M, Nurmi J. A Survey of security in cloud, edge, and fog computing. Sensors 2022;22(3):1–27. https://doi.org/10.3390/s22030927.

[21] Razaque A, Jararweh Y, Alotaibi B, Alotaibi M, Hariri S, Almiani M. Energy-efficient and secure mobile fog-based cloud for the Internet of Things. Futur Gener Comput Syst 2022;127:1–13. https://doi.org/10.1016/j.future.2021.08.024.

[22] R. Du, Y. Li, X. Liang, and J. Tian, "Support vector machine intrusion detection scheme based on cloud-fog collaboration," pp. 431–440, 2022.

[23] Pirozmand P, Ghafary MA, Siadat S, Ren J. Intrusion detection into cloud-fog-based IoT networks using game theory. Wirel Commun Mob Comput 2020;2020. https://doi.org/10.1155/2020/8819545.

[24] Diro A, Chilamkurti N. Leveraging LSTM networks for attack detection in fog-to-things communications. IEEE Commun Mag 2018;56(9):124–30. https://doi.org/10.1109/MCOM.2018.1701270.

[25] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016.

[26] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436–44. https://doi.org/10.1038/nature14539.

[27] Antony C, P M. Investigation of security breaches due to resource sharing in virtual machine migration using hybrid ant colony optimization with ANN. EAI Endorsed Trans Cloud Syst 2018:174150. https://doi.org/10.4108/eai.7-6-2022.174150.

[28] Jeniffer JT, Chandrasekar A. Optimal hybrid heat transfer search and grey wolf optimization-based homomorphic encryption model to assure security in cloud-based IoT environment," *Peer-to-Peer Network*. App 2022;15(1):703–23. https://doi.org/10.1007/s12083-021-01263-7.

[29] Thenappan S, Valan Rajkumar M, Manoharan PS. Predicting diabetes mellitus using modified support vector machine with cloud security. IETE J Re. Nov. 2022; 68(6):3940–50. https://doi.org/10.1080/03772063.2020.1782781.

[30] MM GA, S JNK, R UM, TF MR. An efficient SVM based DEHO classifier to detect DDoS attack in cloud computing environment. Comput Netw 2022;215(May 2021): 109138. https://doi.org/10.1016/j.comnet.2022.109138.

[31] Wang S, et al. HEALER: homomorphic computation of ExAct Logistic rEgRession for secure rare disease variants analysis in GWAS. Bioinformatics 2016;32(2): 211–8. https://doi.org/10.1093/bioinformatics/btv563.

[32] Besharati E, Naderan M, Namjoo E. LR-HIDS: logistic regression host-based intrusion detection system for cloud environments. J Ambient Intell Humaniz Comput 2019;10(9):3669–92. https://doi.org/10.1007/s12652-018-1093-8.

[33] Chen H, et al. Logistic regression over encrypted data from fully homomorphic encryption. BMC Med Genomics 2018;11(Suppl 4). https://doi.org/10.1186/s12920-018-0397-z.

[34] A.B. Saxena, D. Sharma and D. Aggarwal, "Trust prediction tree : use of decision tree in cloud scenarios," vol. 7, no. 5, pp. 271–278, 2022.

[35] Alex S, Dhanaraj KJ, Deepthi PP. Private and energy-efficient decision tree-based disease detection for resource-constrained medical users in mobile healthcare network. IEEE Access 2022;10:17098–112. https://doi.org/10.1109/ACCESS.2022.3149771.

[36] NSL-KDD Dataset. Accessed: Feb. 15, 2019. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html.

[37] Tavallaee M, Bagheri E, Lu W, Ghorbani A. A detailed analysis of the KDD CUP 99 data set. In: Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA); 2009.