Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 6967938, 25 pages
https://doi.org/10.1155/2022/6967938

Research Article

A Deep Intelligent Attack Detection Framework for Fog-Based IoT Systems

Surya Pavan Kumar Gudla,[1] **Sourav Kumar Bhoi**,[2] **Soumya Ranjan Nayak**,[3] **Krishna Kant Singh**,[4] **Amit Verma**,[5] **and Ivan Izonin**[6]

[1]*Department of Computer Science and Engineering, NCR-PMEC Berhampur, Faculty of Engineering, BPUT, Rourkela 769015, Odisha, India*
[2]*Department of Computer Science and Engineering, Parala Maharaja Engineering College (Govt), Berhampur 761003, Odisha, India*
[3]*School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar 751024, Odisha, India*
[4]*Department of CSE, ASET, Amity University, Noida 201301, India*
[5]*Department of Computer Science & Engineering and University Center for Research and Development, Chandigardh University, Mohali 140413, Punjab, India*
[6]*Department of Artificial Intelligence, Lviv Polytechnic National University, Lviv 79013, Ukraine*

Correspondence should be addressed to Ivan Izonin; ivanizonin@gmail.com

Received 29 August 2022; Revised 12 November 2022; Accepted 22 November 2022; Published 22 December 2022

Academic Editor: Tariq Ahamad

Fog computing provides a multitude of end-based IoT system services. End IoT devices exchange information with fog nodes and the cloud to handle client undertakings. During the process of data collection between the layer of fog and the cloud, there are more chances of crucial attacks or assaults like DDoS and many more security attacks being compromised by IoT end devices. These network (NW) threats must be spotted early. Deep learning (DL) assumes an unmistakable part in foreseeing the end client behavior by extricating highlights and grouping the foe in the network. Yet, because of IoT devices' compelled nature in calculation and storage spaces, DL cannot be managed on those. Here, a framework for fog-based attack detection is proffered, and different attacks are prognosticated utilizing long short-term memory (LSTM). The end IoT gadget behaviour can be prognosticated by installing a trained LSTMDL model at the fog node computation module. The simulations are performed using Python by comparing LSTMDL model with deep neural multilayer perceptron (DNMLP), bidirectional LSTM (Bi-LSTM), gated recurrent units (GRU), hybrid ensemble model (HEM), and hybrid deep learning model (CNN + LSTM) comprising convolutional neural network (CNN) and LSTM on DDoS-SDN (Mendeley Dataset), NSLKDD, UNSW-NB15, and IoTID20 datasets. To evaluate the performance of the binary classifier, metrics like accuracy, precision, recall, f1-score, and ROC-AUC curves are considered on these datasets. The LSTMDL model shows outperforming nature in binary classification with 99.70%, 99.12%, 94.11%, and 99.88% performance accuracies on experimentation with respective datasets. The network simulation further shows how different DL models present fog layer communication behaviour detection time (CBDT). DNMLP detects communication behaviour (CB) faster than other models, but LSTMDL predicts assaults better.

1. Introduction

IoT gadgets like IoVT, IoMT, smart grids, and smart electrical appliances, inter alia, are exceedingly raising in current technologies, leading to so many attacks on those devices with their prominence in resource sharing. Physical devices like sensors and actuators give on-demand administration over the cloud, but its centralization is hazardous. All with this, on providing services by cloud to IoT faces high challenges in data abeyance, data security, data obtrusion, and data shielding [1–8].

An abstraction layer called Fog is used to offer services close to the network's edge and solve cloud-based IoT challenges. Fog, a distributed decentralized model, has

evolved that lies between the cloud and client devices [2], in providing services with less latency and less bandwidth utilization in the network (NW). Howbeit, the fog nodes have a low level of data privacy and are vulnerable to assaults such as probe, DDoS, man-in-the-middle, port scan attacks, and many others [9]. As a result, the fog layer needs an attack detection system. Noncellular network protocols including LoRa, COAP, LoRaWAN, and MQTT, are required to enable communication between the smart devices. These protocols help end users by having low latency and low bandwidth utilization [5, 10]. Data collected from end devices gets exchanged and set aside for the devices using fog communication protocols for further speedy retrieval of data. During the exchange of data, the fog layer/fog-node is more susceptible to attacks. Hence, we require a security system in the fog layer to define attack detection. This work uses LSTMDL model to prognosticate fog layer attacks.

Likewise with immense expansion in use of web and huge measure of information move, has caused a more noteworthy number of peculiarities. In equal measure, the reason for attacks is additionally expanding reliably. Numerous associations are consistently working on network attack discovery to offer secure types of aid to end users. Because of high utilization of cloud administrations and IoT over fog layer prompts more expanded hazard of information infringement. Here in such manner compelled to give or configure more secured system by DL algorithms which can distinguish the attacks powerfully.

With truly expanding of web, the general public is moving towards present day advancements to foresee, recognize or order and investigation network conduct using ML and DL approaches are broadly utilized. Henceforth attack detection is turning out to be latest pattern and examination scope for cyber threats.

Due to geo-distribution and location awareness fog layer became exacting in its nature. At first to distinguish attacks ML strategies are profoundly utilized yet inadmissible for enormous magnitude of information. To beat the limit of ML, DL is utilized in distinguishing assaults in the fog layer as it has numerous layers in handling with a high detection rate. On detection of an attacker, the fog node sends the behaviour update of the node to the cloud as malicious and nonmalicious and multilabel classification [1, 4, 9, 11–15].

With a premier detection rate, DL has been used to categorize numerous attacks, producing binary classifications of typical and aberrant behaviour as well as multilabel classifications that are sent to the cloud for node behaviour updates [1, 4, 9, 11, 12]. Because of the resource restraint nature of IoT, it is preposterous to expect to execute complex DL calculations. Along these lines, DL is reasonable to carry out on fog node/fog layer with high precision. In light of the enormous amount of data, DL is superior to ML calculations. The LSTMDL model is used in this work to identify a security attack on an IoT application based on fog nodes.

This work's accomplishments are as follows:

(1) For fog-based IoT systems, a deep intelligent attack detection framework is suggested in this paper. The framework uses the LSTMDL model to find NW security breaches.

(2) The LSTMDL model is set up in a fog node's compute module to analyze the behaviour of end IoT devices. To choose the most accurate DL model at the fog layer, potentiality balancing is performed across HEM, Bi-LSTM, GRU, CNN + LSTM, DNMLP, and LSTM.

(3) The experimentation is done by using Anaconda platform by considering SDN [16], NSLKDD [17], UNSW-NB-15 [18], and IoTID20 datasets [19] for different attacks.

(4) From the upshots, it is found that the LSTMDL model showed finer accuracy than the other five models, and it is considered in this framework to prognosticate the attack. LSTMDL model shows outperforming nature in binary classification with 99.70%, 99.12%, 94.11%, and 99.88% performance accuracies on experimentation with respective datasets.

(5) The network simulation is also performed to show the performance of different DL models for presenting the behaviour detection time at the fog layer. From this upshot, it is found that DNMLP shows smaller communication behaviour detection time (CBDT) than other models; however; the LSTMDL model performs better in predicting the attacks well. Along with, the CBDT gets reduced with the rise in the number of fog-nodes.

The following is a discussion of the remaining sections: The acknowledgements are presented in Section 2, the system design with network design and assault design are mentioned in Section 3, the problem statement is discussed in Section 4, proffered deep intelligent assault prognostication structure is described in Section 5, and the performance evaluation with simulation setup, results, and discussion is presented in Section 6. Lastly, Section 7 presents the conclusion.

2. Literature Review

Numerous studies on the subject are presented in this section, and they present the best DL techniques for an assault prognostication structure for fog-based IoT systems. Samy et al. [1] proposed a framework for attack detection on several cyber-attacks using DL technique resulted high detection rate in multi classification with 99.65% and 99.96% detection accuracy in binary classification, respectively. Lawal et al. [2] used signature- and anomaly-based methods designed framework with two modules for oddity detection. Obtained accuracy for binary and multi classification by module-2 with 99% and 97% for average recall, precision, and F1-score using XGBoost classifier. Module-2 showed six times lesser performance than module-1. Puthal et al. [3] proposed advanced research issues needed for fog architecture and raised chances of threats and discussed the overcoming of threats at each layer in a three-layered architecture. Sudqi Khater et al. [10] considered ADFA-LD and ADFA-WD datasets to address problems on latency,

mobility support, and location awareness on cloud using MLP model of lightweight IDS, which resulted in 92%, 94%, and 95% F1-measure, accuracy, and recall on ADFA-WD dataset. Obtained F1-measure, recall, and accuracy with 74% using Raspberry Pi on ADFA-LD dataset. Bhushan's [20] DDoS attack defence framework is proposed on Kali Linux Machine using LOIC on TCP traffic, allowed only legal request for accessing on cloud by framing rules at fog layer using fog defender. Priyadarshini and Barik [21] designed new DDoS defence model using DLMs by obstructing malicious packets transferred to cloud in order to overcome DDoS attacks on ISCX 2012 IDS and CTU-B Botnet datasets. The experimentation resulted with accuracy of 98.88% accuracy with 10-fold cross validation scheme. Chaudhary et al. [11] made survey on domain of computing and inspect subsisting things related to privacy, security, confronts, limitations, and open directions of research. Douligeris and Mitrokotsa [9] discussed elaborately on segregation of DDoS attack system, advantages, disadvantages, and techniques of defence models. Potluri et al. [12] presented various algorithms like machine learning, deep learning, neural network, blockchain, software defined networks, and genetic algorithm in cloud environment for detection and prevention mechanism. Kalaivani and Chinnadurai [22] designed fog computing intrusion detection model using to predict attacks by CNN and LSTM on NSL-KDD dataset with 96.5% accuracy. To prevent from malicious users the model is deployed in fog layer. This model is used in predicting multiclass attack classifications. By taking into account a variety of criteria, Churcher et al. [23] performed a comparison of various machine learning techniques for binary and multilabel classification. Kilincer et al. [24] performed a comparative study using different ML algorithms on five different datasets, namely, CSE-CIC IDS-2018, UNSW-NB15, ISCX-2012, NSL-KDD, and CIDDS-001. On comparison, the decision tree classifier proved to be better than the remaining two classifiers, SVM and KNN. Many such related research works can also be found in [25–36].

The research gaps that are identified from the study on distinct attack detection frameworks are observed, for instance, (i) performance accuracy or exactness is evaluated on smaller datasets with fewer attributes which fall behind in better attack detection. Hence, we considered newer datasets DDoS-SDN [16] and IoTID20 [19] datasets with huge number of instances and attributes. (ii) Even with the increase in dataset size, most of the prognostications are made on conventional ML algorithms which do not yield better accuracy for attack detection and became cumbersome to decide best ML algorithm on selected datasets. (iii) From the observations on many datasets, we listed only fewer number of attacks and hence need to be considered dataset with more number of attacks which helps in better prognostication of attacks.

3. System Design

In this part, a framework is considered with both the NW and assault designs. The network model portrays about the organization part, the network arrangement, and the correspondence between the organization parts. Attack model portrays how the perpetrator attacks the organization. The notations used in this paper are shown in Table 1.

3.1. Network Design. The network design is designed with a three-layered architecture containing cloud, fog, and IoT end devices in the top layer, amid layer, and bottom layer, respectively, [1–3, 23, 24] as depicted in Figure 1. The Cloud Node (CN), an upper layer that stores updated behaviours (attacker/normal) of the end devices as centralized data storage which is connected to amid layer through gateway (GW) and base station (BSS) using either wired or wireless communication.

The amid layer, called the secure fog layer, containing k number of fog nodes $\mathrm{FN} = \{\mathrm{FN}_1, \mathrm{FN}_2, \ldots, \mathrm{FN}_k\}$ which performs computations, localized communication, and data storage for the nearby IoT end device. They likewise record the way of behaving of the devices promptly. The FN for the most part comprises of a CMFN and MMFN. The CMFN of a FN is empowered where the CMFN is prepared with a DL model to play out an errand to anticipate the ways of behaving of the IoT devices which speaks with the FN in closeness. The FNs are likewise associated with one another through wired/remote interchanges for informational correspondence among them. The secure fog layer is associated with the upper cloud layer through GW and BSs. The correspondence happens utilizing wired/wireless interchanges. The secure fog layer is likewise associated with the lower layer utilizing GW and BSs, through which correspondence happens.

The lowest layer referred to as the sensing layer that mainly composed of IoT devices $\{\mathrm{iot}_1, \mathrm{iot}_2, \ldots, \mathrm{iot}_l\}$ which does enormous amount of end clients information or solicitations to fog or cloud for quick calculation and administration. For communication with the cloud or fog layer, the IoT devices use BSs and GT.

3.2. Assault Design. The peculiarity in the NW now and again causes diversion from the ordinary progression of traffic, which prompts an assault by the attacker P_k. In a fog-based IoT environment, attackers may originate from IoT devices, protocols, applications, and software. Vulnerabilities can arise on various device parts such as web interface, memory, and firmware. Protocols in IoT end devices, by means of communication channels and related applications and software, are also prone to security issues and attacks [20, 21, 37]. Figure 2 shows a typical attack sequence model. By taking possession of the IoT devices that are connected to the closest fog node, the attacker launches various attacks on the fog nodes in the fog layer.

4. Problem Statement

The problem statement defined in the model with l number of IoT end devices as $I = \{i_1, i_2, \ldots, i_l\}$ communicates with k FNs with distinct communication behaviours denoted as $\mathrm{CB} = \{\mathrm{cb}_1, \mathrm{cb}_2, \ldots, \mathrm{cb}_l\}$, where $k \leq l$ and each cb_k has set of communication instances (CI) at different time intervals

Table 1: Notes and elaborations.

SI. no.	Notation	Description
i	CN	Cloud server
ii	GW	Network gateway
iii	BS	Base-station
iv	FN	Set of fog nodes
v	FN_i	i^{th} fog node
vi	CMFN	Compute module of fog node
vii	MMFN	Fog node's memory module
viii	iot_i	i^{th} IoT device
ix	P	Perpetrator/attacker
x	I	Established group of IoT devices
xi	CB	Group of behavior instances
xii	cb	Behavior instance
xiii	T	Time instance
xiv	Accr	Accuracy
xv	DTT	Dataset for training and testing
xvi	T_r	Train data
xvii	T_s	Test data
xviii	TAT	Time of total service
xix	TL	Final Behavior's label
xx	CBDT	Communication behavior detection time
xxi	TTR	Time to refurbish/update FN
xxii	Accr	Accuracy

(TI), represented as $CI = \{ci_1, ci_2, \ldots, ci_m\}$ and $TI = \{t_1, t_2, \ldots, t_m\}$ where m is the number of CI with distinct TI between IoT and FN. On communication, IoT with FN considers different attributes to obtain target label which is denoted as either normal (0) or attacker (1) from the dataset where the set of attributes is denoted as $A = \{a_1, a_2, \ldots, a_p\}$. In this work, the main problem is to predict the behaviour of the IoT devices more accurately by training and testing on different standard datasets by implementing DNMLP, LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM DL models.

