© May 2023| IJIRT | Volume 9 Issue 12 | ISSN: 2349-6002

Fog-based Attack Detection Framework for Internet of Things Using Deep Learning

R.Bhavani[1], M Pavithra[2], K.Umamageshwari[3], M.Vishnu Priya[4]
[1]Dept of CSE, PSRRCollege of Engineering, *Sivakasi*
[2,3,4]Computer science and engineering, PSRRCollege of Engineering, *Sivakasi*

Abstract— The number of cyber-attacks and data breaches has immensely increased across different enterprises, companies, and diligence as a result of the exploitation of the sins in securing Internet of Effects (IoT) bias. The added number of biases connected to IoT and its different protocols has led to the growing volume of zero-day attacks. Deep literacy(DL) has demonstrated its superiority in big data fields and cyber-security. lately, DL has been used in cyber-attack discovery because of its capability of rooting and learning deep features of known attacks and detecting unknown attacks without the need for homemade point engineering. still, DL can not be enforced on IoT bias with limited coffers because it requires expansive calculation, strong power and storehouse capabilities. This paper presents a comprehensive attack discovery frame of a distributed, robust, and high discovery rate to descry several IoT cyber-attacks using DL. The proposed frame implements an attack sensor on fog bumps because of its distributed nature, high computational capacity and propinquity to edge bias. Six DL models are compared to identify the DL model with the stylish performance. All DL models are estimated using five different datasets, each of which involves colourful attacks. trials show that the long short-term memory model outperforms the five other DL models. The proposed frame is effective in terms of response time and discovery delicacy and can descry several types of cyber-attacks with99.97 discovery rate and99.96 discovery delicacy in double bracket and99.65 discovery delicacy in multi-class bracket.

Keywords—element, formatting, style, styling, insert(crucial words).

INTRODUCTION

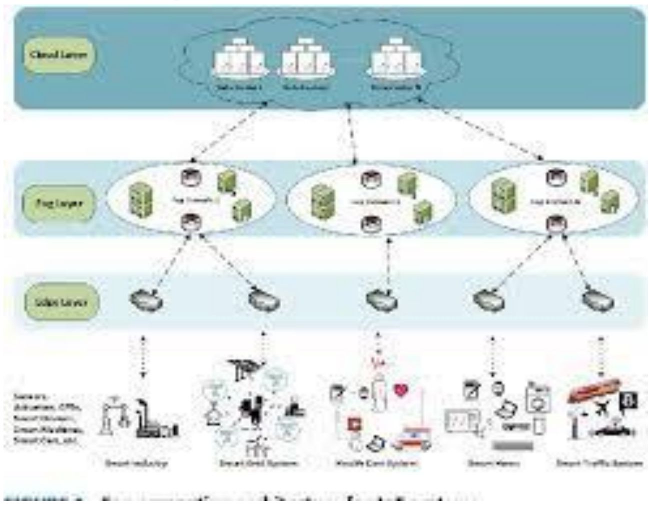
The Internet of effects (IoT) is considered a fleetly developing paradigm in the history of computing. In the once many times, IoT has immensely evolved in different technological fields. It has gathered between hundreds of billions of bias from different systems (similar as smart vehicles, smart health care, smart grid, smart home, etc.) and the internet(1). still, this confluence has redounded in numerous cyber-attacks on IoT systems because IoT integrates the digital world with the physical terrain(2). IoT security has come grueling because of the diversity, large scale, limited tackle coffers, and global availability of IoT systems. Experimenters have used machine literacy(ML) algorithms similar as decision tree(DT), arbitrary timber(RF), support vector machine(SVM), Bayesian network, and K- Means to descry network attacks, as proposed in(3). (4), still, the process requires homemade point engineering (5). attained features similar as the number of bytes transferred and entered, connection time, number of requests, and error count cannot deeply represent the pattern gest of attacks. therefore, ML is infelicitous for detecting cyberattacks and serving as a practical result in the assiduity. The stylish result to overcome the limitations of ML is to use deep literacy (DL). DL can represent data using the multiple processing layers of computational models. In addition, DL can give a deep representation of raw data and prognosticate or classify data more directly than ML because of its multilayer structure(6). still, the direct perpetration of complex DL. models on IoT bias is gruel because of the limited calculation, storehouse, and energy capabilities of IoT bias. thus, using DL for attack discovery in IoT isn't a direct way. DL has been used in numerous proposed intrusion discovery systems (IDSs) for IoT in(7),(8),(9),(10), and(11). DL has a high discovery rate (DR) in feting morphing attacks. On the one hand, DL is a important tool used to dissect huge business volumes and directly distinguish the normal and abnormal gest of different systems by rooting deep complex patterns from raw data(packets). On the other hand, the direct perpetration of complex DL models on IoT bias is unhappy due to the constrained nature of the IoT bias. numerous experimenters have used DL to descry cyber-attacks in IoT(5),(12). still, no bone has explained the perpetration of these computational and power- ferocious models on low-capacity detectors. Thanks to the development of fog computing that can prop the direct perpetration of DL models in IoT bias by processing, assaying, and storing large volumes of data on fog bumps with low quiescence and high response time(13). The idea is to move the implementation of DL from detectors in the edge subcaste to the nearest place of data sources, at which point data analysis takes little time. Fog computing extends traditional pall- grounded services so that they're at the network edge where data are generated. likewise, it provides a distributed terrain, mobility, and scalability(14). Consequently, DL can be enforced on the fog subcaste bumps where fog computing allows the perpetration and prosecution of attack discovery in a distributed, important and scalable manner. In the current work, we present a detailed frame of a distributed and robust attack discovery for IoT that's grounded on fog computing and DL. The proposed fog-grounded attack discovery frame takes a veritably short time to descry attacks and lower response time than pall- grounded attack discovery. We estimate six supervised DL models in our trials and elect the stylish one. We use four new datasets and one old dataset to estimate the performance of the DL models through expansive trials. The five datasets involving different attacks, similar as miri, DDoS, worms, exploits, sybil, billabong, prob, R2L, etc. to corroborate the capability of the proposed frame in detecting several attacks. Grounded on the results of expansive trials, the LSTM model achieves the stylish performance in attack discovery and delicacy. IoT business generated in the edge subcaste is routed to the fog subcaste via smart gateway bias. The discovery system enforced on the fog bumps classifies IoT data to descry attacks. The discovery system is controlled using a pall service to confirm its distributivity, scalability, and velocity. The proposed frame consists of four stages. The first stage aims to train the DL model and tune the hyperparameters to achieve the stylish performance. The alternate stage discusses how to apply the proposed attack discovery frame on fog bumps and how they communicate and controlled by pall service. The third stage discusses the attack discovery grounded on business analysis using the LSTM model. The fourth stage shows how to cover, estimate, and modernize the LSTM model. The proposed frame is explained in detail in Section IV. The main benefactions of this exploration are epitomized as follows of detailed frame of a robust and distributed attack discovery for IoT networks. 2) Comparison of six supervised DL models to elect the stylish model for IoT attack discovery grounded on expansive trials. 3) Evaluation of the performance of six DL models using four up- to- date IDS datasets and one old dataset. 4) Achievement of the loftiest DR and smallest false alarm rate(FAR) in comparison with three other IDSs. 5) Offer of evidence that DL has better discovery capability than ML and that it can descry several types of attacks in IoT through expansive trials. The remaining corridor of the paper are organized as follows. Section II reviews the affiliated work that used DL in attack discovery. Section III discusses the armature and rules of fog computing in IoT. Section IV shows the stages of the proposed frame in detail. Section V provides an overview of DL models used in the trials and the details of the five datasets. Section VI explains the trials and evaluation of results. Section VII presents the conclusion and future work.