5. Proffered Deep Intelligent Assault Prognostication Framework

The proffered assault prognostication framework is presented here is designed to tackle the above issue. The framework principally comprises six stages: (1) network configuration/setup, (2) network's data classification setup, (3) deploying deep learning models and configuring the network, (4) identification of assault, (5) behaviour update at cloud, and (6) network update at FN. In Figure 3, the operational flow model of these six steps is shown.

5.1. Network Setup. The cloud CN is first set up as depicted in Figure 3 which offers various kinds of help to the users. According to the framework, the cloud stores the IoT devices' behaviours and furthermore updates them in a convenient way. Then, the FNs are set in the organization in such a way that the IoT gadgets can convey to get administrations in minimal time. The FNs moreover tackles the issues and offer kinds of help to the IoT gadgets in vicinity. The FNs are associated with one another in a wired/remote way. The IoT gadgets at the bottom layer are associated with the FNs in their vicinity by making use of BSS and GW. They sends and gets information remotely by utilizing 4 G/LTE/3 G/WiMAX. The BSs and GT likewise sends and gets information remotely by utilizing 4 G/LTE/3 G/WiMAX. It is outside the purview of this study to deploy fog nodes and cloud nodes wherever necessary. We just pay attention to how the layers are connected and how the various network elements communicate.

5.2. Network's Data Classification Setup. On establishing network connections at each tier, the FNs are empowered with AI (utilizing DL model) to anticipate the behaviour of the IoT devices. The DL model is carried out at the computing module CMFN of the FN and the model is chosen based on its highest prediction accuracy. For attack detection, the models are trained in prior with what are considered standard datasets. Before training the model, the datasets undergo with various preprocessing steps for selection of features by means of missing value handling, feature scaling, and one-hot encoding. A diagram depicting the preprocessing procedures is shown in Figure 4, the details of which are explained as follows.

5.2.1. Data Preprocessing. The preprocessing of the dataset is as follows:

(1) Handling of missing values: the DL model encounters problems when a sizable fraction of the datasets utilized for classification have missing values. According to the proffered framework, we dealt with the missing values by eliminating the columns or rows which have zeros or null values. Subsequently, we additionally search for mean and median techniques by supplanting the missing values with mean or median. Notwithstanding, it is just employed for numeral data.

(2) Feature scaling: datasets having features of variable types and values will need to have their features scaled to meet the specifications. Normalization and standardization are two of the most well-known methods. To put it simply, the normalization method is used if the data does not have a Gaussian distribution, and the standardization method is used otherwise. The term "normalization" refers to the process of adjusting the absolute values of attributes in a dataset to create a consistent scale without affecting the relative variances between values. In the process of "standardization," the mean is lowered to zero and the standard deviation is raised to one.

(3) One-hot encoding: since the DL model cannot process categorical information, it is necessary to transform the dataset's categorical features into numeral data using one-hot encoding in order to improve prediction. The categorical data is transformed using this method into a categorical new vector, which maps to an integer and each integer is represented by a binary vector.

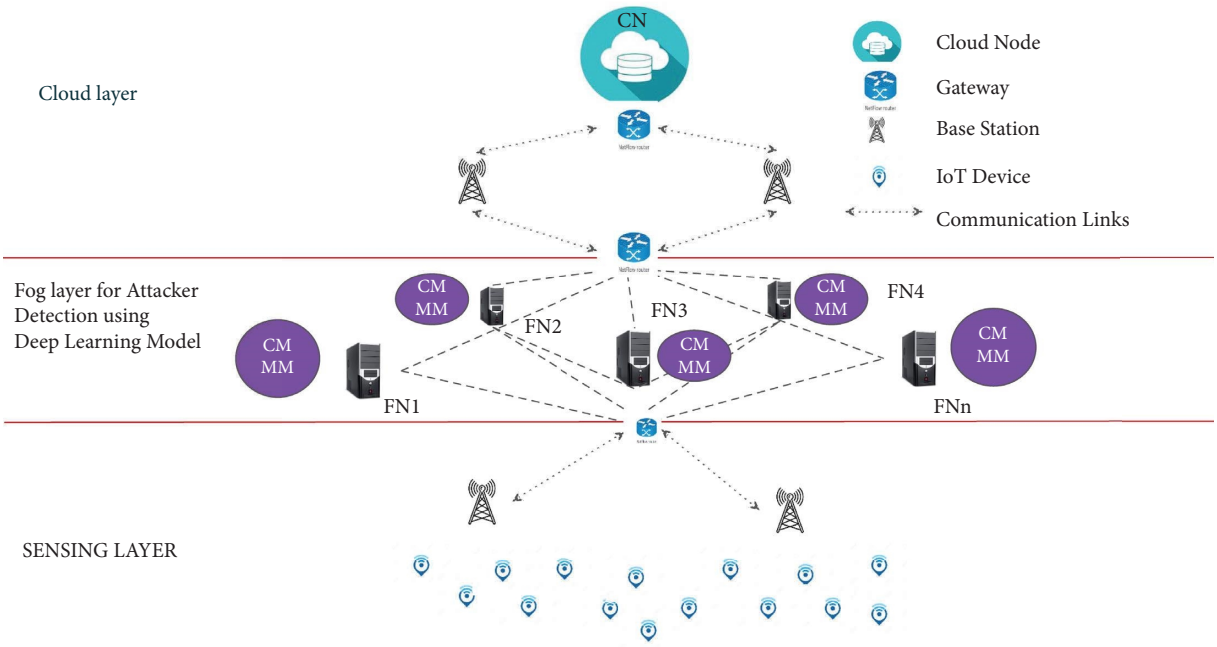

Figure 1: System architecture.

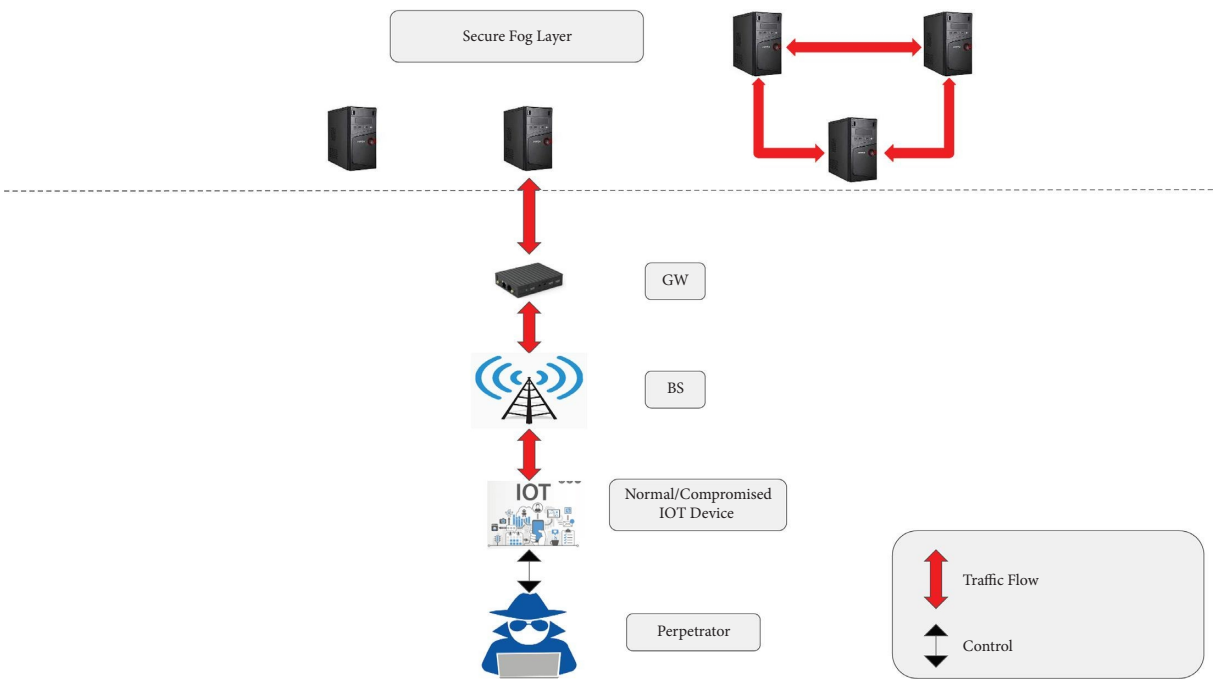

Figure 2: Assault design.

(4) Attribute/feature selection: the dataset that comprises attributes of which some attributes affect the arrangement of attacks that should be taken out from the dataset. Comparably, attributes containing zero values can likewise be eliminated to improve attribute choice. Using feature ranking and feature correlation, we remove features which degrades the capability of detection of DL algorithms. Figure 4 shows the information prehandling, preparing, and testing that is performed involving DL model in FN.

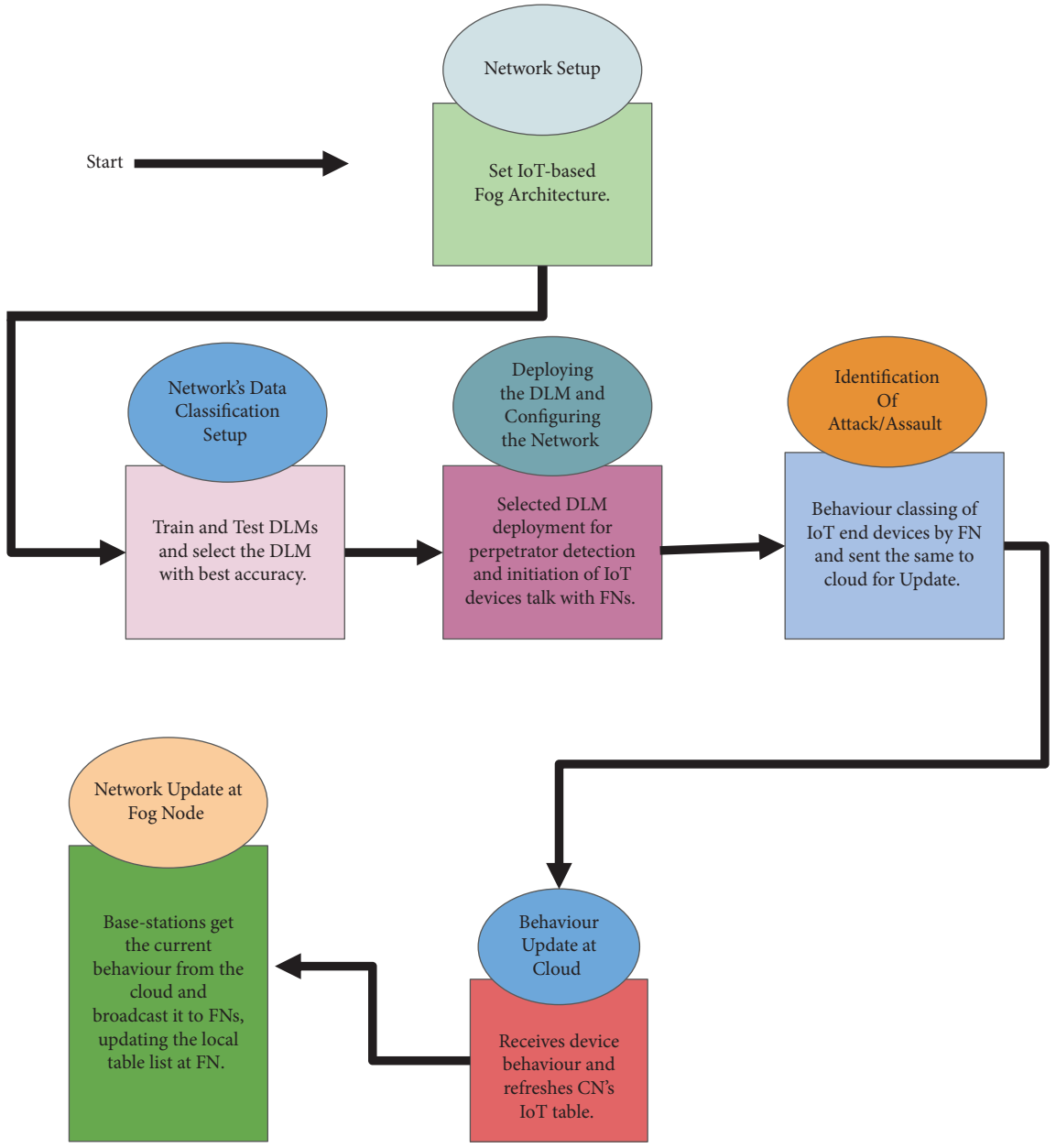

Figure 3: Flow of the proffered assault prognostication framework.

5.2.2. Splitting Dataset. Here, the dataset is partitioned into training and test dataset after completion of data preprocessing. The DLM is trained on training dataset and model accuracy of prediction is tested on test set. The partition process of considered dataset is 80 : 20 ratio.

5.2.3. DLM Used for Prognostication of Attack Behaviour. The fog nodes CMFN are trained on training dataset after partition using DLMs. In the following sections, we considered various DLMs for IoT devices behaviour prediction [1]. The models used are DNMLP, LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM.

(1) DNMLP. Fundamentally, an input layer, an output layer, and a hidden layer with an arbitrarily chosen number of hidden layers make up a DNMLP architecture [10]. Except for the input layer, every neuron in this layer uses a nonlinear activation function. Information flows forward in the DNMLP in order to be described, and the neurons are also set up with a backprop algorithm. The DNMLP design's first step takes into account the sum of information values i_k multiplied by w_k:

$$i_k w_k = i_1 w_1 + i_2 w_2 + \cdots + i_n w_n. \quad (1)$$

In the subsequent advance, bias b_i is added as follows:

$$Y = i_k w_k + b. \quad (2)$$

Now Y value is advanced through the activation function ReLU or Softmax, generally denoted by $\widehat{y}$:

$$\widehat{Y} = \text{maximum}(0, Y). \tag{3}$$

The above function will return zero if $Y < 0$, and if $Y \geq 0$, the result is just input. Now that the final step loss $(Y - \widehat{Y})^2$ has been calculated, it should be limited if it is higher by modifying w_k and b, which should be feasible with an optimizer. As a result, the cost function is calculated as $\sum_{i=1}^{k} = (Y - \widehat{Y})^2$. We arrive at global minima using the backprop technique in a predetermined amount of cycles, and we can consider this to be the success of preparing DNMLP.