II.RELATED WORK

This section discusses the state- of- the- art security approaches that use DL for attack discovery in IoT. Further-more, the capability of rooting latent features and descrying different attacks of different DL models under different network surroundings is delved. Cyber-attacks have immensely increased in the last 10 times with the rapid-fire growth of IoT bias and operations 15). bushwhackers have employed cyber-attacks to compromise thousands of IoT bias that are accessible and relaxed. For case, in 2016, several websites using DNS provider" DNS" were attacked using a DDoS attack. This attack involved several IoT bias to execute the botnet malware(15). Experimenters have proposed numerous security approaches and fabrics to alleviate specific cyber-attacks and internal attacks. still, these security approaches are fleetly compromised by new attacks(16). therefore, IoT requires a distributed security result that can constantly cover IoT bias, descry zero- day attacks, and make sound decisions. Many notorious associations similar as Face book, Yahoo!, Twitter, and YouTube, have developed numerous operations on the base of DL to cover and dissect huge volumes of data generated from billions of druggies(17). DL has been extensively used with the recent enhancement of graphical processing units(GPUs), the vacuity of big data used to train DL Models, and the actuality of important literacy algorithms. To conclude, DL has demonstrated its superiority in big data analysis, and expansive exploration has concentrated on IoT attack discovery using DL(18). An IDS for In- vehicle network security grounded on deep neural networks(DNNs) was proposed in(7). They used a pre-trained unsupervised deep belief network model to initialize the parameters of the DL network and prize the features from in- vehicular network packets. These packets were generated by bluffing In-vehicle network communication. The deep belief network model, combined with a conventional stochastic grade descent system, was used for bracket. The proposed IDS can respond to real time attacks with 98 discovery delicacy on average, and the DL model outperforms traditional ML models. Another DL- grounded security model was developed to descry vicious operations at the edge of a cellular network using mobile edge computing(8). The proposed model comported of two factors, point pre processing and vicious operation discovery machines. For the vicious operation discovery machine, a deep belief network was used for unsupervised point literacy, and a soft max function was employed for vaticination. The proposed model was enforced on 10 different datasets and the result showed that its discovery delicacy was advanced than that of soft max retrogression, SVM, DT and RF. Another study detected Android malware by automatically rooting convolutional neural features from raw opcode sequences(11). After the conversion of a sequence of opcode instructions into one- hot vectors, it was fed to the embedding subcaste. The affair from the bed- ding-dong subcaste was fed to one or further convolutional layers to extract point vectors. The point vectors were passed to a multilayer perceptron(MLP) network for bracket. The proposed model achieved advanced discovery performance with small datasets than other state-of-the-art models. Whereas, it achieved a lower discovery performance with large datasets. Reference(19) used an intermittent neural network(RNN) DL model to descry botnet gest. The authors bandied the capability of the RNN to estimate network business by detecting botnet attacks by using LSTM. The gest of connections between biases was used by the LSTM discovery model to descry botnets. Two different datasets were used for training and testing the discovery model, and another unseen dataset was employed to estimate the performance of LSTM with different connection countries. LSTM was able of detecting different botnet gest. Another exploration(5),(20) conducted attack discovery on the fog-to-effects using an LSTM-grounded deep network. The authors introduced a distributed approach using fog bumps, each of which entered initialization and updated parameters from the coordinating knot. Each fog knot trained the LSTM model on the base of the parameters entered from the coordinating knot and also transferred the weights and bias values back to the coordinating knot. latterly, the added-up parameters were calculated and returned to the fog bumps. The proposed model proved that DL models outperform traditional ML models. The authors demonstrated that distributed attack discovery is more effective and scalable than a centralized approach. Deep autoencoders, which is an unsupervised DL model that has been used by some experimenters to make point literacy for discovery models. The authors in(12) used autoencoders for unsupervised point literacy to descry network-grounded malware. Autoencoders were fed by the features acquired from cybersecurity marvels. idle features generated from autoencoders enhanced the DR of different classifiers, similar as Gaussian Naive Bayes, SVM, and Xg boost. Some studies have combined different DL models to give an ensemble literacy system for literacy and discovery improvement. A distributed attack discovery scheme has been proposed in(21). It used extreme literacy machine(ELM) classifier to classify network business at the edge calculating subcaste. likewise, it moved all expansive coffers operations similar as model training and construction to the pall subcaste using HPC cluster. The proposed scheme carried out model training used anonymized data collected from edge subcaste bias. also, the training model used by classifiers on edge waiters. The trials demonstrated that the proposed scheme achieved high delicacy in scanning, communication, and infected hosts scripts with 99, 74, and 95 independently. A malware discovery system using CNN and LSTM was presented in(22). The proposed system converted the opcode sequence of a malware train into grayscale images, and CNN and LSTM learned from these images. This system outperformed other ML styles, similar as SVM, RF and, direct k- nearest neighbour(KNN). All mentioned affiliated workshop are epitomized in Table 1.

II.FOG COMPUTING ARMATURE AND ITS ROLE IN IOT

In typical IoT armature, smart gateway bias are used to route data from the edge subcaste to upper layers(e.g., fog and pall). Data analysis requires considerable time and gests some detainments because the pall is so far from the edge subcaste(23). therefore, this process is infelicitous for sensitive data that bear a fast response. Fog computing is introduced to attack these challenges by extending the pall to be close to the data sources. Fog computing was proposed by CISCO in 2012 to break the challenges of pall computing. systems, which is divided into three layers, vide licet, edge, fog, and pall layers. The edge subcaste combines billions of miscellaneous IoT bias similar as detectors, vehicles, security cameras, smart wearable bias, smart machines and smart home appliances. This subcaste generates large data size from different places and operations. The fog subcaste is the intermediate subcaste between the edge and pall subcaste. The fog subcaste is divided into numerous connected disciplines, and each



24).Fig. 1 shows the armature of fog computing in IoT

operations, and services. The upper subcaste is the pall subcaste, which is the core subcaste comprising high-performance waiters and storehouse bias. The deployment of DL on fog bumps is useful for IoT, in several ways, including the following sensitive data analysis close to IoT bias that induce data, the quiescence between data sources and data analysis bias can be reduced, network bandwidth can be minimized, the data to be transferred to the pall and the data to be reused on fog bumps can be reused, and mobility services can be supported(25). Authors in(26), proposed a fog vehicle computing(FVC), which is a fog model that uses the available unused coffers of vehicles to produce temporary fog computing coffers for data processing. The proposed model used a pool of parking vehicles at a shopping boardwalk for calculating power. The authors explained the allocation of applicable calculation and storehouse coffers through a policy operation subcaste. also, a decision- making process was introduced to perform the needed services on the available coffers. Cisco, IDC Future Scape state that 40 of the data generated by IoT bias are anatomized on bias near the IoT bias(27). Fog bumps have been extensively used in several technologies, similar as 5G and numerous fog- to- effects operations, because of the adding computational and communication capabilities of their tackle(28). moment, security approaches can be enforced near the edge subcaste using distributed fog bumps to dissect network business and fleetly descry attacks. In the current work, we present a DL- grounded distributed attack discovery frame for IoT with low quiescence, geographical distribute ability, scalability, and high- speed response, by using the advantages of fog computing. where zt, rt, ct and ht are update, reset, cell state and hidden state of the cell. xt represents the input at time t, W represents the weight values, and b represents the bias values. GRU outperforms LSTM in some cases, especially on small datasets [37]. However, LSTM or GRU can't be used for specific tasks. GRU is faster than LSTM in training and can generalize using few data. LSTM requires voluminous data during training and consumes more time than GRU, but it provides better performance in large-scale tasks. GRU used to detect attacks in automated process control system in [38].

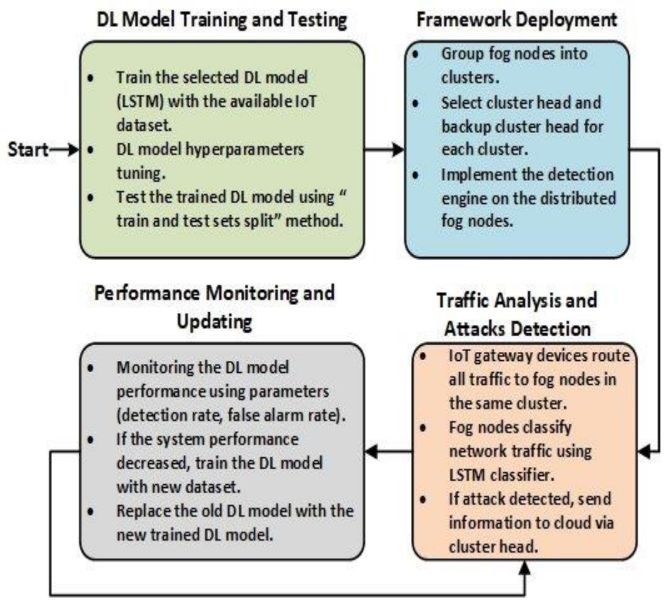

DL Model Training and Testing
Start
Train the selected DL model (LSTM) with the available IoT dataset.
DL model hyperparameters tuning.
Test the trained DL model using " train and test sets split" method.
Framework Deployment
Group fog nodes into clusters.
Select cluster head and backup cluster head for each cluster.
Implement the detection engine on the distributed fog nodes.
Performance Monitoring and Updating
Monitoring the DL model performance using parameters (detection rate, false alarm rate).
If the system performance decreased, train the DL model with new dataset.
Replace the old DL model with the new trained DL model.
Traffic Analysis and Attacks Detection
IoT gateway devices route all traffic to fog nodes in the same cluster.
Fog nodes classify network traffic using LSTM classifier.
If attack detected, send information to cloud via cluster head.