(2) LSTM. LSTM is specifically made to address the issue of RNN's long-term dependencies [38]. It is employed for categorizing data and producing prognostications. Cell state, input gates, forget gates, and output gates make up each LSTM unit. It is employed in language modeling, network anomaly detection, picture captioning, and other processes. Because LSTM can retain data for a long time, it is frequently used to categorize data. A chain of LSTM units can be depicted as in Figure 5.

An LSTM cell's progressive flow is governed by the following equations:

$$e_t = sigm\left[\left(w_e.a_{t-1}\right) + \left(w_{ep}.p_t\right) + b_e\right], \tag{4}$$

$$n_t = sigm\left[\left(w_n.a_{t-1}\right) + \left(w_{np}.p_t\right) + b_n\right], \tag{5}$$

$$x_t = \text{Htangent}\left[\left(w_x.a_{t-1}\right) + \left(w_{xp}.p_t\right) + b_x\right], \tag{6}$$

$$s_t = s_t^e + s_t^n, \tag{7}$$

$$y_t = sigm\left[\left(w_y.a_{t-1}\right) + \left(w_{yp}.p_t\right) + b_y\right], \tag{8}$$

$$a_t = \text{Htangent}\left[\left(s_t\right).y_t\right], \tag{9}$$

where e_t is forget gate, a_t is hidden state, n_t is input gate, x_t is cell state, y_t output gate, and s_t is cell vector.

(3) Bi-LSTM. It stands for Bidirectional LSTM and works on historical data for extracting spatial features and bidirectional time dependencies [40]. It has been developed for many applications, like protein structure prediction, handwritten recognition, and speech recognition. In the former and future sequences, the best benefits result from the input sequence. In this process, the first layer is given an input sequence, and the next layer is given an input of reverse copy, where the primary and secondary layers are connected with the same layer of output.

(4) GRU. GRU is a mechanism of RNN in similar fashion to LSTM but with no output gate [1, 41]. It is considered a variant of LSTM used to overcome the vanishing gradient problem by means of an update and reset gate. Both gates are utilized to regulate the movement of information into and out of memory. GRU outperforms LSTM, which takes longer on large datasets, in comparison. GRU performs better than LSTM for smaller datasets. Speech signal modeling, handwriting recognition, and polyphonic music modeling all make extensive use of GRU. The update gate (u) and the reset gate are the two gates that make up the GRU (rs). The calculation of u and rs gates at time $t-1$ is illustrated in the following equations:

$$u_{t-1} = \text{sigmoid}\left[\left(W_{u-1}.h_{t-2}\right) + \left(W_{u-1}.p_{t-1}\right) + b_u\right], \tag{10}$$

$$rs_{t-1} = \text{sigmoid}\left[\left(W_{rs-1}.hs_{t-2}\right) + \left(W_{rs-1}.p_{t-1}\right) + b_{rs}\right], \tag{11}$$

$$cs_{t-1} = \tanh\left[\left(W_{hs}.p_{t-1}\right) + W_{hs}\left(hs_{t-2} \odot rs_{t-1}\right) + b_{hs}\right], \tag{12}$$

$$hs_{t-1} = (u \otimes cs) \oplus \left((1-u) \otimes hs_{t-2}\right). \tag{13}$$

(5) CNN + LSTM. It is a blended DLM intended for visual time series expectations and text-based classification, such as video depiction and image chaining. Figure 6 depicts the constructed CNN + LSTM model. The CNN + LSTM engineering consolidates CNN layers for feature extraction from inputs and LSTM layers for time sequence expectation. CNN + LSTM has accomplished upgrades in speech recognition on DNN. It is utilized in visual acknowledgment and elucidation in [42].

(6) HEM. In the proposed architecture, a hybrid ensemble method Figure 7 is used for attack detection at FN [25]. This model is constructed into three stages: data preprocessing, hybrid ensemble mechanism, and data gathered from IoT end devices. In the second stage, the hybrid ensemble mechanism is implemented by considering k-fold cross-validation where $k = 10$, which needs to be trained on five different ML algorithms, namely, logistic regression (LR), decision tree (DT), XGBoost, K-Nearest neighbour (KNN), and Gaussian naive bayes (NB). The considered data set is partitioned into k parts, of which k^{th} portion is served as testing set, and the left over $k-1$ part is served for training. On this $k-1$ and k^{th} part, the above five algorithms are executed collaterally, which obtains five different prediction results denoted as $R1$, $R2$, $R3$, $R4$, and $R5$ that are used for final classing on the voting classifier. In the third stage, the data from IoT end devices is collected at FN as test data, which was tested for classing the attack behaviour.

5.2.4. Dataset Description. The DLMs are assessed on old and novel datasets to distinguish the various attacks and characterize the end client conduct (benign/assailant). The datasets used in this framework for training and testing are discussed as follows:

(1) DDoS-SDN: The DDoS-SDN dataset is browsed in Mendeley Data, which includes 104345 records having 23 traits [16]. It is owned to recognize data traffic as harmless or vindictive in light of TCP Syn, UDP flood, and the ICMP attacks. Switch$_i$$d$, Packet$_c$ount, byte$_c$ount, and so on are among the traits. The data-traffic characterization marked 0 for the harmless client and 1 for the malignant client.

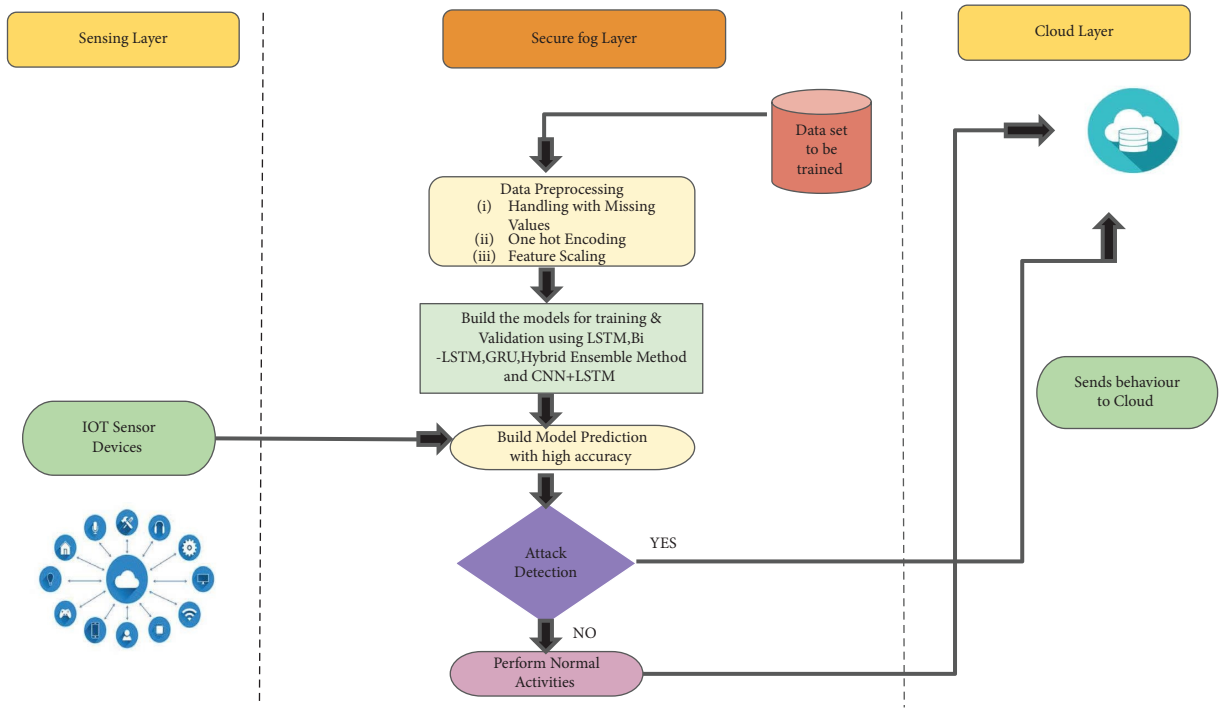

Figure 4: Data preprocessing steps.

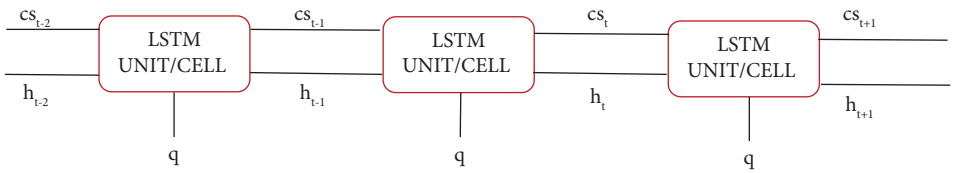

Figure 5: Chain of LSTM units [39].

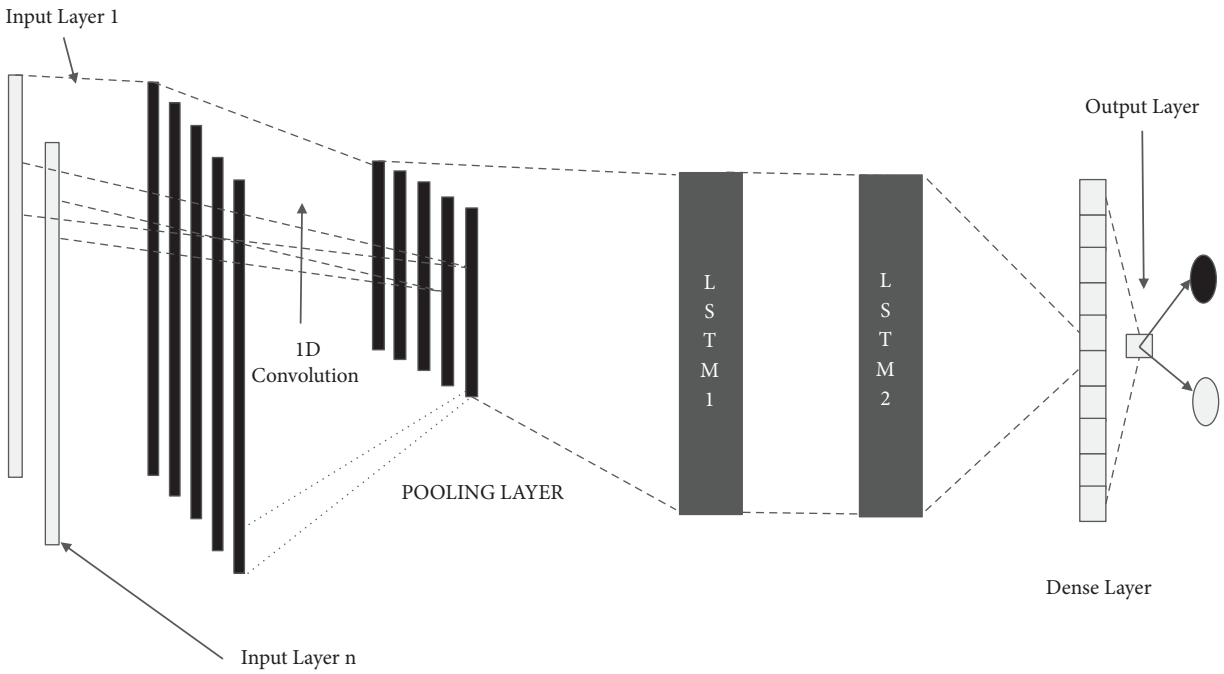

Figure 6: CNN + LSTM architecture.

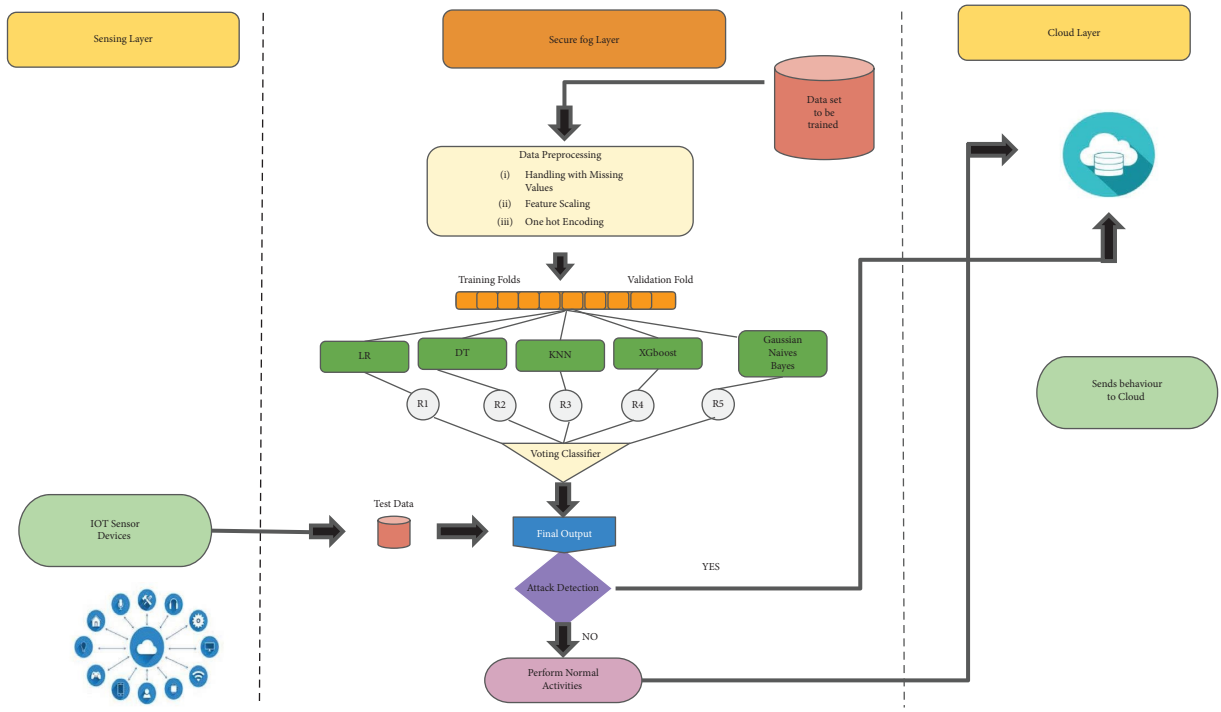

Figure 7: HEM architecture.