FIGURE 2: Proposed framework stages for attack detection in IoT networks

IV. PROPOSED ATTACK DETECTION FRAMEWORK

Gartner, Inc. anticipated that5.8 billion IoT bias will be in use by 2020(29). These multitudinous IoT bias located at different geographical areas induce massive quantities of data that bear rapid-fire analysis to descry attacks. therefore, centralized attack discovery infelicitous for IoT security monitoring. The proposed frame grounded on LSTM DL model enforced on distributed fog bumps, and is controlled and streamlined via the service located in the pall calculating subcaste. The proposed frame consists of four main stages, videlicet, DL model training and testing, frame emplacement, business analysis and attack discovery, and performance monitoring and updating. The stages of the proposed framework. 1) DL Model Training and Testing The selection of an applicable DL model plays a crucial part in the proposed frame's discovery delicacy and effectiveness. The DL model should be as good as the data used to train it. This stage aims to determine the stylish DL model and train it with IoT data in pall subcaste to enhance the performance of the named DL model in detecting colourful attacks. To determine the stylish DL model that full fills the conditions of the proposed frame, we conduct several trials using six different supervised DL models with five different datasets, as bandied in details in section V. The LSTM DL model achieves the loftiest DR as well as small FAR. As we mentioned ahead, all IoT packets changed between IoT bias or routed to upper layers(fog subcaste and pall subcaste) go through smart IoT gateway. A TCP dump is a network sniffer and packet tool used to capture packets that entered or transferred over the network in the edge subcaste. The TCP dump tool run on the smart IoT gateway to collect raw network business(packets) from IoT network in P cap train format. A network business inflow tool called" CICFLOWMETER" used to convert the raw network business to CSV format with further than 80 network business features(30). CICFLOWMETER offers further inflexibility in terms of choosing the features you want to calculate. This stage is composed of several way, the first of which involves training the LSTM model with the available IoT datasets in the CSV train format that collected from the previous step. The training of LSTM model is carried out in the pall subcaste to make it goes briskly. We use Amazon Elastic cipher Cloud (Amazon EC2) virtual garçon case on which we can run Keras for literacy and testing LSTM model. Amazon erected the DL Amazon Machine Image(AMI) AmazonLinux-2.0 for DL on EC2 cases with popular DL frame and also contains the anaconda platform. Amazon Web Services DL AMI are erected to make, train and remedy DL models in EC2 with popular fabrics similar as Keras, TensorFlow, Py Torch and further. We can copy the CSV train collected at the edge subcaste by smart IoT gateways to the AWS case using scp command by using the keypair and the Ip address of the AWS EC2 case as follows scp- i keras aws-keypair. pem- r srcec2-user@54.180.78.7̃ / also, we run the LSTM model for model training and testing. We use the sigmoid activation function in double class bracket and soft max in multi-class classification. The LSTM network used in the proposed frame has one input subcaste with 128 cells and one affair subcaste with(m) cells grounded on the number of classes and has three retired layers with(256) cells each. We use an adaptive literacy rate system called" Adam" as an optimizer that computes individual literacy rates for different parameters and achieves good results presto. It combines the advantages of Adaptive grade Algorithm(Ada Grad) and Root Mean Square Propagation(RMS Prop). also, the hyperparameters, which affect the performance of the DL model, are tuned. Hyperparameters values similar as time number, learning rate, and patch size are set before launch model training. We use numerous values for the DL model's hyperparameters to achieve the stylish performance. After training, the performance of the trained LSTM model is estimated on unseen data grounded on evaluation criteria bandied in section VI. therefore, we use different training and testing datasets using train and test split system. We resolve the datasets into two corridor, vide licet, training and testing corridor. The size of the training dataset is 70 of the entire dataset size, and the remaining 30 is used for testing. However, the hyperparameters are tuned, or the LSTM model becomes deeper until the stylish performance is achieved, If the evaluation result isn't good. Our trials conclude that, LSTM has the loftiest performance that outperform all other DL model in terms of DR, FAR, delicacy, recall, F1-Measure and perfection. 2) Framework Deployment The alternate stage of the proposed frame involves enforcing the frame in fog bumps. Fig. 3, shows the armature of the proposed frame, which comprises the pall, fog, and edge layers. The edge subcaste contains billions of IoT bias with limited coffers, similar as detectors, selectors and security cameras. These bias induce huge volumes of unshaped data that are delicate to dissect it at the edge subcaste. The edge subcaste includes smart homes, oil painting equipages, smart power grids, smart buses , and aeroplanes. All data generated from edge subcaste bias are routed to the fog subcaste, enterprise data centre and pall via a smart IoT gateway. The alternate subcaste is the fog subcaste that contains thousands of waiters, routers, and regulators possessed by an Internet service provider. These bias are more important than edge bias. Fog bias can run processes that bear high memory, computational power, storehouse, and energy. likewise, fog bumps are distributed at different geographical areas that live at service provider(SP) networks and near to the edge subcaste than to the pall subcaste. It has multiple interfaces and services to communicate with different protocols and operations. Distributed data analytics at the fog subcaste allows data processing before transmitting them into the pall. also, it minimizes bandwidth, reduces quiescence, and fleetly response to critical conduct that make the system flexible. The top subcaste is the pall subcaste that delivers computing services over the internet and provides flexible, dependable, and scalable coffers to cloud druggies. pall computing allows data transfer, storehouse, and analysis through the internet. IoT gests high quiescence during data transfer or data analysis in the pall, especially on real- time operations, similar as smart vehicles. To apply the proposed frame in IoT networks, we assume that a clustering algorithm similar as in(31) is used to group fog bumps into clusters as shown inFig. 3, the fog subcaste is divided into N clusters. This clustering algorithm designed for the distributed discovery of clusters of wireless bumps grounded on physical network topology features. This clustering algorithm work without need any information about the anticipated number of clusters and it assumes a zero or low mobility for sharing bumps that make it applicable for our frame. It identifies clusters grounded on some parameters similar as the viscosity of the network graph, the preferential attachment, and the relations among nodes. Clustering fog bumps applied to balance network cargo, increase network scalability and secure the changed business between clusters and pall. One cluster can handle, process and dissect data from different IoT edge networks; for case, cluster1 in Fig. 3 data from a wireless detector network and smart home network. Every cluster has cluster members, one Cluster Head(CH) and one backup cluster head(BCH) that tagged by cluster members. CHs are responsible for process incoming and gregarious business that are generated to or from their cluster members. also, CHs are the links between the pall service and clusters members. All the attack discovery updates from pall service propagated to cluster members via CHs as shown inFig. 3 by black arrows come from the pall subcaste to CH and from CH to cluster members. All data changed between CHs and pall service translated using Triple Data Encryption Standard(3DES). We assume that the paths between smart IoT gateway bias and fog bumps are secured which cipher data before transmission. In case of CH failure, BCH becomes the CH and a cluster member becomes BCH. Smart gateways work as Gomorrah bumps, they collect the business changed in the network and further it to the nearest fog knot. Fog bumps admit network business encouraged from several IoT smart gateways and store it in different lines. A service in the back group on fog bumps work to read and reuse data from the lines. The data processing is rooting the features of each packet in the network business, also feed them to the LSTM classifier to descry attacks. 3) Business Analysis and Attack Discovery After enforcing the attack discovery on the fog bumps in all clusters, it starts to admit network business routed from IoT smart gateway bias from different networks. The proposed attack discovery handles raw network data, classifies business into two classes(normal or attack), and recognizes the type of attack on the base of the dataset used in the training stage. Every cluster member saves the ID of its head knot, and the CH knows the number of cluster members that form its cluster and their IDs. In the case of attack discovery at any fog cluster member, by using the ID of the head knot, complete information about the detected attack(attack type, attack source, protocol, duration, etc.) is propagated from cluster members to the CH that work as information aggregator, also it's propagated from the CH to the pall service as shown in figure 3 by red arrows. The information goes from cluster members to CHs and also to the pall service. The pall service raises an alarm and logs all the information from CHs to be used by a network director in assessing and streamlining the performance of the attack discovery and taking the applicable opinions.

4) Performance Monitoring and streamlining The DL model capability to descry attacks lowered overtime. So, the performance of the attack discovery is tested

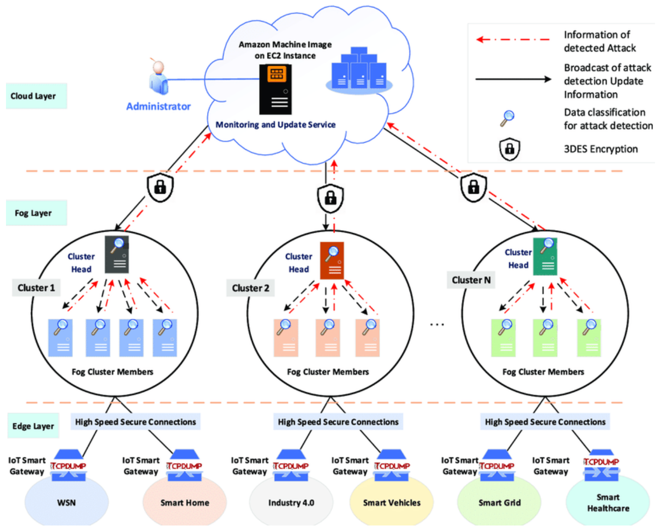
Algorithm 1 Performance Monitoring and Updating

```
1: Input: Detection Rate (DR) and False Alarm Rate (FAR) values.
2: Output: Replace or keep current DL model.
3: Collect new data from edge layer devices as discussed in section IV-1.
4: Test the performance of current attack detection every month.
5: Evaluate the current detection model with samples of new data.
6: if (DR_Test < DR_Train) or (FAR_Test > FAR_Train) then
7: Performance degraded
8: Train current model with new data or combine new and old data.
9: Hyper-parameters tuning.
10: Evaluate the performance of current model with new data.
11: Deploy updated and current model and monitor them in parallel.
12: if Updated model outperform current model then
13: Replace current model with updated model
14: else
15: Keep current model
16: end if
17: else
18: if Performance Improved then
19: Investigate changes in new data
20: else
21: Go to step 8
22: end if
23: else
24: Keep monitoring current attack detection performance.
25: end if
```

and streamlined to corroborate the capability of the discovery machine or the DL model. We measured the capability of the DL model on the base of two parameters, vide licet, the DR and FAR. At different times in the month, new data is collected from the edge subcaste bias and uploaded to the Amazon EC2 garçon after converting it to CSV train format using CICFLOWME TER tool as mentioned in section IV- 1. The collected data may have a huge volume of data, we can elect samples of the collected data to test the performance of the running DL model. The evaluation criteria similar as DR, and FAR are calculated in the testing process using samples of the recently collected data. We compare the evaluation criteria values i the testing process with values in training phase(phase1). The evaluation criteria show whether the performance of the discovery model degrades, improves or become stable. However, or the FAR in the testing process(FAR Test) is lesser than FAR value in the training process(FAR Train), the performance of the model degraded, If the DR value in the testing process(DR Test) is lower than the DR value in the training stage(DR Train). still, the system is stable, If DR and FAR values not changed. For demeaning or stable performance, we train the model using new data and tune hyperactive- parameters to increase the model discovery capability. Two styles are used for the new data during model update. The model is trained by using only new data or by combining samples of the new with old training data. Eventually, the streamlined model is estimated, and the current model is replaced with the streamlined model. We cover the streamlined and current model at applicable time intervals. Doing so enables us to fully modernize the attack discovery on all fog bumps or maintain the function of the being model. Algorithm 1 shows the way in monitoring and streamlining the performance of the proposed attack discovery.

V. OVERVIEW OF DL MODELS AND DATASETS USED IN THE EXPERIMENTS

DL is a subset of ML, and DL is biologically inspired by the mortal brain and neurons(18). DL consists of supervised literacy(discriminational literacy), unsupervised literacy(generative literacy), and mongrel literacy. In this section, we introduce an overview of different discriminational literacy models used in the trials. Artificial neural networks(ANNs) are divided into three classes, videlicet, MLP, CNN and RNN. These classes have flexible infrastructures and have proven their success in colorful problems. We use LSTM, bidirectional LSTM(BiLSTM), and reopened intermittent unit(GRU) as the representative of RNNs. We elect DNN as the representative of MLPs. CNN- LSTM is named to represent cold-blooded network models.


Information of detected Attack
Broadcast of attack detection Update Information
Data classification for attack detection
3DES Encryption
Cloud Layer
Administrator
Amazon Machine Image on EC2 Instance
Monitoring and Update Service
Fog Layer
Cluster 1
Cluster Head
Fog Cluster Members
Cluster 2
Cluster Head
Fog Cluster Members
Cluster N
Cluster Head
Fog Cluster Members
Edge Layer
High Speed Secure Connections
High Speed Secure Connections
High Speed Secure Connections
IoT Smart Gateway
TCPDUMP
WSN
Smart Home
Industry 4.0
Smart Vehicles
Smart Grid
Smart Healthcare

1) RNN

RNN is a supervised DL model that's designed to handle the successional data of some operations, similar as speech recognition, machine restatement, music generation, and sentiment analysis. RNN used to descry botnet gest in(19) and proposed to make an intelligent network attack discovery system in(32) which outperformed SVM. RNN can be considered a group of cells in which each cell performs the same operation on every element in the sequence. In the armature of unrolled RNN, each cell has input X and affair Y and three weight matrices, vide licet, U, W, and V for the inputs, retired countries, and labours independently. U, W and V matrices have the same values in all time way for different inputs. therefore, the total number of variables is reduced and RNN performs faster than other DL models. We denote the input sequence as X = (x0, x1, x2,., xn), the retired vector as H = (h0, h1, h2,., hn), and the affair sequence as Y = (y0, y1, y2,., yn). The affair sequence values are calculated as follows ht = σ(Uxt W ht −1 bh)---(1) yt = V ht by —(2) where σ is a nonlinearity activation function, xt is the input value at time t, ht −1 is the retired state of(t- 1), and bh and by are the bias values. RNN uses backpropagation through time to calculate the weights of RNN to minimize the error in the network affair compared to anticipated affair. RNN suffer from downsides; for illustration, it can not study long sequences and it's prone to evaporating and exploding grade problems(33). likewise, it only uses the information that's earlier in the sequence to make a vaticination but not use information latterly in the sequence. therefore, RNN variants similar as GRU and LSTM, have been proposed to overcome these problems. 2) LSTM LSTM is a variant of RNN that designed to deal with evaporating and exploding grade problems. LSTM was introduced by Hoch reiter and Schmid in 1997(34). LSTM can learn from long dependences . The LSTM cell has three gates, vide licet, forget, input, and affair gates that control and cover cell countries. The Input gate, forget gate, affair gate and state of the memory cell are calculated as follows where it, ft, ot, ct are input, forget, affair gates, and cell state in depend

$$i_t = \sigma(W_i h_{t-1} + W_i x_t + b_i) \qquad (3)$$

$$f_t = \sigma(W_f h_{t-1} + W_f x_t + b_f) \qquad (4)$$

$$o_t = \sigma(W_o X_t + W_o h_{t-1} + b_o) \qquad (5)$$

$$h_t = o_t \tanh(c_t) \qquad (6)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_c x_t + W_c h_{t-1}) \qquad (7)$$

where it, ft, ot, ct are input, forget, output gates, and cell state respectively. σ is the sigmoid function, b is the bias, xt is the value of input subcaste at time t, ht is the retired state of the cell at time t and W is the weight values. Stochastic grade descent(SGD) is an optimization system that generally used in DL to get the minimal loss function that used to modernize the weights of the neural network through backpropagation using literacy rate α. SGD updates the current weight w using grade ∂(L)/ ∂(W) multiplied by α.

$$w_{t+1} = w_t - \alpha \frac{\partial(L)}{\partial(w_t)} \qquad (8)$$

Authors in(9) used LSTM to descry and classify authorization- grounded android malware which achieved the loftiest delicacy using real- world Android malware test dataset. In our proposed frame, we use LSTM in attack discovery. The performance of the LSTM model bandied in details I section VI. 3) Bi LSTM Bidirectional LSTM proposed in(35) to prize spatial features and bidirectional temporal dependences from literal data. Bi LSTM is developed for speech recognition, handwrit ten recognition, and protein structure vaticination. It obtains the stylish benefits from an input sequence on the base of former and unborn sequences. It duplicates the first intermittent subcaste in the network and places them together. The input sequence fed to the first subcaste as it's and a rear dupe of the input is fed to the alternate subcaste. The first and alternate intermittent layers are connected to the same affair subcaste.

4) GRU

In 2014, Cho presented GRU grounded on the LSTM network model with many parameters(36). GRU used in polyphonic music modelling , speech signal modelling , and handwriting recognition. GRU has a simpler structure and smaller cell factors than LSTM. It resists the evaporating grade problem and trains briskly because of its small number of calculations. GRU has two gates, vide licet , update(z) and reset gates(r). The update gate and reset gate are calculate as follows :

$$z_t = \sigma(W_z h_{t-1} + W_z x_t + bz) \quad (9)$$

$$r_t = \sigma(W_r h_{t-1} + W_r x_t + br) \quad (10)$$

$$c_t = \tanh(W_h x_t + W_h(h_{t-1} \odot r_t) + bh) \quad (11)$$

$$h_t = (z \otimes c) \oplus ((1 - z) \otimes h_{t-1}) \quad (12)$$

where zt, rt, ct and ht are update, reset, cell state and hidden state of the cell. xt represents the input at time t, W represents the weight values, and b represents the bias values. GRU outperforms LSTM in some cases, especially on small datasets [37]. However, LSTM or GRU can't be used for specific tasks. GRU is faster than LSTM in training and can generalize using few data. LSTM requires voluminous data during training and consumes more time than GRU, but it provides better performance in large-scale tasks. GRU used to detect attacks in automated process control system in [38].

5) CNN

CNNs have shown remarkable performance in object recognition in images(39). Convolutional, pooling and completely connected layers are piled with each other to produce the CNN armature. The first subcaste in CNN is the convolutional subcaste, which uses multiple equal size pollutants to convolute input dataparameters.However, I, and a two- dimensional smoothing kernel, If we've a two-dimensional image would be calculated as follows:

$$S(i,j) = \sum\sum I(m,n)K(i-m, J-n) \quad (13)$$

The pooling subcaste conducts down slice for the representation of spatial confines(range, height) through maximum pooling or average pooling operations and reduce the number of parameters and thus, overfitting. CNN uses" powerhouse" to reduce overfitting. CNN achieves high performance in learning from raw data but bear a high volume of training data. CNNs bear high computational coffers during network training because they perform huge calculations. therefore, the perpetration of CNNs on bias with limited coffers(detectors, smart bias) is grueling . As mentioned in the affiliated work section, CNN used in(22) to descry malware by converting the opcode to grayscale images.

6) CNN-LSTM

CNN-LSTM is a is a mongrel DL model designed for visual time series prognostications and textual generation from images sequences, similar as exertion recognition and videotape description. The CNN- LSTM armature combines CNN layers for point birth from inputs and LSTM layers for time sequence vaticination. CNN- LSTM has achieved advancements in speech recognition on DNN. It used in visual recognition and description by(40).