The 15 customizable properties in the dataset include 14 features and 1 target variable. The binary classification of the target label is 0 (normal/benign user) and 1 (attacker/assaulter). One category trait named protocol, which is one-hot encoded, is present in the dataset.

(2) NSL-KDD: The NSL-KDD dataset [17] is constituted into reality, by means of modified and tidied-up variant of the KDD99 from the University of New Brunswick. The NSL-KDD dataset has browsed the Kaggle Repository with 148517 (test and train) records with 43 traits. Out of total records, 77054 records are normal and 71463 are anomalies which clearly exhibits its balanced nature. The dataset is converted into CSV format. Attributes that have no impact on the dataset were discarded and thought about just 19 elements. The categorical traits protocol_type, service, and flag were one-hot encoded to refashion over into mathematical traits. The protocol_type attribute is of three types, namely, icmp, tcp, and udp, the service attribute is of 70 types and flag is 11 different types. After one-hot encoding dataset features were increased to a hundred in number. This dataset contains two class labels, namely, normal and anomaly. There are four unique attack types in the NSL-KDD dataset: denial of service (DoS), user to root (U2R), probe, and remote to local (R2L).

(3) UNSW_NB15: The third dataset is UNSW_NB15 is chosen from UNSW_NB15 || Kaggle, developed by Intelligent Security Group, UNSW, and Canberra, Australia store, with 2540044 instances and 49 features [18]. In this work, the dataset considered from Kaggle contains 257673 instances including UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv with 45 features including 01 class label. It was customized to 43 features including 01 class label. The categorical feature proto comprised of 133 unique labels and only 15 unique labels are considered in this work and remaining are discarded. So that a total of 242432 instances are taken into account. It describes in total of nine categories as attackers (fuzzers, analysis, backdoors, DoS, exploits, generic, reconnaissance, shellcode, and worms) with 164673 instances and one as normal with 93000 instances. The target label is divided into two categories as 0 (normal) and 1 (attacker).

(4) IoTID20: The IoTID20 is the fourth dataset used in the proposed work, with 86 columns, in which three are label features and 625783 instances, which is customized to 68 columns using correlation [19]. The three label features are named as binary, category, and subcategory. The binary label feature is distributed as normal and anomaly with 40073 and 585710 records, respectively. The category label feature is distributed as normal, DoS, mirai, MITM, and scan with 40073, 59391, 415677, 35377, and 75265 records. The last subcategory is distributed as ten label features (normal, DoS, mirai ack flooding, mirai Brute Force, mirai HTTP flooding, mirai UDP flooding, MITM, scan host port and scan port OS)

with 40073, 59391, 55124, 121181, 55818, 183554, 35377, 22192, and 53073 records. The main advantages of the IoTID20 dataset; it imitates a cutting edge pattern of IoT network correspondence; it is among the couple of openly accessible IoT intrusion detection dataset.

5.3. Deploying Deep Learning Models and Configuring the Network. We select the DL model after completing the training of the above models on the considered datasets, resulting in high accuracy in prognosticating the behaviour of IoT end devices as normal or malicious. The maximum accuracy attained after training and testing each model is used to make the model selection. Now that the chosen model has been deployed, the entire architecture is prepared for real-time processing where the FN and IoT end devices communicate with one another on the CMFN in the fog layer of fog nodes. The finest algorithm for choosing DL models is Algorithm 1. The network configuration and DL model installation at the fog layer is shown in Algorithm 2.

Theorem 1. *The IoT device iot_is total result time (TRT) is denoted by TRT_{ioti}.*

Proof. Consider an IoT device iot_i nearer to fog node FN_i which sends REQ to FN_i. The time to send REQ is formulated as follows:

$$T_{iot_i-FN_i} = T_{iot_i-BSS} + T_{BSS-GW} + T_{GW-FN_i}, \tag{14}$$

where T_{iot_i-BSS} is the request time from iot_i to BSS, $T_{\mathrm{BSS-GW}}$ is the request time from BSS to GW, and $T_{\mathrm{GW-FN}_i}$ is the request from GW to FN_i. The execution time to process the request by FN_i is represented as $T_{\mathrm{execution}_{FN_i}}$ as follows:

$$T_{\mathrm{execution}_{FN_i}} = T_{\mathrm{queue}_{FN_i}} + T_{\mathrm{compute}_{FN_i}}, \tag{15}$$

where $T_{\mathrm{queue}_{FN_i}}$ is the time spent in waiting queue and $T_{\mathrm{compute}_{FN_i}}$ is the computation time for processing the request to obtain the outcome. Then, outcome is passed to the iot_i with a time of $T_{FN_i-iot_i}$:

$$T_{FN_i-iot_i} = T_{FN_i-GW} + T_{GW-BSS} + T_{BSS-iot_i}, \tag{16}$$

where $T_{\mathrm{FN}_i-\mathrm{GW}}$, $T_{\mathrm{GW-BSS}}$, and $T_{\mathrm{BSS}-iot_i}$ are all the time to send outcome to iot_i. Therefore, the TRT is calculated as follows:

$$TRT_{iot_i} = T_{iot_i-FN_i} + T_{\mathrm{execution}_{FN_i}} + T_{FN_i-iot_i}. \tag{17}$$

□

5.4. Identification of Assault. From the above two algorithms, it is depicted the way in which a DLM model is chosen and introduced in the FNs. A while later, the IoT devices began correspondence with the FNs in nearness for getting administrations. Notwithstanding, after the communication the proposed model predicts the way of behaving of the IoT devices from recorded CB. As the CMFN_i is empowered with DLM, it can predict the way of behaving of the IoT devices (malicious or benign). On completion of classification, the refreshed behaviour of the IoT device at FN is transferred to the cloud CN for accumulation and refurbish. Algorithm 3 shows the classing behaviour of IoT end devices by FN.

Theorem 2. *$Aiot_i$IoT device's communication behaviour detection time (CBDT) is expressed as $CBDT_{iot_i}$.*

Proof. Allow l number of CB for l number of IoT gadgets in the FN queue. Consequently, the $\mathrm{CBDT}_{\mathrm{iot_i}}$ of an IoT device iot_i of a fog node FN_i is computed as follows:

$$CBDT_{iot_i} = T_{\mathrm{queue}_{cb_i}} + T_{\mathrm{prognostication}}, \tag{18}$$

where $T_{\mathrm{queue}_{cb_i}}$ is the time spent in the FN_i waiting queue of the IoT device communication behaviour and $T_{\mathrm{prognostication}}$ is the time needed to detect the IoT device communication behaviour by FN_i. □ □

The time complexity to test (18) depends on its execution, i.e., the model we selected for deploying on FN_i. It is clearly observed from Section 6.2 that the finest DLM obtained for classifying the behaviour of IoT end devices is LSTM model which is deployed on FN_i. The complexity of LSTM model with multiple LSTM layers always depends on its implementation. Generally, any model with neural networks is tested by means of an onward pass. To obtain the complexity for any LSTM network with layers we need to consider the LSTM units which are connected in a recurrent manner.

Equations (4)–(9) which represent the onward pass of LSTM layer generate its time complexity on n_t as $O(n(d + n + 2))$ where n and d are dimensions. The computation of e_t, x_t, and y_t are same as n_t and, thus, the complexity will be $O(4n(d + n + 2))$. Considering the cell vector s_t and the hidden state a_t time complexity of each is $O(2n)$. Hence the total time complexity for single LSTM layer onward pass is $O(4n(d + n + 3))$. According to the (18) the complexity of $\mathrm{CBDT}_{\mathrm{iot_i}}$ also depends on $T_{\mathrm{queue}_{cb_i}}$ for the cb_i insertion into the queue and is (1). Hence, the total complexity for $\mathrm{CBDT}_{\mathrm{iot_i}}$ in a single onward pass is only $O(4n(d + n + 3))$.

5.5. Behaviour Update at Cloud. The cloud node CN updates the IoT device information table with the updated behaviours after receiving the behaviour of IoT end devices from the FN. The device information table is updated after receiving the responses from FN_i. The behaviour update at cloud node CN is shown by Algorithm 4. Here, the storage operations which are external to the main memory depend on the table structure maintained, the type of indexing that is being supported, the number of disc accesses that are done, the complexity of the query, etc.

5.6. Network Update at FN. Here, the cloud CN transmits the smart gadgets $\mathrm{TL}_{\mathrm{cb_i}}$ to the FNs via transmission links $\mathrm{GW}_{\mathrm{CN}}$, $\mathrm{GW}_{\mathrm{CN}}$ to BSS, BSS to GW_i, and GW_i to FN_i to update the local tables at FN closest to BSS. Further if transmission occurs among neighboring FNs, it is done only when the behaviour is verified using the local database. If it is discovered to be an assaulter, it terminates additional

Input: $\text{DLM}_1, \text{DLM}_2, \ldots, \text{DLM}_n$, _Dataset
Output: FINEST_DLM_Chosen
(1) DTT←DataPreprocessing(_Dataset);
(2) T_r, T_s←train_test_split(DTT));
(3) **for**DLM_1 to DLM_n **do**
(4) Train(T_r);
(5) Test(T_s);
(6) Accr_m←Cal_Accr(); ▷ cal: calculate
(7) **end for**
(8) FINEST_DLM_Chosen←Maximum($\text{Accr}_1, \text{Accr}_2, \ldots, \text{Accr}_m$);

Algorithm 1: Method of choosing the finest DL model for the fog tier.

Input:FINEST_DLM_Chosen, $\{FN_1, FN_2, \ldots, FN_k\}$
Output: Outcome
(1) **for**FN_1 to FN_k **do**
(2) CMFN_i←FINEST_DLM_Chosen; ▷ DL model is deployed at all $\text{CMFN}_i i$ is the i^{th} node of the fog layer.
(3) **end for**
(4) NW_init() ▷ NW : Network
(5) {
(6) **for**iot_1 to iot_l**do** ▷ the NW is prepared for operation in real-time.
(7) **if** assistance needed **then**
(8) $iot_i \xrightarrow[\text{Receives RES}]{\text{Sends REQ}} \text{FN}_i$;
(9) $iot_i \leftarrow \text{FN}_i$; ▷ FN_i is the nearest FN
(10) CONN_EST(); ▷ establishment of a connection
(11) FN_i operates REQ;
(12) FN_i stores cb_i; ▷ FN stores the communication behaviour of IoT device.
(13) $\text{FN}_i \xrightarrow{\text{Outcome}} iot_i$; ▷ sends outcome to IoT device.
(14) CONN_CLOSE();
(15) **end if**
(16) **end for**
(17) }

Algorithm 2: Algorithm for DLM installation and network setup.

interaction with the network's adjacent nodes. The network refresh at FN is shown by Algorithm 5.

Theorem 3. *The time to refurbish/update the attacker/assaulter behaviour at FN (TTR) is the total amount of time required by the cloudCN to update IoT end device behaviour at FN.*

Proof. Consider at time T, h attacker devices prognosticated behaviours are denoted as $\{TL_{cb_1}, TL_{cb_2}, \ldots, TL_{cb_h}\}$ for l IoT devices. These prognosticated behaviours are sent as message Msg to the FNs. To calculate TTR, to send Msg from CN to FN is represented as:

$$TTR = T_{CN-GW_{CN}} + T_{GW_{CN}-BSS} + T_{BSS-GW_i} + T_{GW_i-FN_i}, \tag{19}$$

where $T_{CN-GW_{CN}}$ is the time required to send the message Msg from CN to GW_{CN}, $T_{GW_{CN}-BSS}$ is the time required to send the message M from GW_{CN} to BSS, T_{BSS-GW_i} is the time required to send the message Msg from BSS to i^{th} GW, and $T_{GW_i-FN_i}$ is the time required to send the message Msg from i^{th} GW to i^{th} FN. □

6. Performance Evaluation

To test how well the proffered framework works, Python 3 is used as a software requirement, and the core i7-11370 CPU, 3.30 GHz clock speed, and 16 GB RAM are used as hardware requirements. The framework is implemented with various DLM models on four datasets, resulting in different accuracies. The accuracy (Accr), precision (P), recall (R), and F1-Score ($F1\_S$) of DLM models are calculated using confusion matrix parameters, where true positive is (T_P), true negative is (T_N), false positive is (F_P), and false negative is (F_N):

(1) Accuracy (Accr): Accuracy is characterized by the number of correct predictions obtained from the observed values. The notation is as follows:

$$\text{Accr} = \frac{T\_P + T\_N}{T\_P + T\_N + F\_P + F\_N}. \tag{20}$$

(2) F1-Score: The harmonic mean of recall and precision is used to reckon the F1_S in order to provide more accurate results. Below is a representation of an F1_S:

Input: FINEST_DLM_Chosen, $\{FN_1, FN_2, \ldots, FN_k\}$, cb_i
Output: Refurbish cloud node CN
(1) **for** FN_1 to FN_k **do**
(2) **for** all IoT end devices of FN_i **do** ▷ $i = 1,2,3, \ldots, k$
(3) $TL_i = \text{FINEST\_DLM\_Chosen}(cb_i)$; ▷ TL : Target label specifies the communication behaviour of the IoT devices.
(4) $FN_i \xrightarrow{\text{Sends } TL_i} CN$
(5) **end for**
(6) **end for**

Algorithm 3: Classing the behaviour of IoT end devices by FN.

Input: TL_{cb_i} **of** iot_i
Output: refurbish of IoT end device table at CN
(1) iteration()
(2) {
(3) CN accepts TL_{cb_i} of iot_i;
(4) Refurbish_IoTTable(TL_{cb_i}); ▷ refurbish cloud's IoT device table with IoT device's communication behavior.
(5) }

Algorithm 4: Behaviour update at cloud.