7) Datasets Descriptions

Most experimenters have used the KDDCUP99 dataset for training and assessing their proposed models. We use five datasets to train and estimate six supervised DL models in double and multi-class groups. We use two new IoT datasets(RPL- NIDS- 2017 and N-Ba IoT- 2018) to descry IoT attacks and other datasets(UNSW- NB- 2015, CICIDS- 2017 and NSL- KDD) to descry conventional attacks. The datasets have been chosen grounded on the novelty and diversity of attack they have. We elect them to prove that the proposed frame suitable to descry conventional attacks like U2R, R2L, FTP, Worms, Port over look, etc. besides the IoT attacks. likewise, we need to prove that the relinquishment of DL in attack discovery in IoT is a successful idea and important tool. The first data set is UNSW- NB15 for IDSs(41). This dataset created at the Cyber Range Lab of the Australian Cyber Security Centre. It has nine attack orders, vide licet, fuzzers , analysis, backdoor, DoS, exploits, general, canvass naissance, shellcode and worms. The dataset is expressed in CSV format and contains a aggregate of 47 features(state, duration, protocol, service ,etc.). We use the dataset train named(UNSW-NB15-1.csv) because DL models bear a large volume of data for training. This dataset has 700,000 records, and we resolve the dataset into 490,000 samples for training and 210,000 samples for confirmation. The dataset has the following distribution 677,763 normal business, 7,522 general packets, 5,409 exploits, 5,050 fuzzers, 1,759 surveillance, 1,167 DoS, 533 backdoor, 526 analysis, 223 shellcode and 48 worms. All categorical features are decoded into separate features, and the entire data set is regularized using scikit learn by rescaling each row to have a length of 1. In our trial, we train the model for double and multi-class groups. For double bracket, we use the dataset in two classes(normal and attack). For multi-class bracket, we use the dataset in ten classes(normal, fuzzers, analysis, backdoor, DOS, exploits, general, surveillance, shellcode, and worms). The alternate dataset is CICIDS- 2017 for ID evaluation(42). This dataset was created at the Canadian Institute for Cybersecurity(CIC), University of New Brunswick, Canada. The dataset contains benign and over- to- date common attacks similar as Web- grounded, brute force, DoS, DDoS, infiltration, heart- bleed, bot and check up attacks. CIC Flow Meter is used for network business analysis to induce the CSV lines from PCAP business lines. The CIC Flow Meter is a software that available to public on the website of CIC. The CSV lines contain 80 network business features and markers. The dataset is erected on the base of abstract gest of 25 druggies with different proto couloirs, similar as the HTTP, FTP, SSH, and dispatch protocols. To train the DL models, we combine four CSV lines containing normal business, DDoS, port scan, web, SSH, and FTP attacks. The combined dataset has 745,423 records(packets), and we resolve the dataset into 521,796 samples for training and 223,627 samples for confirmation. The dataset has the following distribution 459,199 normal, 123,534 DDoS, 147,329 ports can, 7,935 FTP- Pa ta tor, 5,897 SSH- Pa ta tor, 1,507 web attack-brute force, and 22 web attack- SQL-injection. In the trial, we train the model for double and multi-class groups. For double bracket, we use the dataset in two classes(normal vs attack). For multi-class bracket, we use the dataset in seven classes(normal, DDoS, FTP pa ta tor, SSH-pa ta tor, web attack-brute force, exploits and web attack-SQL- injection). The third dataset is RPL- NIDS17 for IDS in RPL- grounded 6LoWPAN networks(43). This dataset is developed by collecting traces after bluffing different routing attacks against RPL routing protocol. The dataset has 21 features as control_packet_type, source_id,destination_id,app_layer_arrival_time and so on, as well as 1 marker. The dataset is created with and without point encoding, and we use the dataset with point garbling for training and testing. For double bracket, the training dataset has 116,679 records of normal business and 33,337 records of attacks. The testing dataset has 59,560 records of normal business and 16,971 records of attacks. For multi-class bracket, the dataset has seven classes of attacks(clone- ID, hello flood tide, original form, picky forwarding, billabong, billabong and blackhole, and sybil) and normal records. The training dataset is distributed as follows 116,679 normal, 4,405 clone ID attack, 4,822 welcome flooding, 4,822 original form, 4,822 picky forwarding, 4,822 billabong and blackhole, and 4,822 sybil. The testing dataset is distributed as follows 59,560 normal, 2,225 clone ID attack, 2,408 welcome flooding, 2,450 original form, 2,424 picky forwarding 2,495 billabong,2,485 billabong and blackhole, and 2,484 of sybil. The fourth dataset is then Ba IoT dataset developed for detecting IoT botnet attacks using anomaly discovery ways(44). The dataset contains real business collected from nine marketable IoT bias including Dan mini doorbell, Eco bee thermostat, and Provision PT- 737E security camera. The authors compromised the IoT bias used in their testbed using and BASHLITE botnet attacks. The dataset contains five attacks of BASHLITE(check up, junk, UDP, TCP, and Quintet) and five attacks of (check up, Ack, UDP, UDP plain). To prove that the proposed frame can efficiently descry multiple attacks in several datasets especially in IoT datasets, we estimate our proposed approach with five attacks on a Dan mini doorbell. The dataset is expressed in CSV format and has 115 numerical features. The dataset is distributed as follows 49,548 normal business, 102,194 Ack, 107,685 check up, 122,573,237,665 UDP, and 81,982 UDP plain attacks. Then Ba IoT dataset is used for double and multi-class bracket. For double bracket, we use the dataset in two classes(normal vs attack) with 49,548 records for normal business and 652,099 for attacks. For multi-class bracket, we use the dataset in six classes(normal, check up, Ack, UDP, UDP plain). The fifth dataset is the NSL- KDD dataset, which is a refined interpretation of the KDD Mug 99 dataset(45). The NSL KDD dataset is extensively used in exploration to estimate different IDSs. Although, the NSL- KDD dataset has essential disadvantages, we use it to compare our proposed attack discovery frame with state- of- the- art IDSs. The NSL- KDD dataset is available in multiple formats, similar as TXT and ARFF formats for training and testing lines. We used the CSV train format for training and testing datasets. Each record in the dataset has 41 attributes, including duration, proto type, service, flag, bytes, and dst bytes. The dataset has five orders, vide licet, DoS, stoner to root(U2R), remote to original(R2L), inquiry attacks, and normal order. The 42nd column in the dataset contains the data about the five classes distributed as normal or one of the classes of four attacks. The dataset has 125,973 records for training and 22,544 records for testing. The training dataset is distributed as follows 67,343 normal business, 45,927 DoS, 11,656 inquiry, 995 R2l and 52 U2R. The testing dataset is distributed as follows 9,711 normal, 7,458 DoS, 2,754 inquiry, 2,421 R2L, and 200 U2R. In our trial, we train the model for double and multi-class groups. For double bracket, we use the dataset in two classes(normal and attack). For multi-class bracket, we use the dataset in five classes(normal, DoS, inquiry, U2R and R2L). VI. trials AND RESULT EVALUATION All DL models used in our trials are enforced using Keras on TensorFlow. Keras is a high- position API for structure and training DL models. All the trials are conducted on a particular computer with Intel Core i5- 7400 CPU@3.00 GHz, 8 GB memory, and CPU- enabled TensorFlow on 64-bit Windows 10. We apply several types of supervised DL models similar as GRU, LSTM, CNN, CNN LSTM, and DNN. All these models can prize deep features from the raw data fed to them. The features uprooted by the DL models are compared with the test features in the discovery phase. To prove the effectiveness of the proposed fog- grounded attack discovery frame in terms of response time, we calculate the response time of the proposed attack discovery in fog-grounded and pall- grounded. We calculate DR and FAR as evaluation criteria for assessing the DL models. likewise, perfection, recall, F1- Measure and discovery time have been used for performance evaluation and comparison. DR refers to the proportion of the total number of correct groups. FAR refers to the proportion of normal events inaptly classified as vicious. delicacy is the chance of rightly classified samples over the total number of samples. Precision is the rate of rightly prognosticated positive compliances to the total prognosticated positive compliances. Recall calculates the number of positive samples classified as cons. F1- score is the weighted normal of perfection and recall. Discovery time is the time for classifying one packet as normal or an attack. The DL model can be trained offline at a pall garçon using GPU to accelerate the training, although the training time isn't important. In our trial, we calculate the average discovery time by running the discovery 100 times and calculating the average time. The following equations shows the fine representation of the evaluation criteria.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN},$$
$$FAR = \frac{FP}{FP+TN}, DR = \frac{TP}{TP+FN}$$

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN},$$
$$F1 - Measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

To get the best- trained models in double bracket, we use0.1,0.01 and0.001 for learning rate, 32, 64 and 128 for batch size, 100 for time number, and Adam optimization algorithm for all DL models. We got the stylish results using0.01 for learning rate, 64 for batch size, and Adam as an optimization algorithm. DNN has an input subcaste with 1024 cells, and five retired layers with 512 cells each using Re LU activation function, and one affair subcaste with one cell using sigmoid activation function. LSTM has an input subcaste with 128 cells, and three retired layers with 256 cells each, and one affair subcaste with one cell using sigmoid activation function. Bi-LSTM has an input subcaste with 128 cells, and three retired layers with 128 cells each, and one affair subcaste with one cell using sigmoid activation function. GRU has an input subcaste with 64 cells, and three retired layers with 64 cells each, and one affair subcaste with one cell using sigmoid activation function. CNN has three convolutional layers with 64 pollutants each using Re LU activation function, three pooling layers, and one affair subcaste with one cell using sigmoid activation function. CNN- LSTM has three convolutional layers with 64 pollutants each using Re LU activation function, three pooling layers, one LSTM subcaste with 256 cells, and one affair subcaste with one cell using sigmoid activation function. In multiclass bracket, the DL models have the same configuration in double bracket but the affair subcaste has number of cells equal to the number of classes, and soft max activation function used rather of the sigmoid. We use powerhouse to help overfitting. We train and estimate every DL model for double and multi-class groups with five datasets. Each dataset comprises different attacks with different patterns and is used to estimate the capability of DL models of detecting different patterns from raw data. The DL model with the stylish performance across all datasets is also linked. Table 2 shows the evaluation metric values for all DL models and their performance in double bracket. For the UNSW- NB15 dataset, LSTM achieves the loftiest delicacy(99.96), DR(99.97), perfection(99.98), recall(99.97), and F1- Measure(99.98). The performance of Bi-LSTM is close to that of LSTM, whereas the CNN- LSTM achieves the smallest FAR and delicacy. The Bi-LSTM outperforms the GRU, CNN, and CNN- LSTM models. For the CICIDS 2017 dataset, the LSTM model achieves the loftiest delicacy(99.37), perfection(99.28), F1- Measure(99.49), and FAR(1.15), among all the models, and the DNN model achieves the smallest performance. The Bi-LSTM model ranks second after the LSTM model and outperforms DNN, GRU, CNN, and CNN- LSTM. For the RPLNIDS- 2017 dataset, LSTM is superior to all other DL models in terms of ac curacy(98.15), DR(99.07), recall(99.07), and F1- Measure(99.12), whereas the CNN model achieves the smallest FAR(11.38) and loftiest perfection(99.2). CNN LSTM achieves the smallest performance and least delicacy, perfection, and F1- Measure Ba IoT evaluation results show that LSTM is the stylish in terms of(99.81), and smallest FAR(0.1), and loftiest delicacy(99.85). Whilst, DNN provides the smallest performance. LSTM achieves discovery delicacy(99.34), and FAR(0.1) in the NSL- KDD dataset. LSTM is superior and outperforms all other DL models used in the trials for double groups using the five datasets. GRU consumes lower discovery time than LSTM because it has smaller parameters and calculations but LSTM outperform GRU in DR, FAR and discovery de licacy. Fig. 4 and 5 show the performance of LSTM for the UNSWNB15 dataset in the training and confirmation phases in terms of delicacy and loss values. For double bracket, Fig. 4(a) shows the increase in the training and confirmation delicacy with the increase of ages to reach the stylish delicacy after 100 ages with a 64 batch size. Fig. 4(b) shows the drop in training and confirmation losses that gathered after 100 ages with a 64 batch size. For multi-class bracket, Fig. 5(a, b) show the delicacy and loss performance of LSTM with UNSW- NB15. Its delicacy reaches its outside (98.82) at 50 ages, and its loss values to reach its minimum (0.0288) at 50 ages. For multi-class bracket, every dataset has different figures and types of attacks. For illustration, the CICIDS- 2017 dataset has seven business orders with one normal order and six orders of attacks, and UNSW- NB15 has ten business orders one normal and nine orders of attacks. An imbalance of classes that's reflected on the performance of different DL models. Considering space limitation, we present the performance criteria of one dataset (CICIDS 2017) in detail for multi-class bracket. Table 3 shows the values of perfection, recall, and F1- Measure for every DL model used in the trials. LSTM is superior to the other DL models in terms of overall all classes.

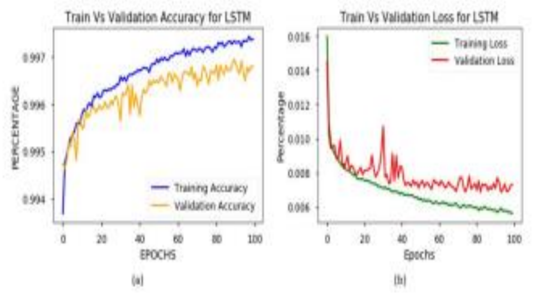


FIGURE 4: LSTM Performance with UNSW-NB15 Dataset in binary classification a) accuracy performance b) loss performance

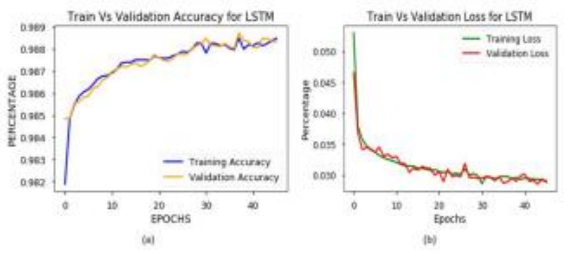


FIGURE 5: LSTM Performance with UNSW-NB15 Dataset in multi-class classification a) a) accuracy performance b) loss performance

Fig.6 shows the accuracies of the DL models with five datasets in multi-class classification and it illustrates that LSTM is superior to all other DL models. The proposed ML algorithms are deep point embedding literacy with grade boosting tree, KNN, DT, LG, NB and SVM. Fig. 7 shows that LSTM is easily better than all ML algorithms in terms of delicacy, perfection, and recall. Fig. 8 shows another comparison between LSTM and the KNN, RF, ID3 and quadratic discriminant analysis (QDA) for the CICIDS- 2017 dataset used in(42). The proposed attack discovery outperforms all ML algorithms in terms of perfection, recall, and F1- score. The proposed attack discovery is analogous to the proposed attack discovery in (20) but with some differences. Table4. illustrates the differences and parallels, of both attack discovery fabrics. The proposed attack discovery is trained at the pall subcaste and enforced in the fog subcaste, whereas the other attack discovery is trained and run in the fog subcaste. We use six DL models, whereas the other attack discovery uses only one DL model. We use five datasets in training.

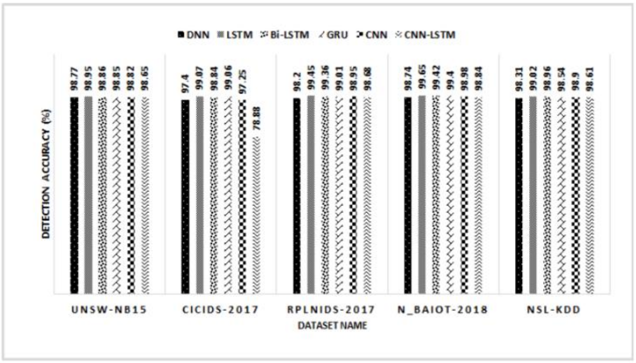


FIGURE 6: Detection accuracy values of six DL models with five datasets

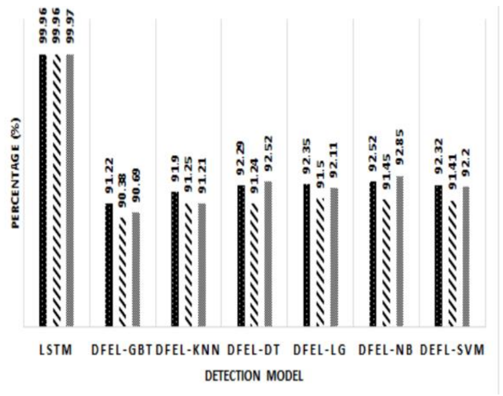


FIGURE 7: Performance comparison between LSTM and modified ML models proposed in [46] on UNSW-NB15 dataset

and testing the DL models, whereas the other model use only one dataset. Our proposed attack discovery achieves high DR in double and multi-class bracket. Our proposed frame is also further scalable than other attack discovery. To compare the effectiveness of the proposed fog- grounded attack discovery system in terms of response time, we apply the proposed attack discovery frame in fog- grounded and pall grounded armature as shown in Fig. 9.

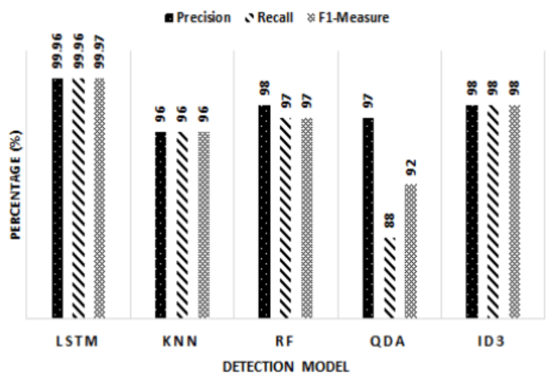


FIGURE 8: Performance comparison between LSTM and ML algorithms used in [42] on CICIDS-2017 dataset.

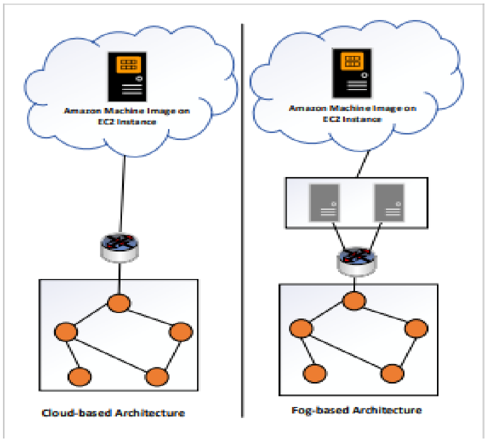

Amazon Machine Image on EC2 Instance
Amazon Machine Image on EC2 Instance
Cloud-based Architecture
Fog-based Architecture

FIGURE 9: Experiment different scenarios

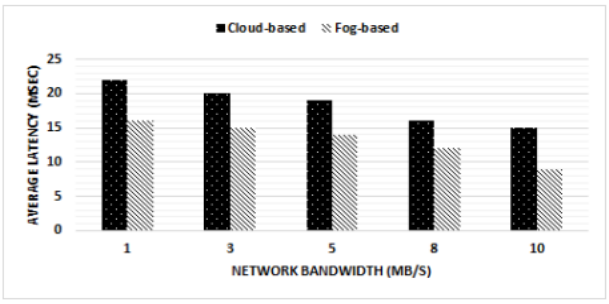


FIGURE 10: Average response time for fog-based and cloud-based detection

We use simulator(47) to pretend the wireless detector network(WSN) molecules to represent the IoT edge subcaste and use CONTIKI- NG(48) platform to apply IoT WSN molecules. The fog subcaste bumps enforced using two computers to represent the fog subcaste bumps, they connected with the simulated edge detectors. In the pall-grounded armature, we apply the attack discovery frame in the pall using Amazon EC2 virtual garçon case. We measure the response time 10 times and calculate the average value for different network pets. As shown in Fig. 10, the response time of the proposed fog- grounded armature is lower than the pall-grounded since the fog bumps is near to the edge subcaste and can descry attack with low quiescence. LSTM can learn from long sequences and ground inputs with long time gaps by using the forget gate to store information about the former state of the network. therefore, LSTM can use literal data from former network business to descry attacks, similar as DoS and DDoS, for a long time period. The data generated from IoT bias are unshaped, and LSTM can learn effectively from unshaped data and excerpt deep perceptivity. also, the performance of LSTM increases with the increase of training data volume because data are the heart and soul of DL. These conditions lead to the superiority of LSTM to the rest of the DL models. The deep structure, point scale, and the huge number of weights and variables calculated by DL models make them better than ML algorithms.

VII. CONCLUSION AND FUTURE WORK

In this paper, we've proposed an attack discovery frame work grounded on LSTM DL model that used for IoT business bracket. In the proposed frame, the edge subcaste business collected and transferred to the pall subcaste to train the LSTM model. also, the trained model installed on the fog subcaste bumps as a discovery machine to descry attacks. Fog computing provides a distributed terrain with numerous fog bumps near to IoT bias in the edge subcaste. therefore, we apply the discovery system on the fog bumps to dissect the data near to the edge subcaste to minimize quiescence. We cover the performance of the LSTM model and update it using a pall service. The trials have shown the success of the DL models to be espoused to Cybersecurity to descry several attacks with high discovery and delicacy rates. It's also demonstrated that the DL models can descry conventional and Cyberattacks was in different datasets. We conclude that

the LSTM model is superior to all other supervised DL models used in the trial because it has forget gate to store the former state information and can learn from long sequences. This proposed frame overcomes the problems of how to apply the heavy DL discovery system directly on limited capacity IoT bias, descry several attacks with high discovery rate and high delicacy rates, and how to cover the discovery system and update it to descry new attacks. still, it has a debit in labelling the data that collected in the edge subcaste to train the LSTM model in the pall may be delicate. In the future, we will compare the proposed attack discovery with unsupervised DL models and underpinning literacy using a distributed computing terrain like Apache Spark with different datasets.