Input: $TL_{cb_1}, TL_{cb_2}, \ldots, \text{Target}_{cb_l}$
Output: a refresh of the data in the regional table at FN
(1) iteration()
(2) {
(3) $CN \xrightarrow{TL_{cb_i}} TL_{cb_i}GW_CN$
(4) $GW_CN \xrightarrow{TL_{cb_i}} TL_{cb_i}BSS$
(5) $BSS \xrightarrow{TL_{cb_i}} TL_{cb_i}GW_i$
(6) $GW_i \xrightarrow{TL_{cb_i}} TL_{cb_i}FN_i$
(7) **for** all FNs nearer to BSS **do**
(8) Refurbish_LocalTable(TL_{cb_i});
(9) **end for**
(10) }
(11) **if** communication takes place between FN_i and FN_j **then**
(12) **if** $FN_i == \text{P}$ **then** ▷ FN_j checks its local table.
(13) there is no interaction between the parties;
(14) **else**
(15) communicate;
(16) **end if**
(17) **end if**

Algorithm 5: Network update at FN.

$$F1\_S = \frac{2 \times P \times R}{P + R}. \tag{21}$$

(3) Precision: Precision is a model's consistency in categorizing the model as positive and is denoted as follows:

$$P = \frac{T\_P}{T\_P + F\_P}. \tag{22}$$

(4) Recall: Recall is the ability how well a model can identify positive samples and below is the representation:

$$R = \frac{T\_P}{T\_P + F\_N}. \tag{23}$$

6.1. Attack Simulation Using DLMs. Python 3 DNMLP, LSTM, Bi-LSTM, GRU, CNN + LSTM DL models, and HEM are used to discover the most accurate model. The DDOS-SDN dataset, NSLKDD dataset, UNSW-NB15 dataset, and IoTID20 dataset, [16, 19, 42] are utilized for prognosis. Anaconda's Keras module on TensorFlow is used for deep learning implementation. The SkLearn Package was used to

implement and evaluate the hybrid ensemble model. The matplotlib is used obtain graphs on accuracy and loss performance.

Using the DDOS SDN dataset, NSLKDD dataset, UNSW-NB15 dataset, and IoTID20 dataset, we trained and evaluated the abovementioned models for binary classification (normal or attacker). Different attacks are involved with considered datasets [16, 19, 42] and are utilized to recognize the ability of DNMLP, LSTM, Bi-LSTM, GRU, CNN + LSTM DL, and HEM models for attack identification. In our work to prognosticate the attacks, some features from the considered datasets are discarded on the basis of high correlation among the traits, or those features that don't affect the prognostication. On the expulsion of these attributes, the computation burden is lowered, and thus the framework is built with vital information. Utilizing a standardization strategy, the dataset is scaled on different traits for the fluctuating sizes of values and divided into two proportions of 80 : 20 as train and test data. The point of apportioning the dataset in the proportion of 80 : 20 is to prepare the model with sufficient data and to corroborate the model with suitable data. To procure the most accurate trained model in the proposed framework using the LSTMDL model, we considered a mini batch of 32 with 100 epochs on the Adam optimizer using the learning rate (LR) of 0.001 and considered beta values as arguments for the first- and second-moment exponential decay rate estimates as 0.9 and 0.999, which prevents an adverse effect on optimization for binary classification. The callback function on early stopping is called on TensorFlow, which keeps track of flow to decide the termination condition on validation loss. For the datasets under investigation, NN is built using Keras on TensorFlow using the aforementioned models. In this work, we constructed a model for DNMLP as shown in Figure 8 on new IoTID20 dataset and also a model is built using LSTM is shown in Figure 9, in a similar way the model is built on Bi-LSTM and GRU. On the same dataset, the model is also built on CNN + LSTM as shown in Figure 10. We used ReLU as the activation function in the dense layers of DL models and sigmoid activation function in the output layer as we performed binary classification. Using a stacking approach with a voting classifier, the model is built on HEM using the same IoTID20 dataset as discussed in Section 5.2.3. As described above for constructed models on the IoTID20 dataset, in the same way, constructed models were created on the remaining datasets after performing one-hot encoding. The sequential model is used to create NN with Keras, and it accepts the result of each layer as a contribution to the subsequent layer that uses the add-on model. The dense from the Keras package was used to determine the completely associated layer.

In the implementation of HEM using the stacking approach as discussed in Section 5.2.3, after preprocessing, in stage 1, five algorithms such as LR, DT, XGBoost, KNN, and NB are imported from sklearn machine learning library. In the second stage, we used a voting classifier by importing the package using "sklearn.ensemble import VotingClassifier" for final classing.

6.2. Results and Discussion. The accuracy of the DL models for binary classing on four datasets is evaluated. The experiment evaluation revealed that a LR of 0.001, a mini-batch of 32, and 100 epochs produced the best performance accuracy. The best performance accuracy over all the datasets is obtained with the LSTMDL model as shown in Table 2 and the model accuracy, model loss, model recall, and model precision graphs of the IoTID20 dataset are shown in Figures 11–14. As IoTID20 is a novel dataset on which only ML models are implemented in previous studies [43], in Section 6 we focused on DLM models on the IoTID20 dataset, whose upshots are depicted in graphs. The execution measures for each model on the datasets taken into consideration for binary classification are shown in Table 2. For the IoTID20 dataset, LSTM achieves better accuracy (99.88%) with precision (99.77%), recall (98.4%), and F1_S (99.08%). The discharge of HEM is comparable to that of LSTM and it outperforms the Bi-LSTM, GRU, CNN + LSTM, and MLP models. With UNSW-NB15 dataset, LSTM achieves better performance (94.11%) with precision (95.87%), recall (94.47%), and F1_S (95.16%). Bi-LSTM performs like LSTM and it outperforms the GRU, CNN + LSTM, HEM, and MLP model. With NSLKDD dataset, LSTM achieves better accuracy (99.12%) with precision (99.22%), recall (99.08%), and F1_S (99.15%). MLP performs like LSTM and it outperforms the Bi-LSTM, GRU, CNN + LSTM, and HEM models. With DDOSSDN dataset, LSTM achieves better accuracy (99.7%) with precision (99.6%), recall (99.64%), and F1_S (99.62%). The discharge of GRU and Bi-LSTM performs like LSTM and they outperform the CNN + LSTM, HEM, and MLP models. For binary classification, excluding DDOSSDN dataset, GRU did not perform well. In implementing GRU, we used dropout mechanism at every stage of constructing model. So, we discarded dropout mechanism and implemented L2 regularization in GRU for better performance. On contrast with all the models on the considered datasets, the false-positive rate (FPR/FAR/1-specificity) on LSTM may not outperformed at its best, but on overall comparison LSTM proved to be performed well with FPR.

The ROC-AUC score of LSTMDLM on all four considered datasets are shown in Figures 15–18. ROC curves are attained by marking out T_P rate (TPR/Recall) versus FPR. AUC summarizes the ROC curve and takes the value between 0 and 1 where one indicates the classifier's exactness in prediction and zero, otherwise. It is evident from above graphs in Figures 15–18 that LSTMDLM exhibited higher AUC score which indicates the ability to classify positives and negatives exactly. Remaining algorithms on all four datasets also showed AUC score between 0.98 and 1.

The study of execution measures amid the DLMs and HEM on the considered datasets is depicted in Figures 19–22. In terms of accuracy, LSTM performed better than the considered DLMs and HEM, as shown in Table 2 with bold values. The accuracies of all DLMs and HEM with binary classing are shown in Figure 23. Hence the LSTM model is prognosticated to be greater in rank compared to all others.

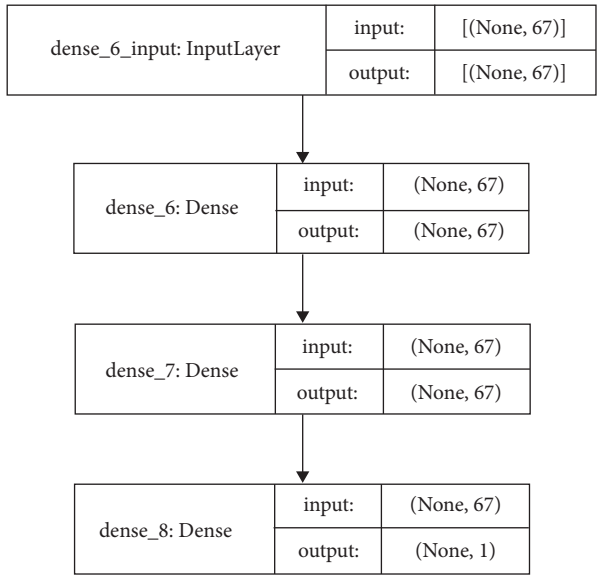

Figure 8: Constructed DNMLP model.

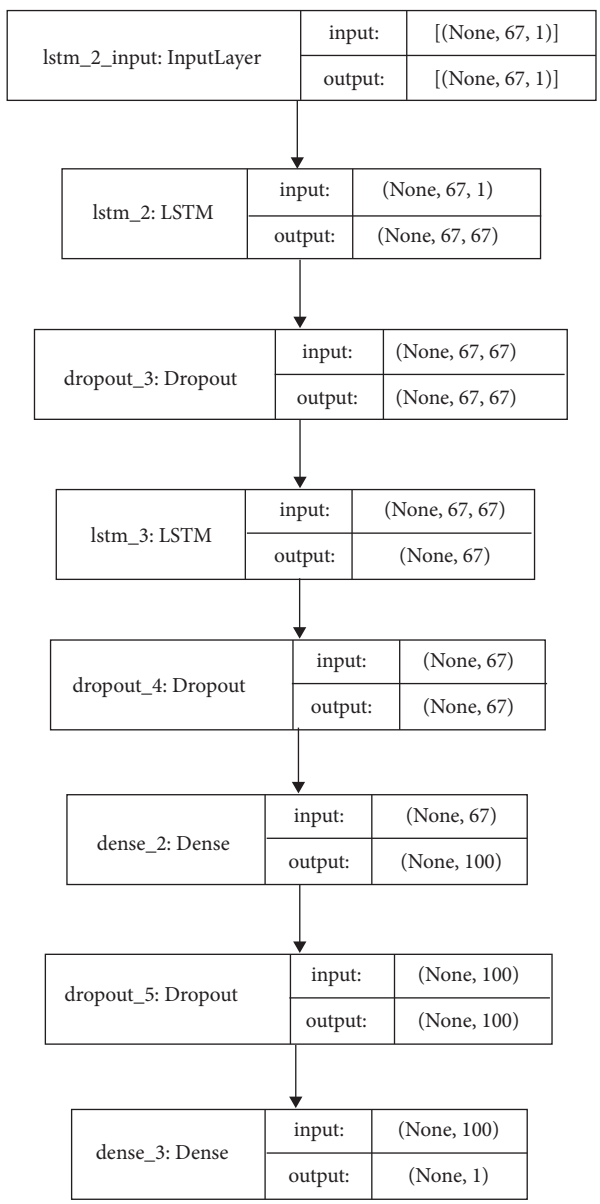

Figure 9: Constructed LSTM model.

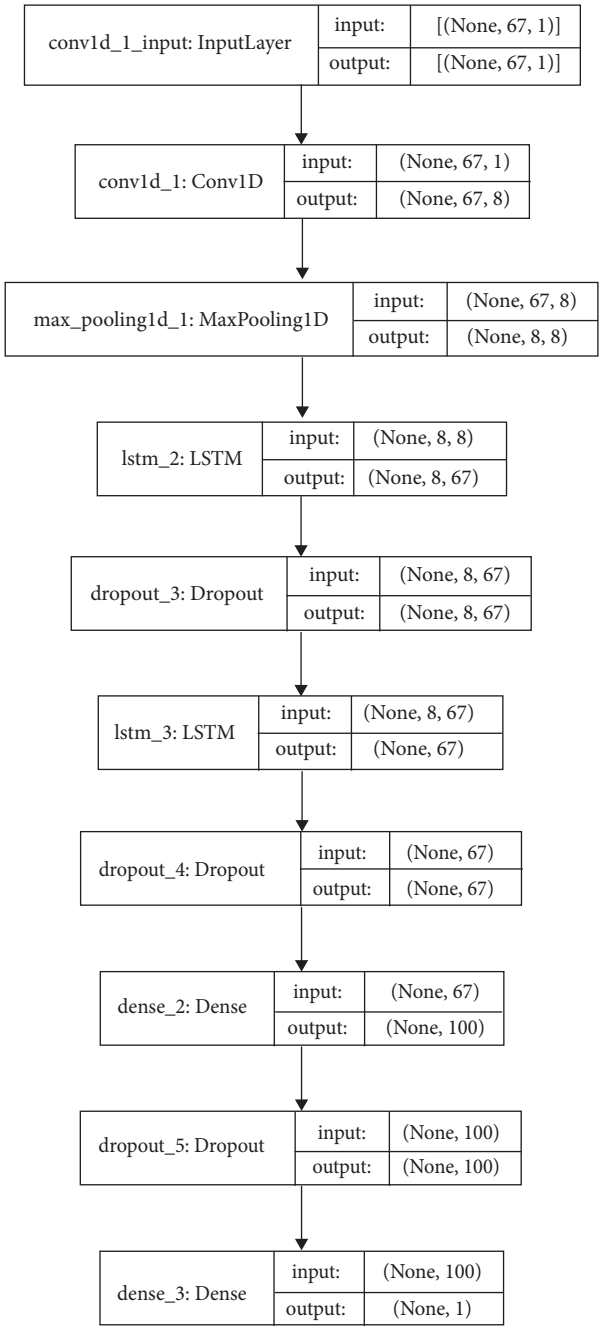

Figure 10: Constructed CNN + LSTM model.

Utilizing datasets, we trained and evaluated DLMs and HEM model for binary classification and found LSTMDLM showed preferred exactness over any remaining models in anticipating the way of behaving of the IoT end devices as normal or attack. On using balanced dataset, accuracy is only considered to be an essential measure in assessing a model. But in this work, excluding NSLKDD, the other remaining datasets are imbalanced, hence there is possibility of having more F_P and F_N. In these circumstances, it is smarter to pay attention to the other execution measures like precision, recall, and $F1\_S$. Recall, in all actuality, does just think about F_N and T_P and subsequently, recall might be high. Precision really does just consider F_P and T_P, it might endure with low worth. The $F1\_S$, will have its significance to choose the presentation of the model furthermore. It is obviously clear by the outcomes showing the most elevated worth of $F1\_S$ (99.62%, 99.15%, 95.16%, and 99.08%) on DDoS-SDN (Mendeley Dataset), NSLKDD, UNSW-NB15, and IoTID20 datasets with LSTMDLM as shown in Table 2.

Table 2: Performance metrics of considered DLMs.