REFERENCE

[1] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1250–1258, Oct 2017.

	Proposed Framework	Proposed attack detection in [5]
Architecture	Distributed	distributed
DL model Training	At cloud layer	At fog layer
Layers used for implementation	Fog and cloud layers	Fog layer
Number of DL models used	Six DL models	One DL model
Number of datasets used	Five datasets	One dataset
Detection Accuracy	99.85 % in binary classification 99.65% in multi-class classification	99.2 % in binary classification 98.27% in multiclass classification
Scalability	Scalable	limited scalability

[2] B. B. ZarpelÃco, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A ˇ survey of intrusion detection in internet of things," Journal of Network and Computer Applications, vol. 84, pp. 25 – 37, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S10848045 17300802

[3] M. E. KarsligÐ ¸Tl, A. G. Yavuz, M. A. GÃijvensan, K. Hanifi, and H. Bank, "Network intrusion detection using machine learning anomaly detection algorithms," in 2017 25th Signal Processing and Communications Applications Conference (SIU), May 2017, pp.

[4] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in 2010 IEEE Symposium on Security and Privacy, May 2010, pp. 305–316.

[5] A. Diro and N. Chilamkurti, "Leveraging lstm networks for attack detection in fog-to-things communications," IEEE Communications Magazine, vol. 56, no. 9, pp. 124–130, Sep. 2018.

[6] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," IEEE Access, vol. 6, pp. 35 365–35 381, 2018.

[7] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," PLOS ONE, vol. 11, no. 6, pp. 1–17, 06 2016. [Online]. Available: https: //doi.org/10.1371/journal.pone.0155781

[8] Y. Chen, Y. Zhang, and S. Maharjan, "Deep learning for secure mobile edge computing," CoRR, vol. abs/1709.08025, 2017. [Online]. Available: http://arxiv.org/abs/1709.08025

[9] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Deep android malware detection and classification," in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Sep. 2017, pp. 1677–1683.

[10] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," IEEE Access, vol. 5, pp. 21 954–21 961, 2017.

[11] N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé, and G. Joon Ahn, "Deep android malware detection," in Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, ser. CODASPY '17. New York, NY, USA: ACM, 2017, pp. 301–308.[Online].Available: http://doi.acm.org/10.1145/3029806.3029823

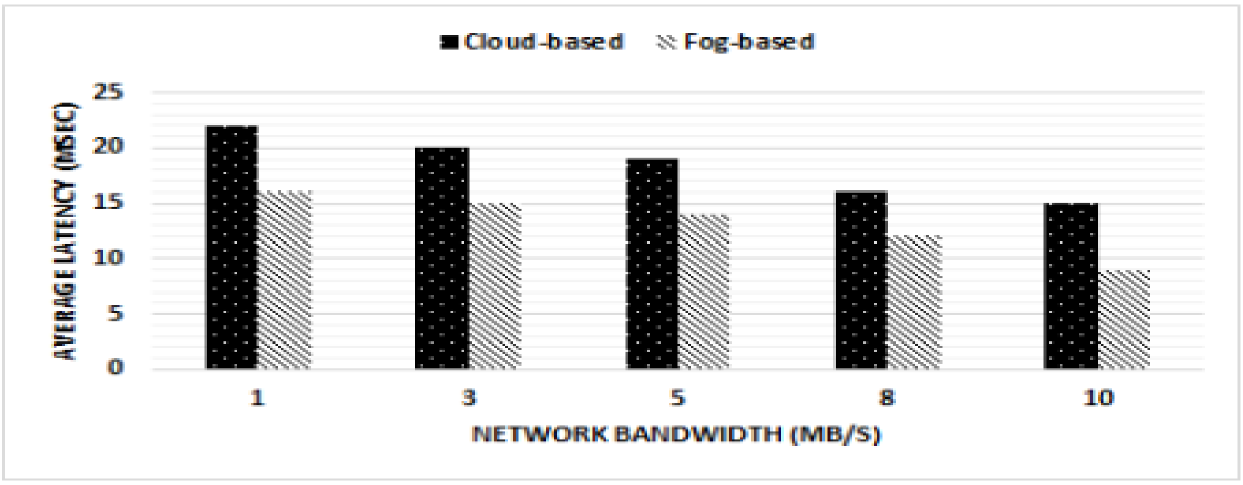
[12] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in 2017 International Joint Conference on Neural Networks (IJCNN), May 2017, pp. 3854–3861.

[13] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1125–1142, Oct 2017.

[14] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the internet of things: Security and privacy issues," IEEE Internet Computing, vol. 21, no. 2, pp. 34–42, Mar 2017.

[15] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," IEEE Internet of Things Journal, pp. 1–1, 2019.

FIGURE 10: Average response time for fog-based and cloud-based detection



We use Cooja simulator [47] to simulate the wireless sensor network (WSN) motes to represent the IoT edge layer and use CONTIKI-NG [48] platform to implement IoT WSN motes. The fog layer nodes implemented using two computers to represent the fog layer nodes, they connected with the simulated edge sensors. In the cloud-based architecture, we implement the attack de tection framework in the cloud using Amazon EC2 virtual server instance. We measure the response time 10 times and calculate the average value for different network speeds. As shown in Fig.10, the response time of the proposed fog-based architecture is less than the cloud-based since the fog nodes is closer to the edge layer and can detect attack with low latency. LSTM can learn from long sequences and bridge inputs with long time gaps by using the forget gate to store in formation about the previous state of the network. Thus, LSTM can use historical data from previous network traffic to detect attacks, such as DoS and DDoS, for a long time period. The data generated from IoT devices are unstructured, and LSTM can learn effectively from unstructured data and extract deep insights. Moreover, the performance of LSTM increases with the increase of training data volume because data are the heart and soul of DL. These conditions lead to the superiority of LSTM to the rest of the DL models. The deep structure, feature hierarchy, and the huge number of weights and variables calculated by DL models make them better than ML algorithms.

VII. CONCLUSION AND FUTURE WORK

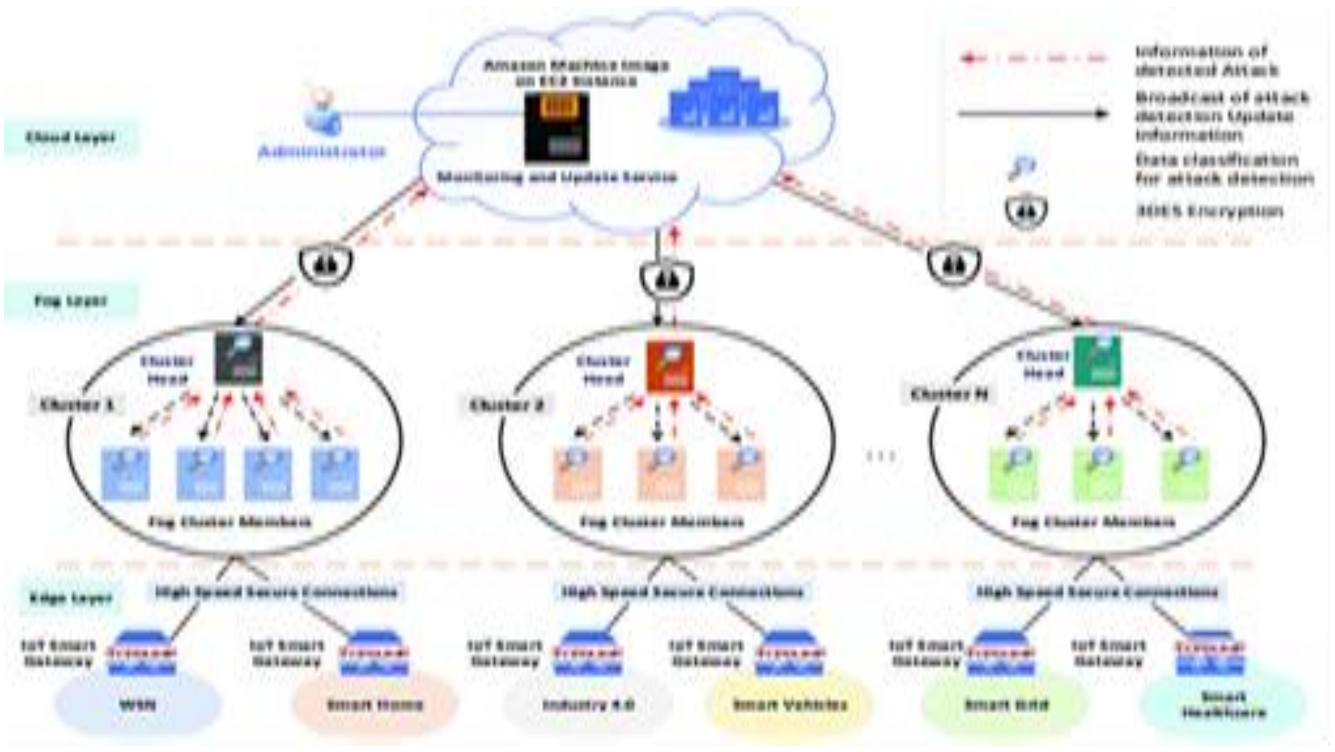
In this paper, we have proposed an attack detection framework based on LSTM DL model that used for IoT traffic classification. In the proposed framework, the edge layer traffic collected and sent to the cloud layer to train the LSTM model. Then, the trained model installed on the fog layer nodes as a detection engine to detect attacks. Fog computing provides a distributed environment with many fog nodes near to IoT devices in the edge layer. Thus, we implement the detection system on the fog nodes to analyze the data close to the edge layer to minimize latency. We monitor the performance of the LSTM model and update it using a cloud service. The experiments have shown the success of the DL models to be adopted to Cybersecurity to detect several attacks with high detection and accuracy rates. It is also demonstrated that the DL models can detect conventional and Cyber-attacks existed in different datasets. We conclude that the LSTM model is superior to all other supervised DL models used in the experiment because it has forget gate to store the previous state information and can learn from long sequences. This proposed framework overcomes the problems of how to implement the heavy DL detection system directly on limited capacity IoT devices, detect several attacks with high detection rate and high accuracy rates, and how to monitor the detection system and update it to detect new attacks. However, it has a drawback in labeling the data that collected in the edge layer to train the LSTM model in the cloud may be difficult. In the future, we will compare the proposed attack detection with unsupervised DL models and reinforcement learning using a distributed computing environment like Apache Spark with different datasets