Dataset name	ML/DL model	Accr (%)	Precision (%)	Recall (%)	F1_S (%)
DDOS SDN	MLP	99.46	99.25	99.4	99.32
	LSTM	**99.7**	99.6	99.64	99.62
	Bi-LSTM	99.63	99.78	99.29	99.53
	GRU	99.66	99.55	99.6	99.57
	HEM	97.87	99.17	97.28	98.22
	CNN + LSTM	96.43	95.71	95.27	95.49
NSLKDD	MLP	99.05	99.04	99.12	99.08
	LSTM	**99.12**	99.22	99.08	99.15
	Bi-LSTM	98.95	98.85	99.13	98.99
	GRU	98.43	98.41	98.56	98.48
	HEM	98.28	99.62	96.8	98.19
	CNN + LSTM	98.19	97.79	98.72	98.25
UNSW-NB15	MLP	93.41	95.74	93.51	94.61
	LSTM	**94.11**	95.87	94.47	95.16
	Bi-LSTM	94.04	95.86	94.43	95.14
	GRU	93.57	95.75	93.76	94.74
	HEM	94.05	92.08	92.37	92.22
	CNN + LSTM	92.85	94.37	94.05	94.21
IoTID20	MLP	99.84	99.7	97.84	98.76
	LSTM	**99.88**	99.77	98.4	99.08
	Bi-LSTM	99.86	99.57	98.27	98.92
	GRU	99.84	99.61	98.01	98.8
	HEM	99.84	99.83	99.99	99.92
	CNN + LSTM	99.76	99.7	96.58	98.11

The simulation condition is configured as a three level framework with one cloud server coupled to numerous fog nodes to investigate the expandability problem. We make the assumption that the closest fog nodes are connected to 10–100 IoT devices (smart gadgets) in the final layer. For instance, if there is 1 fog node and 10 smart gadgets, then 10 smart gadgets can connect directly to the fog node. If there is more than 1 fog node, then the number of fog nodes splits the number of smart gadgets equally to provide the required service. Therefore, one fog node will provide aid to 5 smart gadgets if there are 2 fog nodes. In this case, we'll assume that a smart gadget links to the fog node and produces one sample (row). The fog node then processes this sample to forecast behaviour (attack/assault or normal/benign). The average CBDT from the aforementioned examination using DNMLP is discovered to be 0.0000672 seconds for 10 smart gadgets, 0.0024924 seconds for 10 smart gadgets for LSTM, 0.004164 seconds for 10 smart gadgets for Bi-LSTM, 0.0021 seconds for 10 smart gadgets for GRU, 0.000476 seconds for 10 smart gadgets for CNN + LSTM, and 0.008 seconds for HEM, respectively. In this experiment, we looked at how the number of smart gadgets compared to the number of fog nodes might affect the time it takes to identify behaviour. Behaviour detection time (BDT) is the period of time during which fog nodes using DNMLP, LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM can determine whether a certain number of smart gadgets are benign or an assaulter. The variables and values for the NW simulation are displayed in Table 3.

From Figures 24–28, it is apparent that as the number of Internet of Things grows in the NW, so does the time it takes for all IoT devices to detect their behaviour. The result is depicted in Figure 24 when there are 1 fog node in the NW and 10–100 IoT devices. According to this graph, DNMLP has a faster time to detect behaviour than LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM. The DNMLP, LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM average CBDT for 100 IoT devices were determined to be 0.003695 sec, 0.22902 sec, 0.1155 sec, 0.02618 sec, and 0.44 sec, respectively. The result is shown in Figure 25 when there are 3 fog nodes in the NW and 100 smart gadgets overall. According to this graph, DNMLP has a faster time to detect CB than LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM. The average CBDT for 100 IoT devices is determined to be 0.001232 seconds for DNMLP, 0.045694 seconds for LSTM, 0.0385 seconds for CNN + LSTM, and 0.146666 seconds for HEM. The outcome is shown in Figure 26 when there are 5 fog nodes in the NW and 100 smart gadgets in total. According to this graph, DNMLP has a faster time to detect CB than LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM. The average CBDT of 100 IoT devices is determined to be 0.000739 sec, 0.027416 sec, 0.045804 sec, 0.0231 sec, 0.005236 sec, and 0.088 sec for DNMLP, LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM, respectively. The result is shown in Figure 27 when there are 7 fog nodes in the NW and 100 smart devices in total. According to this graph, DNMLP has a faster time to detect behaviour than LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM. The average CBDT of 100 smart gadgets is determined to be 0.000527 sec, 0.019583 sec, 0.032717 sec, 0.016499 sec, 0.00374 sec, and 0.062857 sec for DNMLP, LSTM, Bi-LSTM,

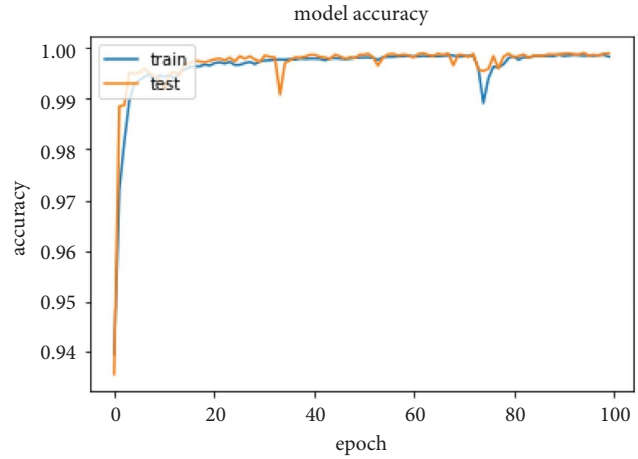

Figure 11: Model accuracy of IoTID20.

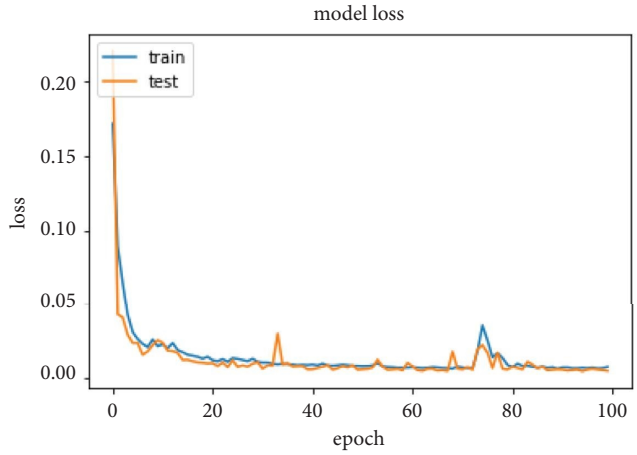

Figure 12: Model loss of IoTID20.

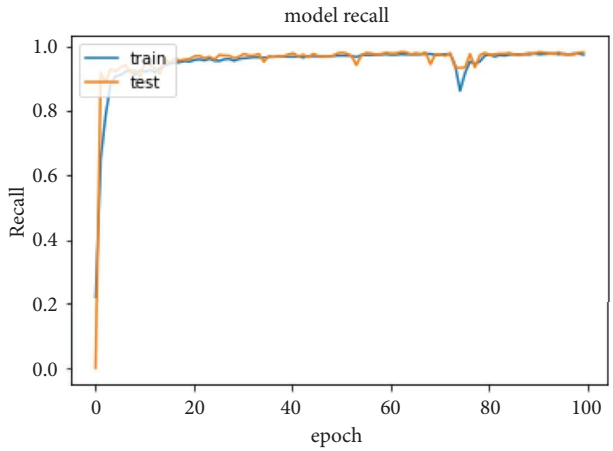

Figure 13: Model recall of IoTID20.

GRU, CNN + LSTM, and HEM, respectively. The result is shown in Figure 28 when there are 9 fog nodes in the NW and 100 smart devices total. According to this graph, DNMLP has a faster time to detect CB than LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM. The average CBDT of 100 smart gadgets is determined to be 0.00041 sec, 0.015231 sec,

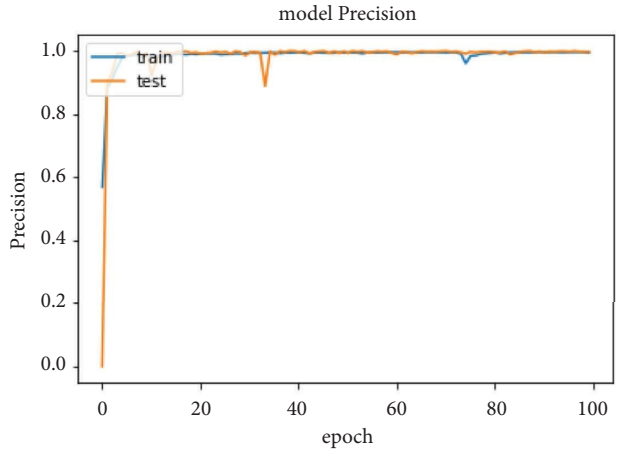

Figure 14: Model precision of IoTID20.

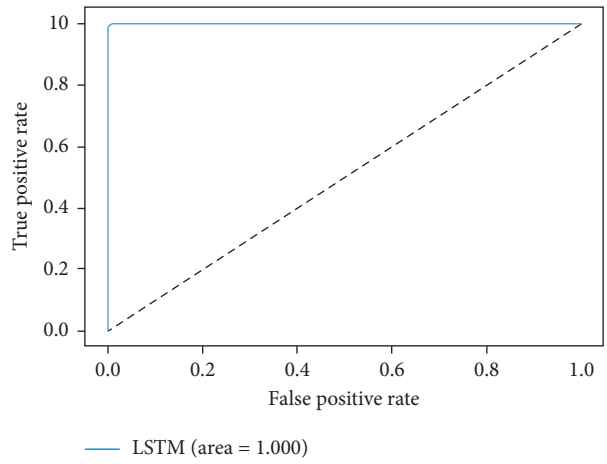

Figure 15: DDOS-SDN_LSTM_ROC-AUC.

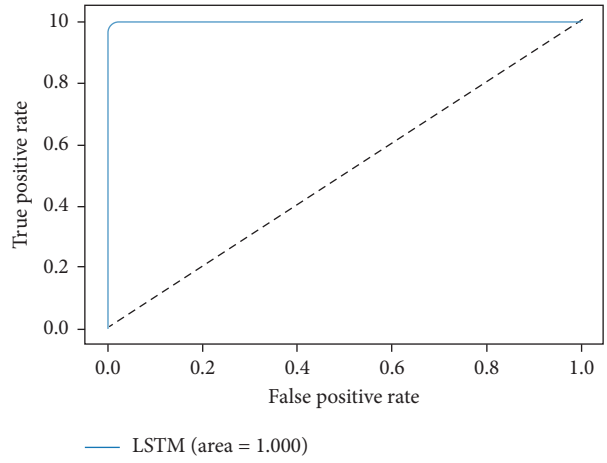

Figure 16: NSL-KDD_LSTM_ROC-AUC.

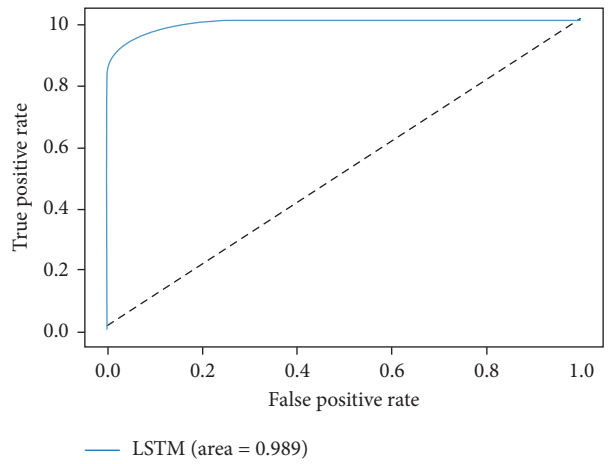

Figure 17: UNSW-NB15_LSTM_ROC-AUC.

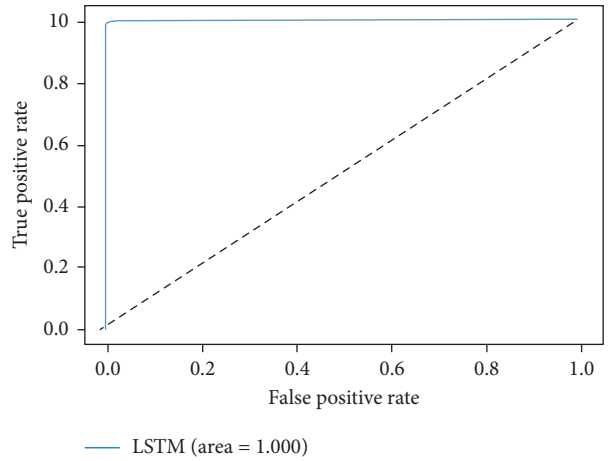

Figure 18: IoTID20_LSTM_ROC-AUC.

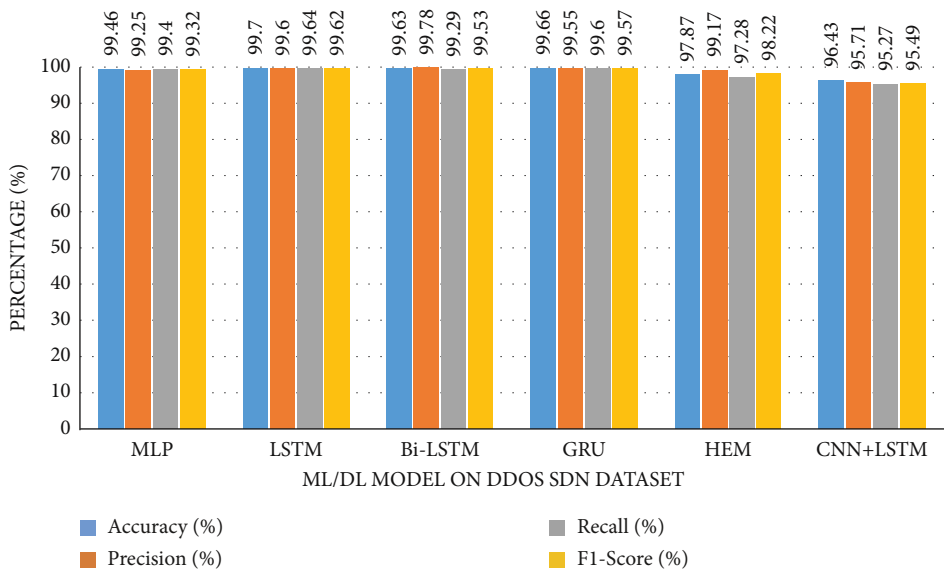

Figure 19: Metrics of DDoS-SDN dataset.

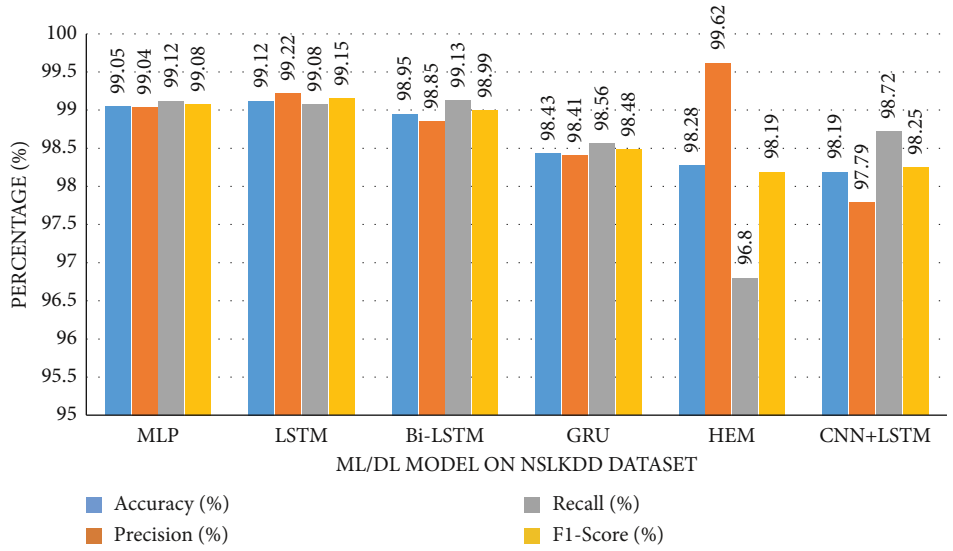

Figure 20: Metrics of NSL-KDD dataset.

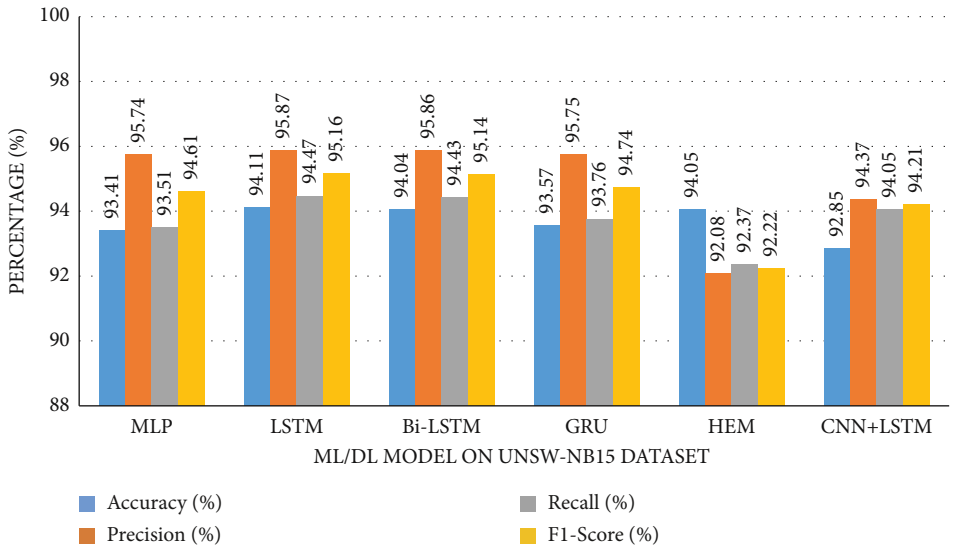

Figure 21: Metrics of UNSW-NB15 dataset.

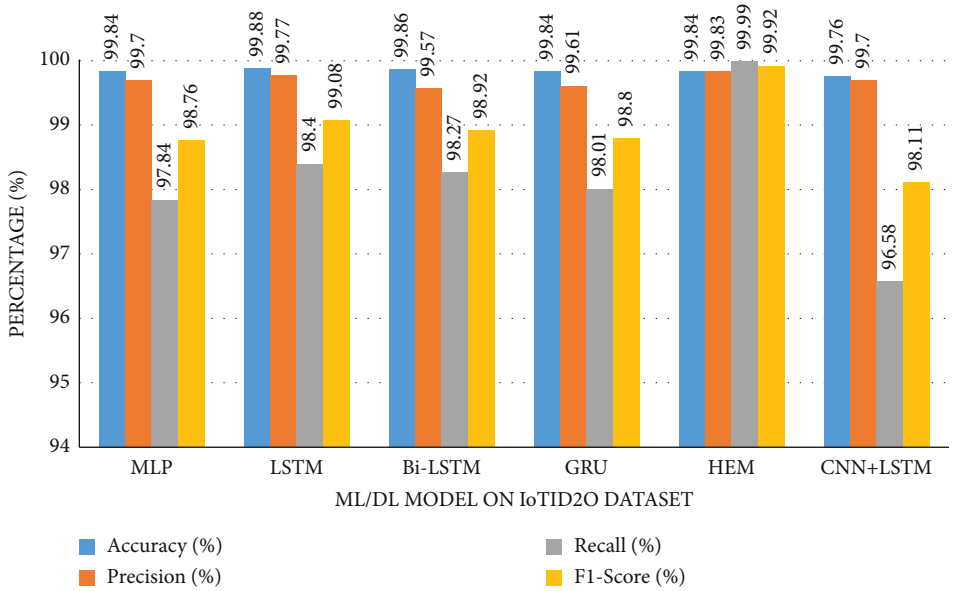

Figure 22: Metrics of IoTID20 dataset.

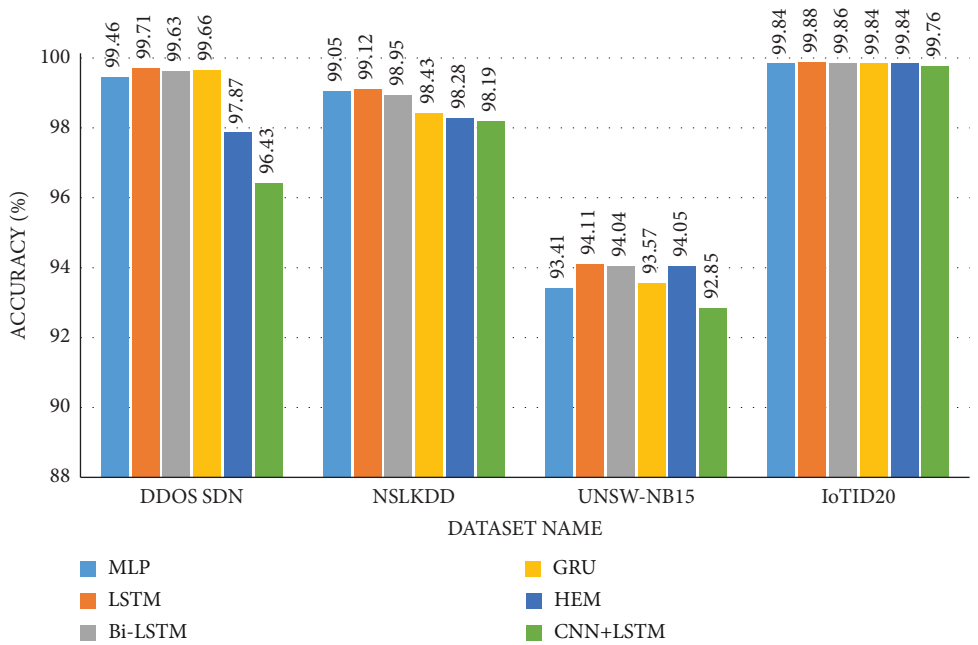

Figure 23: Accuracies of all DLMs on considered datasets.

Table 3: Simulation of IoT device communication behaviour at the fog layer.

Sl. no.	Variable	Value
i	Cloud server	1
ii	Fognode count	1–10
iii	Connected device count	10–100
iv	Dataset used	IoTID20
v	Average CBDT of DNMLP for 10 specimens	0.0000672 sec
vi	Average CBDT of LSTM for 10 specimens	0.0024924 sec
vii	Average CBDT of Bi-LSTM for 10 specimens	0.004164 sec
viii	Average CBDT of GRU for 10 specimens	0.0021 sec
ix	Average CBDT of CNN + LSTM for 10 specimens	0.000476 sec
x	Average CBDT of HEM for 10 specimens	0.008 sec
xi	Count of simulations run	10

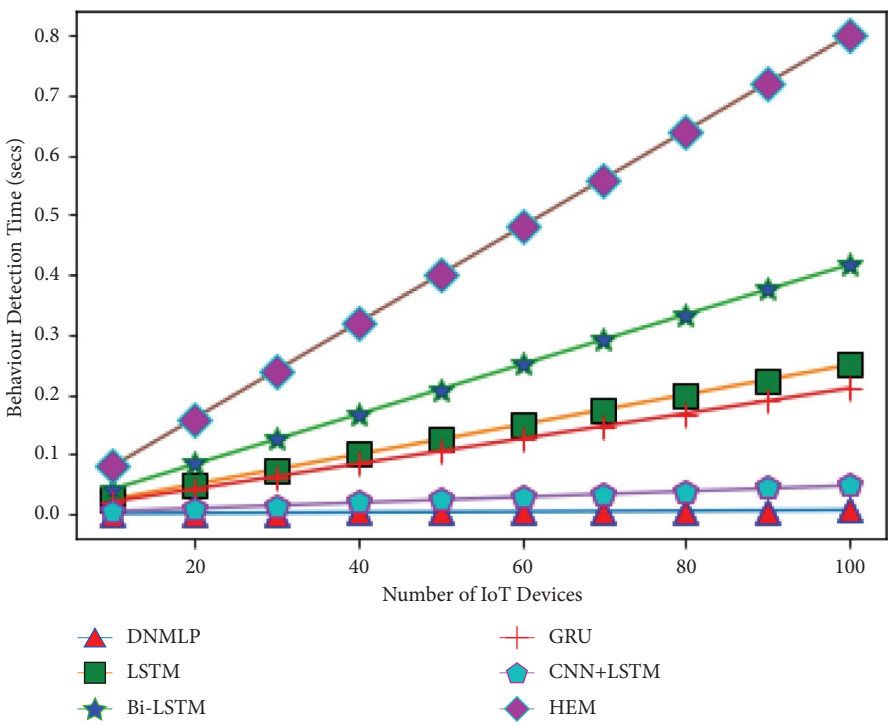

Figure 24: Examination of CBDT for different models having 1 fog node in the NW.

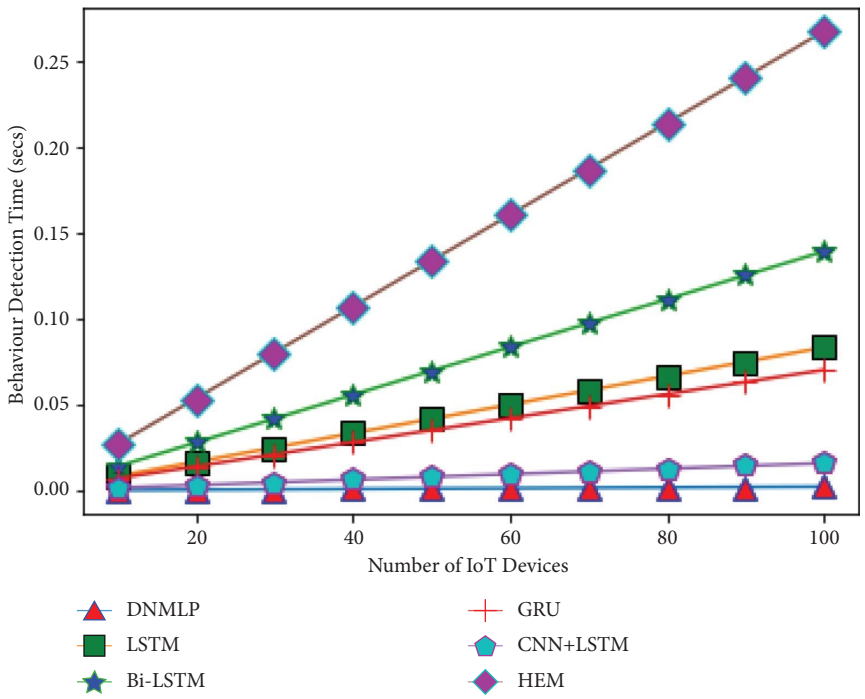

Figure 25: Examination of CBDT for different models having 3 fog nodes in the NW.

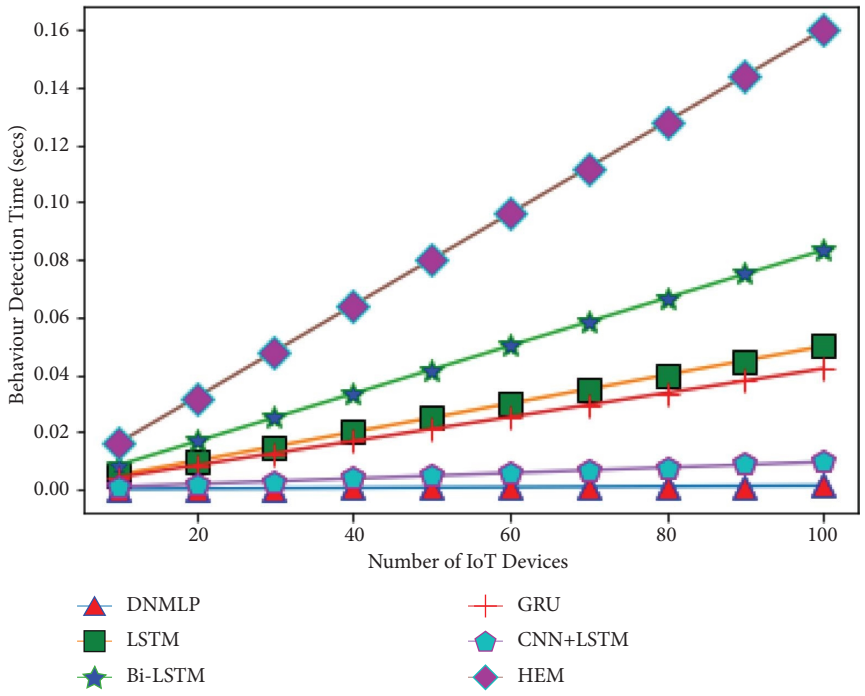

Figure 26: Examination of CBDT for different models having 5 fog nodes in the NW.

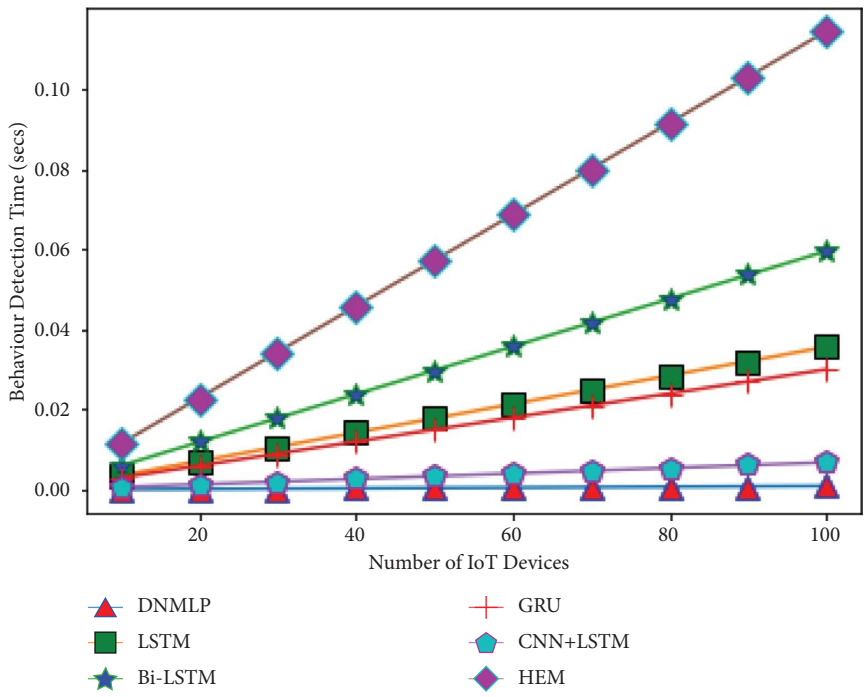

Figure 27: Examination of CBDT for different models having 7 fog nodes in the NW.

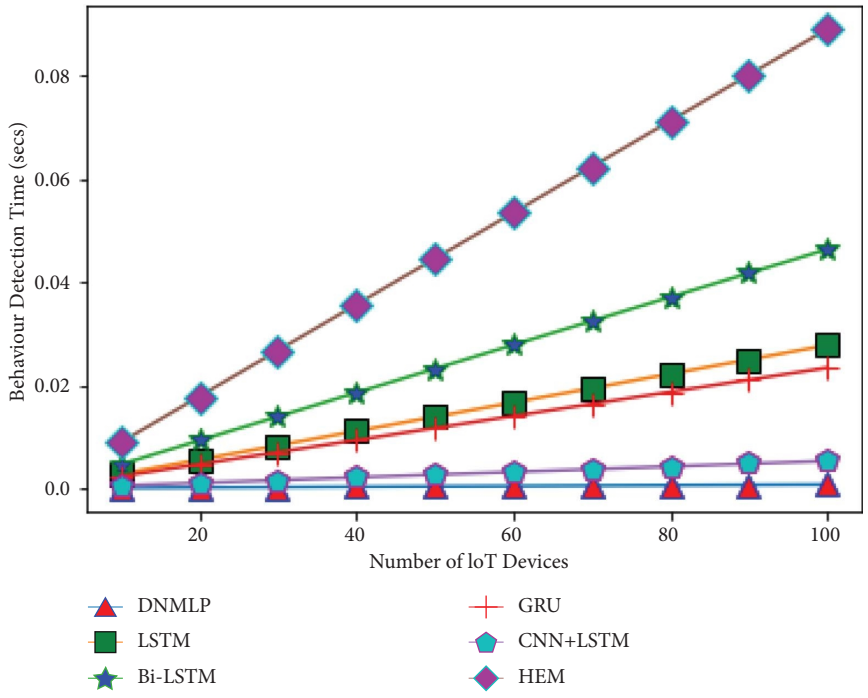

Figure 28: Examination of CBDT for different models having 9 fog nodes in the NW.

0.025446 sec, 0.012833 sec, 0.002908 sec, and 0.048888 sec for DNMLP, LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM, respectively.

From the aforementioned findings, it is also shown that communication behaviour detection time is reduced as the number of fog nodes in the network increases.

7. Conclusion

This study proposes a DL model-based assault prognostication system for fog-based Internet of Things environment. The network consists of a smart sensing tier, a secure fog tier, and a cloud tier. Following this, a variety of deep learning (DL) models, including DNMLP, LSTM, Bi-LSTM, GRU, CNN + LSTM, and HEM, are assessed to prognosticate the most accurate model with high exactness for installation at the fog nodes. The DDOS SDN, NSL-KDD, UNSW-NB15, and IoTID20 datasets indicate 99.70%, 99.12%, 94.11%, and 99.88% accuracy using the LSTMDL model, respectively. As a result, the fog tier in the NW is installed using LSTM, with every fog node being equipped with the LSTMDL model. The deployed model performs binary classing as two classes, 1 and 0, as the assailant or benign separately and sends the device CB to the cloud for refurbishing. The cloud then sends the misbehaviour data to the fog nodes, each of which is aware of the local attack situation in the fog layer. The individual fog nodes decide whether to communicate with these attacking devices in the future by evaluating their current behaviour. In a fog-based Internet of Things condition, the proffered model will be a finer stratagem against the attacks from securing the fog layer, which conquers the stratagem of deploying the DLMs in the sensing layer. The results of the proposed framework prove that the considered DLMs can be acquired for cybersecurity to identify cyberattacks that prevailed in distinct datasets. Additionally, network simulation is used to demonstrate how well various DL models portray the CBDT in the fog layer. The LSTMDL model outperforms DNMLP in terms of accurately forecasting the attacks, although it takes longer to identify the activity (CBDT) than other models, according to this study. Additionally, it has been discovered that the CBDT decreases as the fog nodes in the NW grows. We will implement a similar strategy for attack forecasting in the future using multiclass classing. Similar to how specific attacks can be discovered by using more recent datasets, new multilayer deep neural network models such as AlexNet, ResNet, VGGNet, DenseNet, and Shufflenet, can be created by prepping the fog nodes with a larger dataset.

Data Availability

Data are available upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

The authors thank Parala Maharaja Engineering College (Govt.), Berhampur, India, and BPUT Rourkela (Govt.), India, for providing adequate facility and infrastructure for conducting this research work.

References

[1] A. Samy, H. Yu, and H. Zhang, "Fog-based attack detection framework for internet of things using deep learning," *IEEE Access*, vol. 8, pp. 74571–74585, 2020.

[2] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "An anomaly mitigation framework for iot using fog computing," *Electronics*, vol. 9, no. 10, p. 1565, 2020.

[3] D. Puthal, S. P. Mohanty, S. A. Bhavake, G. Morgan, and R. Ranjan, "Fog computing security challenges and future directions [energy and security]," *IEEE Consumer Electronics Magazine*, vol. 8, no. 3, pp. 92–96, 2019.

[4] S. B. Nath, H. Gupta, S. Chakraborty, and S. K. Ghosh, "A survey of fog computing and communication: current researches and future directions," 2018, https://arxiv.org/abs/1804.04365.

[5] J. Granjal, E. Monteiro, and J. Sa Silva, "Security for the internet of things: a survey of existing protocols and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[6] A. Tewari and B. B. Gupta, "Security, privacy and trust of different layers in internet-of-things (iots) framework," *Future Generation Computer Systems*, vol. 108, pp. 909–920, 2020.

[7] S. K. Bhoi, P. K. Sahu, M. Singh, P. M. Khilar, R. R. Sahoo, and R. R. Swain, "Local traffic aware unicast routing scheme for connected car system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2360–2375, 2020.

[8] S. P. K. Gudla, S. K. Bhoi, S. R. Nayak, A. Verma, and Di-ads, "DI-ADS: a deep intelligent distributed denial of service attack detection scheme for fog-based IoT applications," *Mathematical Problems in Engineering*, vol. 2022, Article ID 3747302, 17 pages, 2022.

[9] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: a classification," in *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology*, pp. 190–193, IEEE Cat. No. 03EX795, Darmstadt, Germany, December, 2003.

[10] B. Sudqi Khater, A. W. B. Abdul Wahab, M. Y. I. B. Idris, M. Abdulla Hussain, and A. Ahmed Ibrahim, "A lightweight perceptron-based intrusion detection system for fog computing," *Applied Sciences*, vol. 9, no. 1, p. 178, 2019.

[11] D. Chaudhary, K. Bhushan, and B. B. Gupta, "Survey on ddos attacks and defense mechanisms in cloud and fog computing," *International Journal of E-Services and Mobile Applications*, vol. 10, no. 3, pp. 61–83, 2018.

[12] S. Potluri, M. Mangla, S. Satpathy, and S. N. Mohanty, "Detection and prevention mechanisms for ddos attack in cloud computing environment," in *Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6, IEEE, Kharagpur, India, July, 2020.

[13] S. Azizpour and M. Majma, "Nada: new architecture for detecting dos and ddos attacks in fog computing," *Journal of Computer Virology and Hacking Techniques*, pp. 1–14, 2022.

[14] S. Li, Y. Lu, and J. Li, "Cad-ids: a cooperative adaptive distributed intrusion detection system with fog computing," in

Proceedings of the 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 635–640, IEEE, Hangzhou, China, May, 2022.

[15] Z. Abou El Houda and L. Khoukhi, "A hierarchical fog computing framework for network attack detection in sdn," in *Proceedings of the ICC 2022-IEEE International Conference on Communications*, pp. 4366–4371, IEEE, Seoul, South Korea, May, 2022.

[16] N. Ahuja, G. Singal, and D. Mukhopadhyay, *Ddos Attack Sdn Dataset*, Mendeley Data, 2020.

[17] Kiran, "Dataset for Intrusion Detection System," 2020, https://www.kaggle.com/datasets/kiranmahesh/nslkdd.

[18] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, IEEE, Canberra, Australia, November, 2015.

[19] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks," in *Canadian Conference on Artificial Intelligence*, pp. 508–520, Springer, Berlin/Heidelberg, Germany, 2020.

[20] K. Bhushan, "Ddos attack defense framework for cloud using fog computing," in *Proceedings of the 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pp. 534–538, IEEE, Karnataka, India, May, 2017.

[21] R. Priyadarshini and R. K. Barik, "A deep learning based intelligent framework to mitigate ddos attack in fog environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 3, pp. 825–831, 2019.

[22] K. Kalaivani and M. Chinnadurai, "A hybrid deep learning intrusion detection model for fog computing environment," *Intelligent Automation & Soft Computing*, vol. 29, no. 3, pp. 1–15, 2021.

[23] A. Churcher, R. Ullah, J. Ahmad et al., "An experimental analysis of attack classification using machine learning in iot networks," *Sensors*, vol. 21, no. 2, p. 446, 2021.

[24] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: datasets and comparative study," *Computer Networks*, vol. 188, Article ID 107840, 2021.

[25] P. Kumar, G. P. Gupta, and R. Tripathi, "A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, pp. 9555–9572, 2021.

[26] P. Kumar, G. P. Gupta, and R. Tripathi, "Design of anomaly-based intrusion detection system using fog computing for iot network," *Automatic Control and Computer Sciences*, vol. 55, no. 2, pp. 137–147, 2021.

[27] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, S. Garg, and M. M. Hassan, "A distributed intrusion detection system to detect ddos attacks in blockchain-enabled iot network," *Journal of Parallel and Distributed Computing*, vol. 164, pp. 55–68, 2022.

[28] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "A ddos attack mitigation framework for iot networks using fog computing," *Procedia Computer Science*, vol. 182, pp. 13–20, 2021.

[29] S. Askar, "Deep learning and fog computing: a review," *International Journal of Science and Business*, vol. 5, no. 6, pp. 197–208, 2021.

[30] M. A. Ferrag, L. Shu, H. Djallel, and K.-K. R. Choo, "Deep learning-based intrusion detection for distributed denial of service attack in agriculture 4.0," *Electronics*, vol. 10, no. 11, p. 1257, 2021.

[31] Y. Labiod, A. Amara Korba, and N. Ghoualmi, "Fog computing-based intrusion detection architecture to protect iot networks," *Wireless Personal Communications*, vol. 125, no. 1, pp. 231–259, 2022.

[32] T. A. Ahanger, U. Tariq, A. Ibrahim, I. Ullah, Y. Bouteraa, and F. Gebali, "Securing iot-empowered fog computing systems: machine learning perspective," *Mathematics*, vol. 10, no. 8, p. 1298, 2022.

[33] C. A. de Souza, C. B. Westphall, and R. B. Machado, "Two-step ensemble approach for intrusion detection and identification in iot and fog computing environments," *Computers & Electrical Engineering*, vol. 98, Article ID 107694, 2022.

[34] S. Ghosh, V. Chandra, and A. Adhya, "Machine learning for fog computing-based iot networks in smart city environment," in *Intelligent Internet of Things for Healthcare and Industry*, pp. 267–285, Springer, Berlin/Heidelberg, Germany, 2022.

[35] A. Mihoub, O. B. Fredj, O. Cheikhrouhou, A. Derhab, and M. Krichen, "Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques," *Computers & Electrical Engineering*, vol. 98, Article ID 107716, 2022.

[36] S. Sambangi, L. Gondi, and S. Aljawarneh, "A feature similarity machine learning model for ddos attack detection in modern network environments for industry 4.0," *Computers & Electrical Engineering*, vol. 100, Article ID 107955, 2022.

[37] R. V. Deshmukh and K. K. Devadkar, "Understanding ddos attack & its effect in cloud environment," *Procedia Computer Science*, vol. 49, pp. 202–210, 2015.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] R. Dolphin, "LSTM Networks A Detailed Explanation," 2021, https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9.

[40] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," 2018, https://arxiv.org/abs/1801.02143.

[41] K. Cho, B. Van Merriënboer, C. Gulcehre et al., "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014, https://arxiv.org/abs/1406.1078.

[42] J. Donahue, L. Anne Hendricks, S. Guadarrama et al., "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2625–2634, Boston, MA, USA, June, 2015.

[43] D. K. K. Reddy, H. S. Behera, J. Nayak, B. Naik, U. Ghosh, and P. K. Sharma, "Exact greedy algorithm based split finding approach for intrusion detection in fog-enabled iot environment," *Journal of Information Security and Applications*, vol. 60, Article ID 102866, 2021.