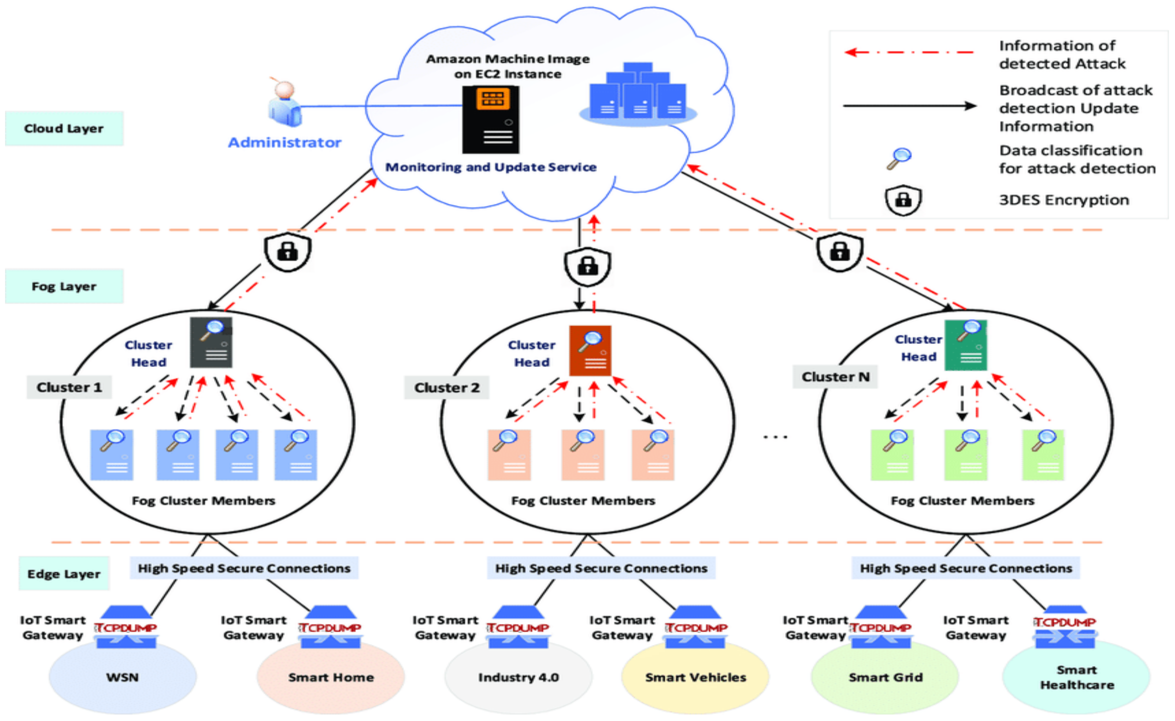
REFERENCE

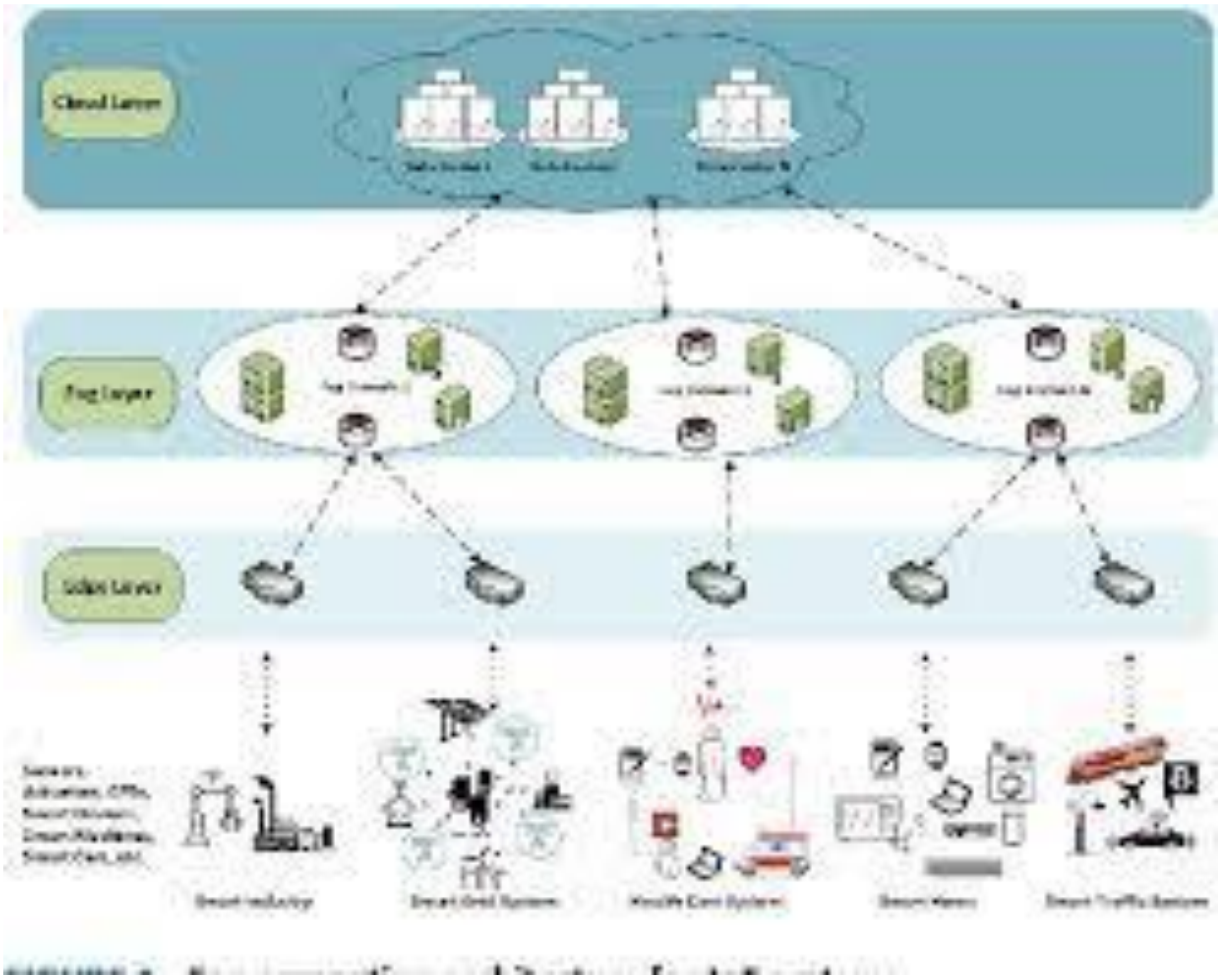
[1] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, “A survey on security and privacy issues in internet-of-things,” IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1250–1258, Oct 2017.

[2] B. B. ZarpelÃco, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, “A ˘ survey of intrusion detection in internet of things,” Journal of Network and Computer Applications, vol. 84, pp. 25 – 37, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517300802

[3] M. E. KarsligĐ ¸Tl, A. G. Yavuz, M. A. GÃijvensan, K. Hanifi, and H. Bank, “Network intrusion detection using machine learning anomaly




Amazon Machine Image on EC2 Instance
Cloud Layer
Administrator
Monitoring and Update Service
Information of detected Attack
Broadcast of attack detection Update Information
Data classification for attack detection
3DES Encryption
Fog Layer
Cluster Head
Cluster 1
Fog Cluster Members
Cluster Head
Cluster 2
Fog Cluster Members
Cluster Head
Cluster N
Fog Cluster Members
...
Edge Layer
High Speed Secure Connections
High Speed Secure Connections
High Speed Secure Connections
IoT Smart Gateway
TCPDUMP
WSN
IoT Smart Gateway
TCPDUMP
Smart Home
IoT Smart Gateway
TCPDUMP
Industry 4.0
IoT Smart Gateway
TCPDUMP
Smart Vehicles
IoT Smart Gateway
TCPDUMP
Smart Grid
IoT Smart Gateway
TCPDUMP
Smart Healthcare



Figures and Table

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE TYPE STYLES

Table Head	Table Column Head		
	Table column subhead	*Subhead*	*Subhead*
copy	More table copy[a]		

Sample of a Table footnote. (*Table footnote*)

Ref	DL model Name	Attack Detected	Dataset Used	Highest Accuracy	Implementation Layer	Strategy
[7]	DBN	CAN network attacks	Packet generated using OCTANE	98%	edge layer	centralized
[8]	DBN	eavesdropping and jamming attacks	10 datasets	Outperforms other ML models by 6%	mobile edge computing	centralized
[11]	CNN	android malware	three datasets	89.7%	mobile devices	centralized
[19]	RNN	botnet attack	Two datasets	96.8%	desktop computers	centralized
[5]	LSTM	Vulnerabilities of wireless communications attacks	Two datasets	99.91%	fog layer	distributed
[12]	AEs	network malware	Two datasets	83.3%	desktop computers	centralized
[21]	ELM	DoS, botnets	One dataset	99%	cloud and edge layers	distributed
[22]	CNN and LSTM	network malware	One dataset	99.83%	desktop with GPU	centralized

Fig. 1. Example of a figure caption. (*figure caption*)

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes