Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 3747302, 17 pages
https://doi.org/10.1155/2022/3747302

Research Article

DI-ADS: A Deep Intelligent Distributed Denial of Service Attack Detection Scheme for Fog-Based IoT Applications

Surya Pavan Kumar Gudla,[1,2] Sourav Kumar Bhoi,[2] Soumya Ranjan Nayak ,[3] and Amit Verma[4]

[1]*Faculty of Engineering (Computer Science and Engineering), BPUT, Rourkela 769015, Odisha, India*
[2]*Department of Computer Science and Engineering, Parala Maharaja Engineering College (Govt.), Berhampur 761003, Odisha, India*
[3]*Amity School of Engineering and Technology, Amity University, Noida 201301, UP, India*
[4]*Department of Computer Science & Engineering and University Center for Research and Development, Chandigarh University, Mohali 140413, Punjab, India*

Correspondence should be addressed to Soumya Ranjan Nayak; srnayak@amity.edu

Received 15 May 2022; Revised 26 June 2022; Accepted 4 July 2022; Published 8 August 2022

Academic Editor: Rajesh Kaluri

Nowadays, fog computing plays a very vital role in providing many services to end-based IoT (Internet of Things) systems. The end IoT devices communicate with the middle layer fog nodes and to the above cloud layer to process the user tasks. However, this large data communication experiences many security challenges as IoT devices are being compromised and thus the fog nodes at the fog layer are more prone to a very critical attack known as Distributed Denial of Service (DDoS) attack. The attackers or the compromised IoT devices need to be detected well in the network. Deep Learning (DL) plays a prominent role in predicting the end-user behavior by extracting features and classifying the adversary in the network. But, due to IoT device's constrained nature in computation and storage facilities, DL cannot be administered on those. In this paper, a deep intelligent DDoS attack detection scheme (DI-ADS) is proposed for fog-based IoT applications. The framework mainly uses a deep learning model (DLM) to detect DDoS attacks in the network. The DLM is installed on the computation module of the fog node that predicts the end IoT device behavior. For the selection of the best DLM model at the fog layer, the performance comparison is made on Deep Neural Multilayer Perceptron (DNMLP) and Long Short-Term Memory (LSTM) models along with the conventional machine learning (ML) models such as Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Logistic Regression (LR), and Random Forest (RF). The simulation is performed using the Python Anaconda platform by considering a new DDoS-SDN (Mendeley Dataset) dataset that consists of three DDoS attacks such as TCP Syn, UDP Flood, and ICMP attacks. From the results, DNMLP showed the best accuracy of 99.44% as compared to other DL and ML models. By outperforming nature in the detection of DDoS attacks, DNMLP is considered in the proposed framework for being implemented at the fog layer.

1. Introduction

IoT is a profoundly arising stage in the present progressing world. It is planned with different connected devices such as smart vehicles, smart healthcare, smart grid, surveillance, etc. Its prominence of monstrous use brought about many assaults on connected devices in the trading of resources. Because of its centralized nature, more openness to hazards like unpathed weaknesses, weak authentication, and weak APIs may happen. Information on IoT devices is controlled by sensors and actuators over the cloud which gives on-request administration to the end clients. Nonetheless, offering types of assistance to IoT by the cloud has its issues in the method of high latency, data security, and protection and interruption [1–6].

To mitigate the issues of IoT with the cloud, another layer is developed called fog, the arising innovation that plays between the cloud and end devices. It disseminated a decentralized computing model which helps in offering types of assistance to the end clients by latency and bandwidth utilization and makes information nearer to the edge of the network. Still on, fog devices are profoundly inclined to attacks that get compromised with data privacy. The most occurring of attacks are DDoS, IP Spoofing, Man-in-Middle attack, and port scan attack [7]. To emerge from this, many attack detection mechanisms are required in fog architecture. On providing benefits to the end-users such as low latency and mobility an intermediate layer is built between IoT and cloud [2]. To bring out communication among the connected devices, IoT devices require communication protocols. For instance, consider a surveillance system where huge data will be collected and stored in the cloud for future retrieval and IT services, but we require only summarized data in the cloud to prevent the wastage of bandwidth and storage capacity. To achieve low latency, effective utilization of bandwidth, and storage we require non-cellular-based network protocols for communication (COAP, MQTT, LoRa, and LoRaWAN) [5, 8–10].

IoT devices generated data is communicated and stored near to devices employing fog using above protocols for fast accessing of resources. In this case, there is more chance of happening attacks on fog nodes/fog layer. So, we need to define attack detection in the fog layer to create a secure system. Due to the increase in usage of the Internet and a large amount of data transfer, a greater number of anomalies may be present. In parallel, the cause of attacks is also increasing consistently. Many organizations are continually working on network attack detection to provide secure services to the end-users. The usage of cloud services and IoT over the fog layer also leads to more increased risk of data violation. In this regard to provide or design a more secured system over legacy methods using high-end DL algorithms we can identify the attacks dynamically. With the ever-increasing Internet, society is moving towards modern technologies to predict, detect, or classify, and analysis of network behavior using DL approaches is being widely used. Hence attack detection is becoming the most recent trend and research scope for cyber threats.

Fog layer security has become challenging because of its geo-distribution and location awareness. Initially to detect attacks machine learning (ML) techniques are highly used but unsuitable for a huge volume of data. To overcome the limitation of ML, DL is used in detecting attacks in the fog layer. DL is preferable over ML for huge data as it has multiple layers in processing. DL has been utilized in classifying many attacks with a high detection rate and results in the binary classification of normal and abnormal behavior and multilabel classification and sends it to cloud for behavior update of a node [1, 4, 7, 11, 12]. Due to resource constraint nature of IoT, it is not possible to implement complex DL algorithms; therefore DL is suitable to implement on fog node/fog layer with high accuracy. Hence, DL is the best over ML algorithms against a huge volume of data. In this work, a DDoS attack is detected based on fog-based IoT applications using DLM.

The proposed work is designed with a deep intelligence DDoS attack scheme (DI-ADS) based on fog and DL. By this framework, the DDoS attack is detected in less span with low latency compared to attack detection based on the cloud. To define the best model for attack detection we tested the potentiality of two DLMs and four ML models at the computational module of the fog node considering the DDoS-SDN dataset. The dataset involves three different types of DDoS attacks, namely, TCP Syn, UDP flood, and ICMP attacks. From accuracy, it is proved that the DNMLP model is the best model for attack detection and thus installed at the fog node. The communication between end IoT devices and fog is routed by gateways on which fog nodes detect attacks by classifying IoT data. The framework is implemented in six stages. The network setup is implemented in the first stage as three layers (IoT device layer, fog layer, and cloud layer) through interfaces. The network traffic classification setup aims at selecting the best model as the second stage. In the third stage, model deployment and network initialization are done by deploying the selected model on the fog node for attack detection. The fourth stage specifies how the fog node classifies the behavior of IoT devices and updates to the cloud. The cloud update is done in the fifth stage by updating the current behavior on received information from the fog. The sixth stage shows the local table update at fog for future communication and attack detection.

The major contribution of this work is stated as follows:

(1) In this paper, a deep intelligent DDoS attack detection scheme (DI-ADS) is proposed for fog-based IoT applications. The framework mainly uses DLM to detect DDoS attacks in the network.

(2) The DLM is mainly installed at the computation module of the fog node that predicts the end IoT device behavior. For the selection of the best DLM model at the fog layer, the performance comparison is made between DNMLP and LSTM and also conventional ML models such as SVM, k-NN, LR, and RF.

(3) The simulations are performed using the Python Anaconda platform by considering a new DDoS-SDN Mendeley dataset [13] that consists of three DDoS attacks such as TCP Syn, UDP Flood, and ICMP attacks.

(4) From the result, it is found that the DNMLP model shows better prediction accuracy of 99.44% as compared to other models and it is implemented at fog nodes for detection of DDoS attacks.

The rest of the sections are discussed as follows. Section 2 presents the related works. Section 3 presents the system model. Section 4 presents the problem statement. Section 5 presents the proposed DI-ADS model. Section 6 presents the simulations and results. Section 7 presents the conclusion and future scope.

2. Related Works

In this section, many research works are discussed related to this concept and they come up with best Deep Learning

Techniques in proposing attack detection framework for fog-based IoT system. Ahmed et al. [1] using DL technique proposed attack detection framework for several cyber-attacks which results in high detection rate with 99.96% detection accuracy (DA) in binary classification and 99.65% DA in multiclassification. Lawal et al. [2] designed two modules' framework for oddity detection using signature-based and anomaly-based methods. Module-1 is six times faster than module-2. XGBoost Classifier is used for binary and multiclass classification by module-2 to obtain accuracy of 99% and 97% for average recall, precision, and $F1$ score, respectively. Puthal et al. [3] discussed a 3-layered architecture and possibilities of threats and unfolding from threats at each layer and also discussed advanced research issues required for present architecture. Khater et al. [8] presented a lightweight IDS using MLP model on vector space representation. Latency on cloud, mobility support, and location awareness problems on cloud are addressed using ADFA-LD and ADFA-WD datasets which resulted in 94%, 95%, and 92% of accuracy, recall, and $F1$-measure, respectively. On ADFA-LD dataset, 74% of accuracy, recall, and $F1$-measure is found, respectively, using Raspberry Pi. Bhushan and Deepali [14] proposed a framework for defense against DDoS attack by generating TCP traffic using LOIC on Kali Linux Machine. Fog defender is used and applying rules at fog layer allowed only legitimate requests to cloud for accessing. Priyadarshini and Barik [15] proposed a novel source based DDoS defense mechanism to mitigate DDoS attacks using DLMs by blocking infected packets disseminated to cloud on CTU-13 Botnet and ISCX 2012 IDS dataset. The training and testing dataset taken is in ratio 90 : 10 with 10-fold cross validation scheme that resulted in 98.88% accuracy. Chaudhary et al. [11] surveyed various systems and explored existing things based on security, privacy, limitation, and challenges and open directions of research in the domain of computing. Douligeris and Mitrokotsa [7] presented the classification of DDoS defense system, pros and cons, and effective defense mechanism and techniques for better understanding. Potluri et al. [12] discussed DDoS attack, its detection, and prevention mechanism in cloud computing environment by using various approaches like ML, DL, NN, blockchain, SDN, and genetic algorithms. Kalaivani and Chinnadurai [16] proposed an intrusion classification model using CNN and LSTM to predict attacks accurately using DL models. The dataset used for this purpose is NSL-KDD, and it obtains 96.5% accuracy on attack detection using integrated CNN with LSTM FCID model. The model is deployed in fog layer which monitors network traffic, and it protects from malicious users on providing services to the IoT devices by cloud. It is used in multiclass attack classification such as DoS, U2R, R2L, and probe attacks. ICNN-FCID model is provided with different activation function such as ReLU, Sigmoid, and Hyperbolic Tangent (Tanh) activation function of which ReLU provided high accuracy compared to two other functions on ICNN-FCID model. Churcher et al. [17] used several ML algorithms such as KNN, SVM, DT, NB, RF, ANN, and LR for comparing both binary and multilabel classification. Considering several parameters such as accuracy, precision, recall, $F1$-score, and log loss, the above algorithms are compared, RF accuracy of which is 99% for HTTP DDoS attack. But, based on simulation results on abovementioned parameters RF outperforms in binary classification, and for multilabel classification KNN outperforms with accuracy of 99% compared to RF. Kilincer et al. [18] used CSE-CIC IDS-2018, UNSW-NB15, ISCX-2012, NSL-KDD, and CIDDS-001 datasets on 3 different ML algorithms such as SVM, KNN, and DT for classification of attacks by performing min-max normalization on dataset for comparative study. The result of this study shows that DT classifier is more successful that the other two classifiers. Many such related research works can also be found in [19–21, 21–30].

From the literature survey done on various attack detection systems, the following research gaps are found; for example: (i) Most of the papers evaluated the performance accuracy on a small dataset with a smaller number of attributes that do not provide actual information on attack detection. Hence, we used DDoS-SDN dataset [13] in the present study with a greater number of attributes for better classification of attacks. (ii) The second flaw observed is the usage of classifiers for the classification of attacks made on only one classifier model. It is difficult to express which ML method performs better on the selected dataset at the highest level. Also, conventional ML algorithms are used for attack detection which is difficult when larger datasets are used. In this work, we used DLMs for performance evaluation. (iii) Many studies reported a limited number of attacks based on the dataset used. Here, we considered a dataset that can examine different DDoS attack types.

3. System Model

In this section, a system model is discussed with both the network model and attack model. The network model describes the network components, network topology, and communication between the network components. The attack model describes how the attackers attack the network. The notations used in this work are shown in Table 1.

3.1. Network Model. The network model mainly consists of a 3-layered architecture such as the cloud as the upper layer, fog as the middle layer, and IoT or smart devices as the lower layer [1–3, 17, 18] as shown in Figure 1.

The upper layer is called cloud layer which consists of a cloud node C which provides centralized data storage that stores the updated behaviors of the IoT devices. The cloud C also updates the IoT devices at the lower layer at regular intervals for updating the behaviors (normal/DDoS attacker) of the IoT devices. The cloud layer is connected to the middle layer through a gateway (GT) and base stations (BSs). The communication takes place using wired/wireless communications.

The middle layer is called the fog layer that consists of fog nodes $\{FN_1, FN_2, \ldots, FN_n\}$ to accomplish a considerable quantity of storage, computation, and local communication. These nodes give services to the IoT devices in proximity.

Table 1: Notations and descriptions.

Sl. No.	Notation	Description
1	C	Cloud
2	GT	Gateway
3	BS	Base station
4	FN	Fog node
5	CM_{FN}	Computation module of fog node
6	MM_{FN}	Memory module of fog node
7	iot	IoT device
8	A	Attacker
9	I	Set of IoT devices
10	B	Set of behavior instances
11	b	Behavior instance
12	T	Set of time instances
13	t	Time instance
14	Acc	Accuracy
15	DT	Dataset for training and testing
16	tr	Training data
17	ts	Testing data
18	TST	Total service time
19	Target	Label assigned to final behavior
20	BDT	Behavior detection time
21	TUD	Time to update IoT devices
22	CA	Classification accuracy

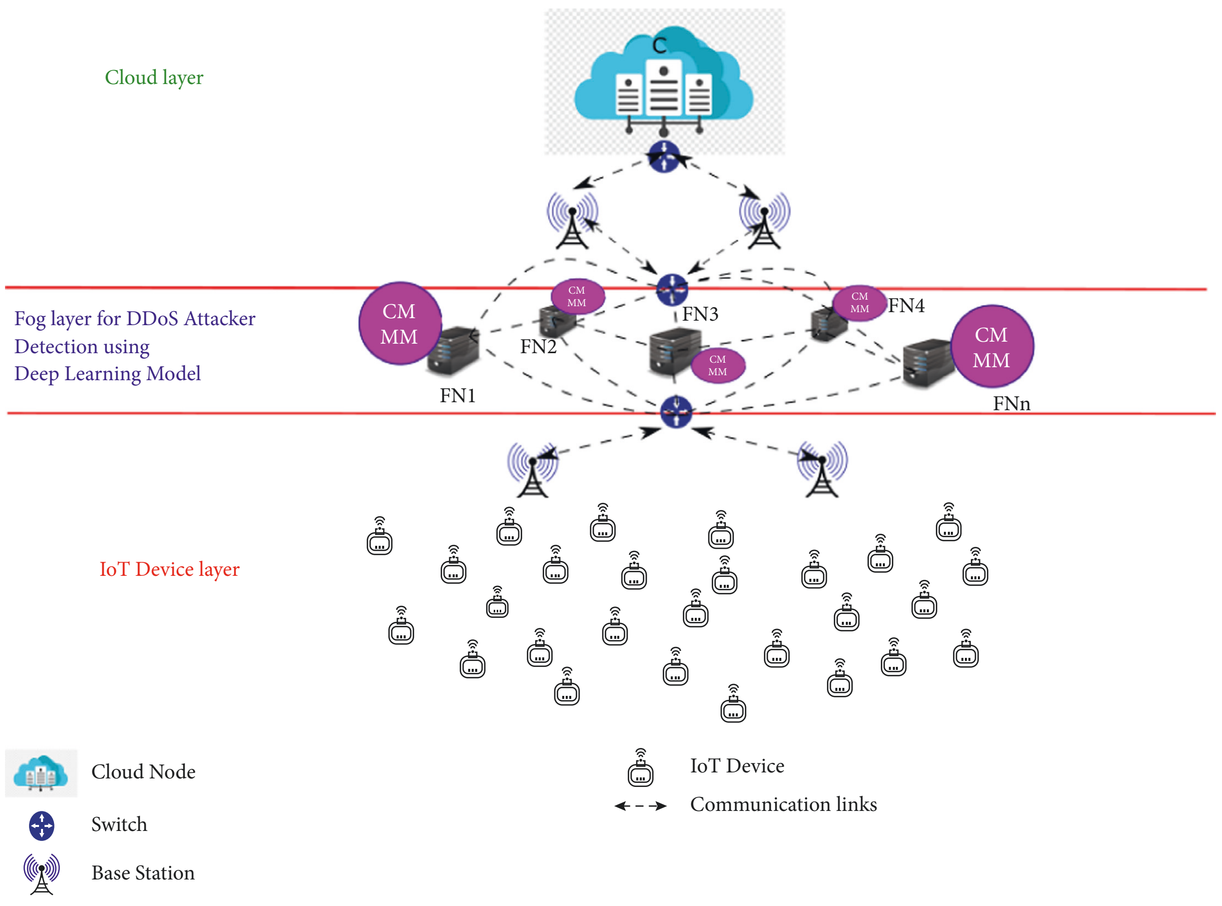

Figure 1: System architecture.

They also record the behavior of the devices in a timely manner. The fog node mainly consists of a computation module (CM_{FN}) and a memory module (E). The CM_{FN} of a fog node is enabled where the CM_{FN} is trained with a DLM to perform a task to predict the behaviors of the IoT devices which communicate with the fog node in proximity. The fog nodes are also connected to each other through wired/wireless communications for data communication between them. The fog layer is connected to the upper layer through GT and BSs. The communication takes place using wired/wireless communications. The fog layer is also connected to the lower layer using GT and BSs through which communication takes place.

The lower layer is called the IoT device layer that comprises IoT devices $\{\text{iot}_1, \text{iot}_2, \ldots, \text{iot}_n\}$ which carries out a large volume of end-users data or requests to fog or cloud for fast computation and service. The IoT devices communicate using BSs and GT for communication with the fog layer or cloud layer. They have limited storage and computational capability.

3.2. DDoS Attack Model. In this section, the attack model is discussed, where the attacker A_i attacks the network components in any format for its control over the network or disruption of the services provided by the network components. In this model, we have assumed that the IoT devices at the lower layer communicate with the fog nodes for getting services from the cloud or fog nodes in proximity.

From Figure 2 it is seen that a DDoS attacker A controls the IoT devices in the network by taking control of the IoT devices by hacking the devices. The IoT devices in the network are now compromised by the attacker. The attacker overwrites the particular attack code over the normal function of the IoT devices by which they behave as a malicious node as per the behavior of the attack. The compromised IoT devices communicate and attack the fog nodes in many ways such as sending unnecessary requests or signals to the fog nodes for jamming the network and for performing any malicious activity or taking control of the network [14, 15, 31]. In this work, we have considered three attacks of DDoS such as TCP Syn, UDP flood, and ICMP attacks by considering a new dataset [13] for training and testing.

4. Problem Statement

As per above model, there are n number of IoT devices $I = \{\text{iot}_1, \text{iot}_2, \ldots, \text{iot}_n\}$ which communicate with a fog node FN with communication behaviors $B = \{b_1, b_2, \ldots, b_n\}$, where b_n is the set of communication instances that is denoted as $b = \{\text{cin}_1, \text{cin}_2, \ldots, \text{cin}_m\}$ at different time instances $T = \{t_1, t_2, \ldots, t_m\}$ where m is the number of communication instances made by an IoT device with a fog node FN. Each communication instance is set of attributes or features $Ft = \left\{ft_1, ft_2, \ldots, ft_p\right\}$ where p is the number of attributes or features such as source IP, destination IP, packets sent, packets received, acknowledgement, etc. with a target label attribute or feature as attack (1) or normal (0). The problem is to predict the communication instance cin_m as attack (1) or normal (0) by implementing a DLM with high accuracy. Before this, the problem is also to train and test different DLMs with standard dataset and select the most appropriate DLM for the fog nodes.

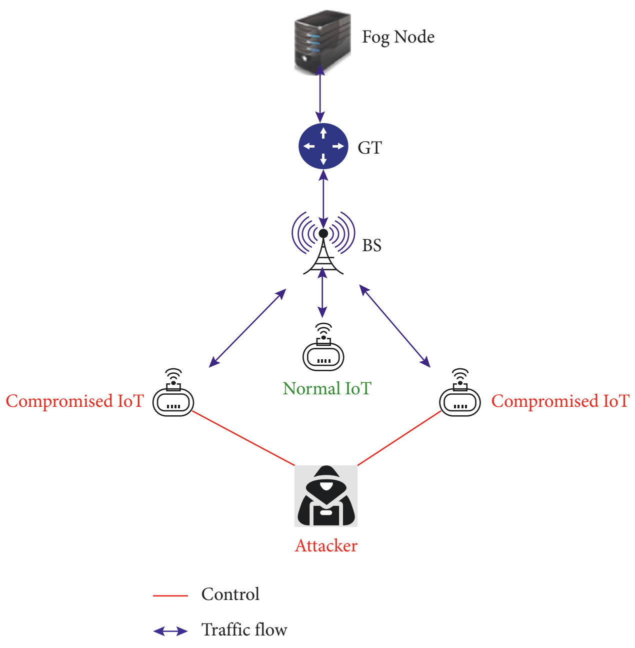

Figure 2: Attacker model.

5. Proposed DI-ADS Scheme

In this section, the proposed DI-ADS scheme is described to solve the above problem. The DI-ADS scheme mainly consists of six steps such as (1) Network Setup, (2) Network Traffic Classification Setup, (3) DLM Deployment and Network Initialization, (4) Attack Detection, (5) Cloud Update, and (6) Fog Node Update. The process flow model of DI-ADS with six steps is presented in Figure 3. The steps of the DI-ADS scheme are described as follows.

5.1. Network Setup. In the first step as shown in Figure 3 of the DI-ADS process flow model, the network is first set and the components of the model are then connected. The cloud node C is first set that provides different services to the users such as computing, storage, platform, networking, etc. As per DI-ADS, the cloud node stores the IoT device's behaviors and also updates them promptly. Then the fog nodes are set in the network in such a manner that the IoT devices can communicate to get services in minimum time. The fog nodes also solve the issues and provide services to the IoT devices in proximity. The fog nodes are connected to each in a wired/wireless manner. The IoT devices at the lower layer are connected to the fog nodes in proximity using BSs and GT. They send and receive data wirelessly using 4 G/LTE/3 G/WiMAX communications. The BSs and GT also send and receive data wirelessly using 4 G/LTE/3 G/WiMAX. The placements of fog nodes and cloud nodes in any region or place are out of the scope of this work. We only focus on the connection of the layers and how communication takes between the network components.

5.2. Network Traffic Classification Setup. After setting the network and connections of each layer, the fog nodes are

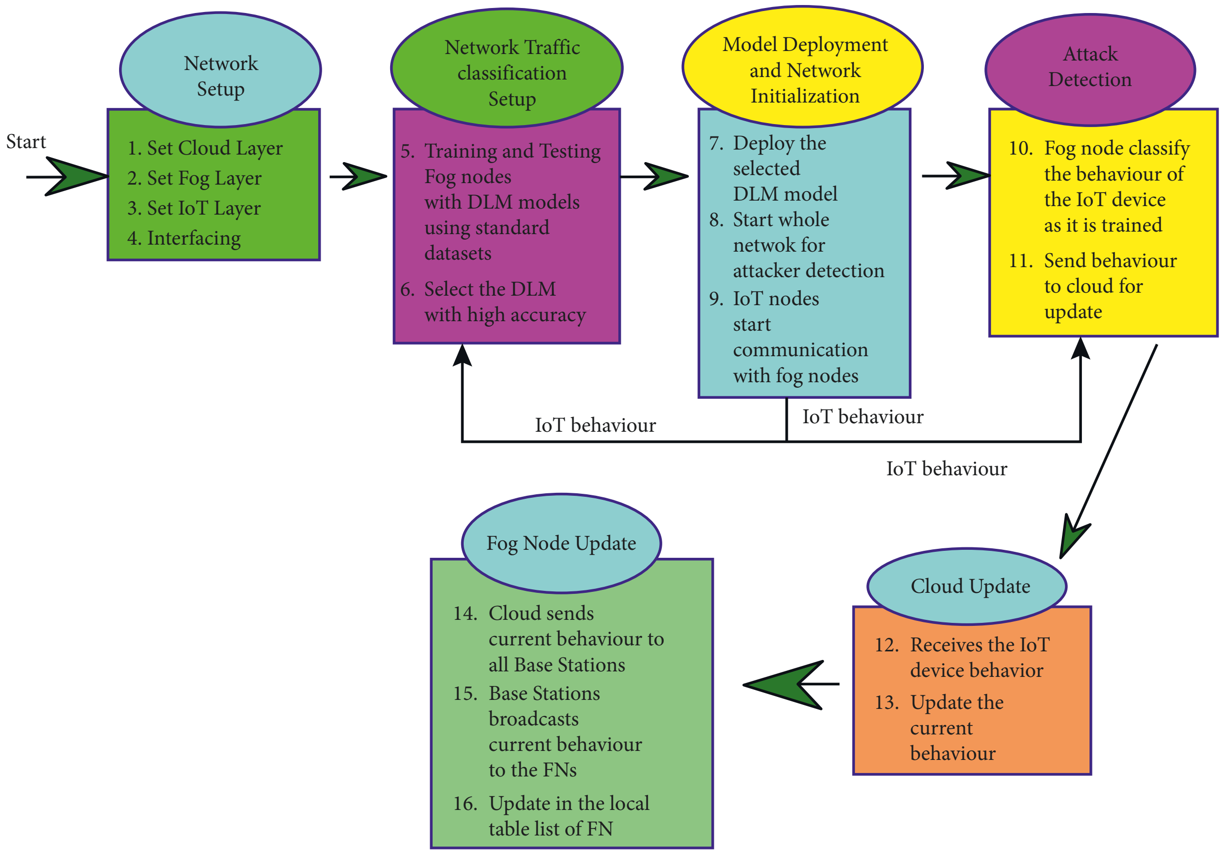
Start
Network Setup
1. Set Cloud Layer
2. Set Fog Layer
3. Set IoT Layer
4. Interfacing
Network Traffic classification Setup
5. Training and Testing Fog nodes with DLM models using standard datasets
6. Select the DLM with high accuracy
Model Deployment and Network Initialization
7. Deploy the selected DLM model
8. Start whole netwok for attacker detection
9. IoT nodes start communication with fog nodes
Attack Detection
10. Fog node classify the behaviour of the IoT device as it is trained
11. Send behaviour to cloud for update
IoT behaviour
IoT behaviour
IoT behaviour
Fog Node Update
14. Cloud sends current behaviour to all Base Stations
15. Base Stations broadcasts current behaviour to the FNs
16. Update in the local table list of FN
Cloud Update
12. Receives the IoT device behavior
13. Update the current behaviour

Figure 3: Process flow model of DI-ADS scheme.

enabled with AI (using DLM) to predict the behavior of the IoT devices in the network which communicate with the fog nodes. The DLM is implemented at the computation module CM_{FN} of the fog node. At the CM_{FN} AI is implemented using DLM. The model is selected based on the highest prediction accuracy. The models are first trained using a standard dataset of DDoS attack detection. The dataset needs to undergo preprocessing steps before training; it goes with preprocessing steps for feature selection using handling missing values, feature scaling, one-hot encoding, and feature selection. The data preprocessing steps are shown in Figure 4 and discussed below as follows.

5.2.1. Data Preprocessing. The dataset used is preprocessed as follows:

(1) Handling Missing Values: Most of the datasets used for classification may have a missing value which creates a problem for the DLM. As per the proposed scheme, we handled the missing values by first removing the rows or columns which have zeros or null values. Afterward, we also look for mean and median methods by replacing the missing values with mean or median. However, it is only used for numerical data.

(2) Feature Scaling: Datasets that have various features with a varying magnitude of values need to be scaled based on the requirement. There are two popular techniques named normalization and standardization. In general, if the data does not follow Gaussian distribution, the normalization technique is used; otherwise standardization technique is used. Normalization is used to alter the numerical values of attributes in a dataset to standardize the scale without changing the differences in value ranges.

(3) One-Hot Encoding: Dataset containing symbolic features cannot be processed by DL/ML model and should be converted into numerical values using one-hot encoding for better prediction. In this encoding scheme, we assign a binary variable for each integer encoded variable for each unique variable.

(4) Feature Selection: The dataset contains several features of which some features have no effect on the classification of attacks that need to be removed from the dataset [16, 32]. Similarly, the features containing zero values can also be removed for better feature selection. Figure 4 shows the data preprocessing, training, and testing that is performed using DLM in the fog node.

5.2.2. Splitting Dataset. After the data preprocessing stage, the dataset was split into a training set and a testing set. The training set is used for training the DLM and the test set is used for testing the model accuracy of prediction. The

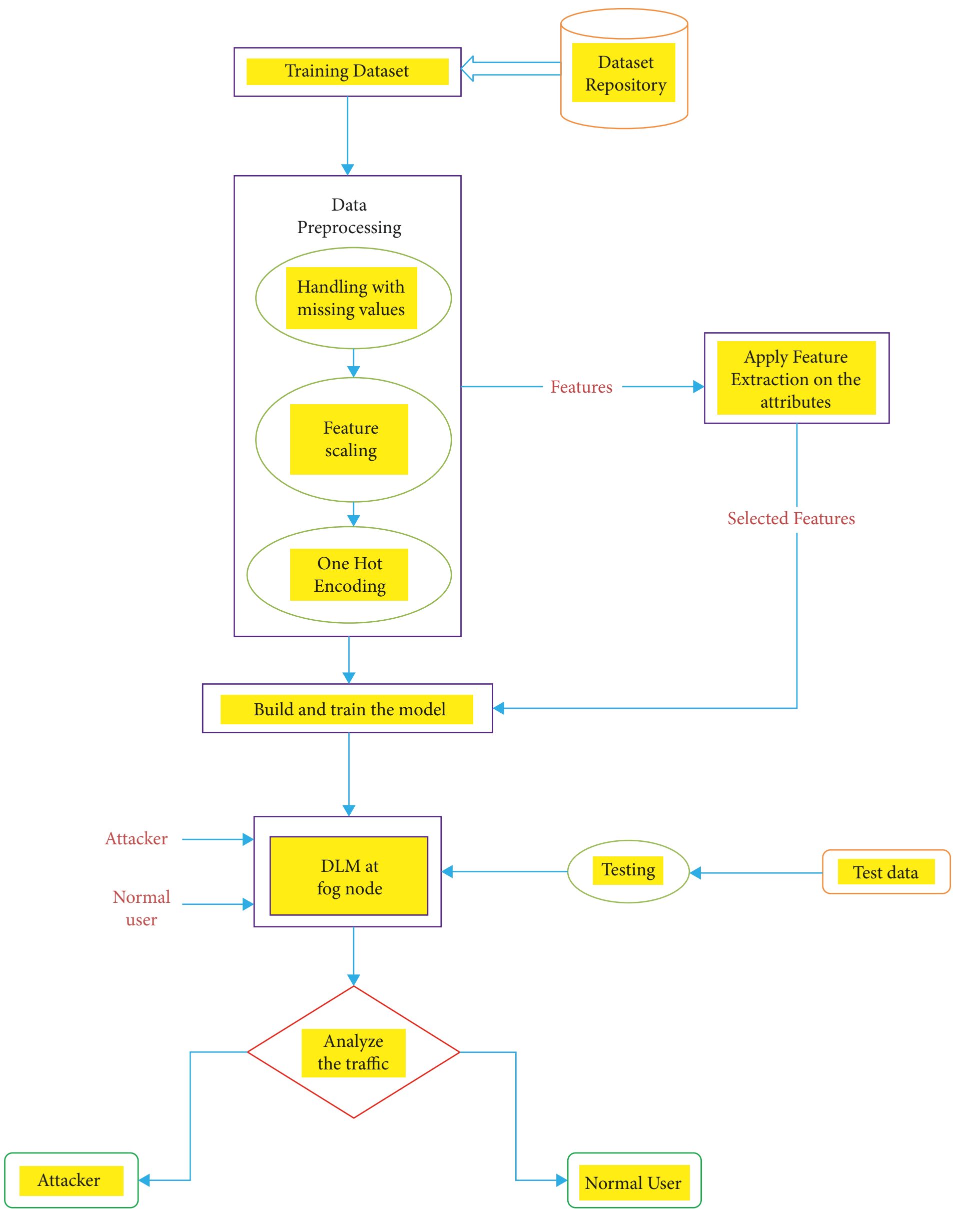

Figure 4: Data preprocessing, training, and testing using DLM at fog nodes.

dataset splitting process of the taken dataset is completely discussed in the performance evaluation section.

5.2.3. DLM Used for Prediction of DDoS Behavior. After splitting the dataset, we have to train the FN_i with this training dataset using DLMs. In this section, we discuss the DLMs used for the prediction of the behavior of IoT devices [1]. The models used are two DL models such as DNMLP and LSTM.

(1) DNMLP: A DNMLP model minimally consists of 3 layers, namely, an input layer, hidden layer, and output layer with an arbitrary number of hidden layers. All the neurons in this layer use a nonlinear activation function excluding the input layer. In DNMLP, data flow in the forward direction to get the data classified, and neurons in DNMLP are also trained with a backpropagation algorithm. In the first step of DNMLP model the input value a_i is multiplied with w_i and summed up.

$$a_i w_i = a_1 w_1 + a_2 w_2 + \cdots + a_n w_n. \tag{1}$$

In the second step, bias b of the hidden layer is added as

$$Z = a_i w_i + b. \tag{2}$$

In the third step, obtained Z value is progressed through the activation function ReLU and Softmax, generally denoted by $\hat{y}$:

$$\hat{y} = \max(0, Z), \tag{3}$$

where if $Z < 0$ the function will output zero and if $Z \geq 0$ the output is simply input. Then, Softmax can be defined as

$$\hat{y} = \frac{e^{Z_i}}{\sum_{j=1}^{n} e^{Z_j}}. \quad (4)$$

In the fourth step loss $(y - \hat{y})^2$ is calculated and if it is higher, it should be minimized by changing w_i and b which can be done by an optimizer and thus cost function is calculated as $\Sigma_n^{i=1} = (y - \hat{y})^2$. Using this backpropagation in a certain number of iterations we arrive at global minima where we can treat this as completion of training of DNMLP. The DNMLP architecture is shown in Figure 5.

(2) LSTM: It is explicitly designed to overcome the problem of Long-Term Dependencies of RNN also called Recurrent Neural Network (RNN) in DL [1, 9, 10]. It is used for classifying and making predictions on data. Every LSTM unit is composed of four things as cell state, input gate, forget gate, and output gate. It is used in language modelling, anomaly detection in network, image captioning, etc. LSTM can retain information for long run and hence used in highly classifying data. Equations involved in the progressive flow of a LSTM cell are

$$\begin{aligned}
f_t &= \text{sigmoid}\left[\left(W_f.h_{t-1}\right) + \left(W_{fq}.q_t\right) + b_f\right],\\
i_t &= \text{sigmoid}\left[\left(W_i.h_{t-1}\right) + \left(W_{iq}.q_t\right) + b_i\right],\\
g_t &= \tanh\left[\left(W_g.h_{t-1}\right) + \left(W_{gq}.q_t\right) + b_g\right],\\
cs_t &= c_t^f + c_t^i,\\
o_t &= \text{sigmoid}\left[\left(W_o.h_{t-1}\right) + \left(W_{oq}.q_t\right) + b_o\right],\\
h_t &= \tanh\left[(cs_t).o_t\right],
\end{aligned} \quad (5)$$

where f_t, i_t, g_t, cs_t, o_t, h_t are forget gate, input gate, input node gate, cell state, output gate, and activation functions. The LSTM architecture is shown in Figure 6.

5.2.4. Used Dataset. The DLMs were evaluated on a standard new dataset to detect the different DDoS attacks and classify the end-user behavior (normal/attacker). The DDoS-SDN new dataset is chosen from Mendeley Data which has 104345 rows with 23 attributes [13]. Dataset is used to detect traffic type as benign or malicious based on TCP Syn attack, UDP flood attack, and the ICMP attack. A total of 23 attributes are available including Switch_id, Packet_count, byte_count, and many so with a total of 1,04,345 rows of data. The traffic classification is labeled as 0 for benign and 1 as a malicious user. The dataset is customized to 18 attributes of which 17 are features and 01 is the target variable. Target label binary is classified with 0 (normal user) and 1 (attacker). The dataset contains one categorical attribute named protocol which is one-hot encoded.

5.3. Deep Learning Model Deployment and Network Initialization. After training the above models with the standard dataset, we need that DLM model which provides the highest prediction accuracy for predicting the behavior of the IoT device (attack or normal) with high probability. We found the accuracy of each model trained and tested and selected the model which has maximum accuracy. The selected model is implemented and deployed at the CM_{FN} on the fog nodes in the fog layer. Afterward, as the model is deployed successfully in the fog layer the network is initialized for real-time processing where the IoT devices start communication with the fog nodes for getting services as per their requirement. Algorithm 1 shows the selection of the best DLM. Algorithm 2 shows the deployment of DLM at the fog layer and network initialization for starting the network.

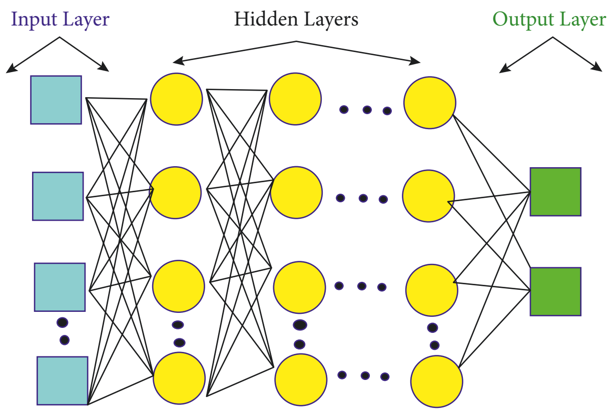

Figure 5: DNMLP architecture.

Theorem 1. *The total service time for an IoT device iot_i is represented as TST_{iot_i}.*

Proof. There is an IoT device iot_i and it has a nearest fog node FN_i at the proximity. Firstly, iot_i sends a request to the FN_i at proximity. The time to send the request is calculated as follows:

$$t_{\text{iot}_i-\text{FN}_i} = t_{\text{iot}_i-\text{BS}} + t_{\text{BS}-\text{GT}} + t_{\text{GT}-\text{FN}_i}, \quad (6)$$

where $t_{\text{iot}_i-\text{BS}}$ is the time to send the request from iot_i to BS, t_{BS} is the time to send the request from BS to the GT, and t_{GT-FN_i} is the time to send the request from GT to FN_i. Then, the time required by FN_i for processing the request is denoted by $t_{\text{processing}_{FN_i}}$ as follows:

$$t_{\text{processing}_{\text{FN}_i}} = t_{\text{queuing}_{\text{FN}_i}} + t_{\text{computation}_{\text{FN}_i}}, \quad (7)$$

where $t_{\text{queuing}_{\text{FN}_i}}$ is the waiting time of the request in the queue and $t_{\text{computation}_{\text{FN}_i}}$ is the time to process the request to find the result. Then the result is transferred to the IoT device iot_i with a time of $t_{\text{FN}_i-\text{iot}_i}$.

$$t_{\text{FN}_i-\text{iot}_i} = t_{\text{FN}_i-\text{GT}} + t_{\text{GT}-\text{BS}} + t_{\text{BS}-\text{iot}_i}, \quad (8)$$

where $t_{\text{FN}_i-\text{GT}}$ is the time to send the result from FN_i to GT, $t_{\text{GT}-\text{BS}}$ is the time to send the result from GT to BS, and $t_{\text{BS}-\text{iot}_i}$ is the time to send the result from BS to iot_i. Therefore the total service time (TST) is calculated as follows:

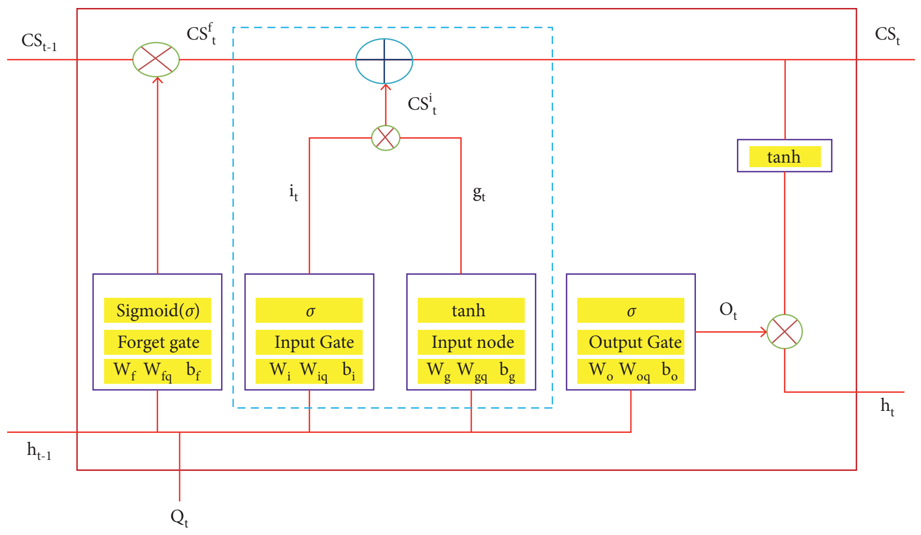

FIGURE 6: LSTM architecture.

```
    Input: $DLM_1, DLM_2, \ldots, DLM_n$, DDoS_Dataset
    Output: DLM_Selected
(1)     DT←Preprocessing (DDoS_Dataset);
(2)     tr, ts←train_test_split (DT));
(3)     for $DLM_1$ to $DLM_n$ do
(4)         Train (tr);
(5)         Test (ts);
(6)         $Acc_n$←Calculate_Accuracy ( ); ▷ Acc: Accuracy
(7)     end for
(8) DLM_Selected←Max ($Acc_1, Acc_2, \ldots, Acc_n$);
```

ALGORITHM 1: Algorithm for selection of best prediction model for the fog layer.

```
     Input: DLM_Selected, $\{FN_1, FN_2, \ldots, FN_n\}$
     Output: Result
(1)      for $FN_1$ to $FN_n$ do
(2)          $CM_{FN_i}$←DLM_Selected; ▷ DLM is installed at each $CM_{FN_i}$ of the fog layer, where $i$ here is the number of fog nodes.
(3)      end for
(4)      network_start ( )
(5)      {
(6)      for $iot_1$ to $iot_m$ do ▷ The network is now ready for real time processing.
(7)          if service needed then
(8)              $iot_i \xrightarrow{\text{Sends Request}} FN_i$;
(9)              $iot_i \xleftarrow{\text{Receives Response}} FN_i$; ▷$FN_i$ in proximity
(10)             Connection_Established ( );
(11)             $FN_i$ process Request;
(12)             $FN_i$ records the $b_i$; ▷ Fog node records the behavior of iot device.
(13)             $FN_i \xrightarrow{\text{Result}} iot_i$; ▷ Sends result to IoT device.
(14)             Connection_Closed ( );
(15)         end if
(16)     end for
(17)     }
```

ALGORITHM 2: Algorithm for DLM model deployment at fog layer and network initialization.

$$\mathrm{TST}_{\mathrm{iot}_i} = t_{\mathrm{iot}_i-\mathrm{FN}_i} + t_{\mathrm{processing}_{\mathrm{FN}_i}} + t_{\mathrm{FN}_i-\mathrm{iot}_i}. \tag{9}$$

□

5.4. Attack Detection. In the previous step, it is described how a DLM model is selected and installed in the fog nodes. Afterward, the IoT devices started communicating with the fog nodes in proximity for getting services. However, after the communication, the DI-ADS scheme predicts the behavior of the nodes from recorded behavior. As the $\mathrm{CM}_{\mathrm{FN}_i}$ of a fog node is enabled with DLM it can predict the behavior of the IoT devices (DDoS attack or normal). After classification, the updated behavior of the node is sent to the cloud C for storage and update. Algorithm 3 shows the classification of IoT device behavior by fog node.

Theorem 2. *The behavior detection time (BDT) of an IoT device iot$_i$ is represented as BDT$_{iot_i}$.*

Proof. There are n number of behaviors for n number of IoT devices in the queue. So, the $\mathrm{BDT}_{\mathrm{iot}_i}$ of an IoT device iot_i of a fog node FN_i is calculated as

$$\mathrm{BDT}_{\mathrm{iot}_i} = t_{\mathrm{queuing}_{b_i}} + t_{\mathrm{prediction}}, \tag{10}$$

where $t_{\mathrm{queuing}_{b_i}}$ is the time a behavior of an IoT device waits in the queue of FN_i to get processed and $t_{\mathrm{prediction}}$ is the time required by FN_i to detect the behavior of the IoT device. □

5.5. Cloud Update. After the behavior of an IoT device is sent from a fog node FN_i to cloud C the cloud node receives the behavior of the IoT device and updates the behavior of the device in the IoT device information table. This table is updated always after a response is received from any fog node FN_i. Algorithm 4 shows the steps to update the behavior of IoT devices in the local memory of the cloud.

5.6. Fog Node Update. In this stage, cloud C sends the Target_{b_i} of IoT devices to FNs through communication channels GT_C, GT_C to BS, BS to GT_i, and GT_i to FN_i for updating the local tables at FN nearer to BS. In future if any communication takes place between neighbouring FNs, the communication is performed only after the behavior verification from the local table information. If found to be attacker then further communication with neighbouring node in the network is stopped. Algorithm 5 shows network update at FN.

Theorem 3. *The total time cloud C takes to update at FN about the attacker behavior at any time t is represented by TUD (time to update device layer about attackers).*

Proof. Let, at time t, the set of predicted behaviors for n attacker devices be represented as $\{\mathrm{Target}_{b_1}, \mathrm{Target}_{b_2}, \ldots, \mathrm{Target}_{b_m}\}$ for m IoT devices. This set is sent as a message M to the FNs in the whole network. For this, the cloud node C sends M to the FN_i in TUD time which is calculated as follows:

$$\mathrm{TUD} = t_{C-\mathrm{GT}_C} + t_{\mathrm{GT}_C-\mathrm{BS}} + t_{\mathrm{BS}-\mathrm{GT}_i} + t_{\mathrm{GT}_i-\mathrm{FN}_i}, \tag{11}$$

where $t_{C-\mathrm{GT}_C}$ is the time to send the message M from C to GT_C, $t_{\mathrm{GT}_C-\mathrm{BS}}$ is the time to send the message M from GT_C to BS, $t_{\mathrm{BS}-\mathrm{GT}_i}$ is the time to send the message M from BS to GT_i, and $t_{\mathrm{GT}_i-\mathrm{FN}_i}$ is the time to send the message M from GT_i to FN_i. □

6. Performance Evaluation

The performance of the proposed framework is evaluated using Python 3. The machine used for this performance evaluation has Windows 10 OS, core i7-11370 processor, 3.30 GHz processor speed, and 16 GB RAM. The DNMLP model used in the present framework is compared with the LSTM model and some conventional ML models such as SVM, KNN, LR, and RF. The performance is evaluated using the following performance parameters:

(1) CA (classification accuracy): The number of predictions made correct from the observed values is called CA and it is represented below as follows:

$$CA = \frac{TP + TN}{TP + FN + TN + FP}, \tag{12}$$

where TP is the true positive, TN is the true negative, FP is the false positive, and FN is the false negative.

(2) $F1$-Score: The harmonic mean of precision and recall to know the accuracy better is shown by $F1$-score and it is represented below as follows:

$$F1 = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{13}$$

(3) Precision: In which proportion the instances are correctly classified of a particular positive class from total classified instances of that class and it is represented below as follows:

$$\mathrm{Precision} = \frac{TP}{TP + FP}. \tag{14}$$

(4) Recall: Recall means the proportion of actual instances correctly classified for a particular class and it is represented below as follows:

$$\mathrm{Recall} = \frac{TP}{TP + FN}. \tag{15}$$

6.1. Simulation Setup for DDoS Attack Detection Using DLMs. Firstly, using Python 3, DNMLP and LSTM DL models and RF, LR, KNN, and SVM ML models are implemented to find the best model with high accuracy. The DDoS attack SDN dataset [13] mentioned in Section 5.2.4 is used for prediction. For the implementation of DNMLP and LSTM deep learning models, the Keras package on TensorFlow is installed in Anaconda for deep learning support. The RF, LR, KNN, and SVM models are also implemented using SkLearn Package and are also used for performing preprocessing and

```
    Input: DLM_Selected, $\{FN_1, FN_2, \ldots, FN_n\}$, $b_i$
    **Output:** Update cloud $C$
(1)   **for** $FN_1$ to $FN_n$ **do**
(2)     **for** all IoT devices of $FN_i$ **do** ▷ $i = 1, 2, 3, \ldots, n$
(3)       $Target_i$ = DLM_Selected($b_i$); ▷ Target is the label assigned to the behavior of the IoT device.
(4)         $FN_i \xrightarrow{\text{Sends } Target_i} C$
(5)     end for
(6)   end for
```

Algorithm 3: Classification of IoT device behavior by fog node.

```
    **Input**: $Target_{b_i}$ of $iot_i$
    **Output**: Cloud IoT device table update
(1)   loop()
(2)   {
(3)   $C$ receives $Target_{b_i}$ of $iot_i$;
(4)   Update_Table($Target_{b_i}$); ▷ Update the behaviour of the IoT device at cloud IoT device information table.
(5)   }
```

Algorithm 4: Cloud update.

```
     **Input**: $Target_{b_1}, Target_{b_2}, \ldots, Target_{b_n}$
     **Output**: updating of local table at FN
(1)    loop()
(2)    {
(3)    $C \longrightarrow {}^{Target_{b_i}} GT_C$
(4)    $GT_C \longrightarrow {}^{Target_{b_i}} BS$
(5)    $BS \longrightarrow {}^{Target_{b_i}} GT_i$
(6)    $GT_i \longrightarrow {}^{Target_{b_i}} FN_i$
(7)    **for** all FN near BS **do**
(8)      Update_LocalTable($Target_{b_i}$);
(9)    **end for**
(10)   }
(11)   **if** communication starts between $FN_i$ and $FN_j$ **then**
(12)     **if** $FN_j$ == DDoS attacker ▷ $FN_i$ checks in local table of itself. then
(13)       No communication;
(14)     **else**
(15)       Communication occurs;
(16)     **end if**
(17)   **end if**
```

Algorithm 5: FN update.

performance evaluation metrics. Accuracy and loss performance graphs are obtained using the package matplotlib.

Using DDoS-SDN dataset, we trained and evaluated DNMLP and LSTM DLMS and other ML models for binary classification (normal or attacker). DDoS-SDN dataset [13] is used to detect the capability of DNMLP, LSTM DLMs, and ML models for attack detection. The dataset contains a total of 23 features including the target label. In our work to predict the attacks, we considered 14 features, and 8 features are discarded as some of the features are having zeros and some are having no impact on the target label. The features of the list (1, 2, 3, 4, 17, 20, 21, and 22) were removed from the dataset. On removal of these features, the computational overhead is reduced, and also the model is trained on necessary data. Using the standardization technique the dataset is scaled on various features with varying magnitudes of values and partitioned into two subsets in the ratio of 80 : 20 as training and test dataset. The aim of partitioning the dataset in a ratio of 80 : 20 is for training the model with adequate information and to substantiate the created model with appropriate information.

In the proposed framework to obtain the best-trained model using DNMLP, we considered a batch size of 10 with 40 epochs on the Adam optimizer for binary classification. Keras on TensorFlow is used for constructing NN on the DNMLP model for the DDoS-SDN dataset, considering 14 input values and 1 output value. One categorical attribute, namely, protocol, is one-hot encoded. In this model, we used two hidden layers with a dimension of 16. The model used to create NN using Keras is sequential which takes the output of each layer as input to the next layer using the add-on model. To specify a fully connected layer we used Dense from the Keras package on 16 input dimensions to generate 16 output dimensions on the ReLu activation function by passing as an argument to add function. In the last layer, the output dimension is 1 to obtain binary classification by using the sigmoid as an activation function.

In LSTM, we used a learning rate of 0.001 for batch size 64 with 40 epochs on the Adam optimizer algorithm that has an input layer with 16 input dimensions and 16 as output dimension space, two hidden layers with 16 input dimensions and 16 as output dimension, and one output layer using the sigmoid activation function. From [1], it is observed that, with an increase in the number of hidden layers on a batch size of 128 with 100 epochs on different learning rates of 0.01, and 0.001 LSTM obtains higher accuracy.

Sklearn package is the most useful library for ML in Python. It is used to model the data on both supervised and unsupervised ML algorithms. Here we considered supervised ML algorithms such as RF, LR, KNN, and SVM. For building an RF ensemble classifier we used sklearn.ensemble.RandomForestClassifier which contains parameters of max_depth and random_state. The max_depth as 5 represents a tree with maximum depth from root to leaf being 5 and parameter value of random_state can be either int or none which we considered as 0 and also by default *n*_estimators are 100 which contain 100 decision trees. LR is also used for binary classification which is imported from sklearn.Linear_model.LogisticRegression. By default, lbfgs solver in LR is used in the optimization problem. To implement SVM the module from sklearn.svm.LinearSVC is imported. The LinearSVC is highly used on larger datasets and by default the RBF kernel is used in implementing the algorithm. In KNN the algorithm is imported from sklearn.neighbors.KNeighborClassifier with a parameter n_neighbor of either int or default which takes a value of 5 and represents the number of neighbours. In this work based on RMSE, we found $K = 1$ which obtain high accuracy among the other K values.

6.2. Results and Discussion. The performance of DNMLP for the DDoS-SDN dataset [13] in training and validation accuracy is shown in Figure 7. Binary classification with a batch size of 10 and an increase in epochs shows an increase in training and validation accuracy in Figure 7, but the best accuracy is obtained at the 34th epoch as 99.55% from experiment evaluation. From Figure 8, the training and validation loss decrease converged at the 40th epoch. With an increase in batch size to 64 at the 40th epoch, the model witnessed a slight decrease in all performance metrics but by the increase in epochs, the increase in validation accuracy and decrease in validation loss were observed with good fit learning curves by overcoming underfitting and overfitting. The training and validation accuracy of the DDoS-SDN

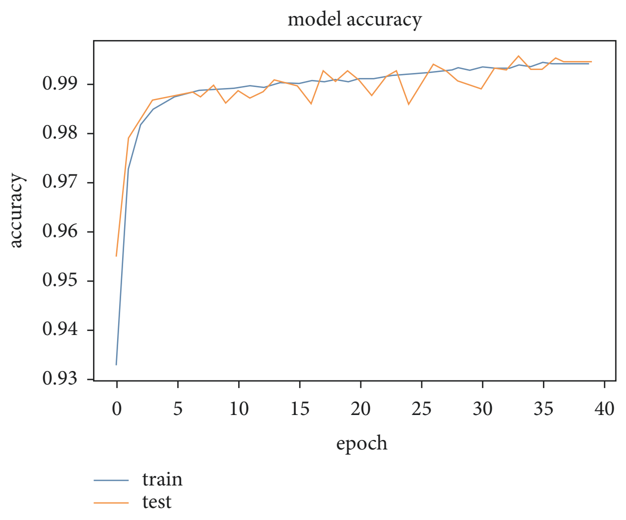

Figure 7: Model accuracy for DNMLP using accuracy vs. epoch graph.

dataset using LSTM is shown in Figure 9. Binary classification with a batch size of 64 and an increase in epochs shows an increase in training and validation accuracy in Figure 9, but the best accuracy is obtained at 28th, 33rd, and 39th epochs as from experiment evaluation. From Figure 10, the training and validation loss decrease converged after the 40th epoch of 64 batch size.

Figure 11 shows the comparison of performance metrics among the two DLMs and four ML models on the DDoS-SDN dataset. In terms of accuracy, precision, recall, and $F1$-score, DNMLP performs better than considered ML models and LSTM. The accuracies of all DNMLP, LSTM, and other ML models with binary classification are shown in Figure 12 such that the DNMLP model is predicted to be a higher ranking than all other models.

Using the DDoS-SDN dataset we trained and evaluated DNMLP, LSTM, and ML models for binary classification and found DNMLP shows better accuracy than all other models in predicting the behavior of the IoT devices as an attacker or normal. For the DDoS-SDN dataset, DNMLP achieves 99.44% accuracy, 99.02% precision, 99.60% recall, and 99.30% $F1$-score. LSTM achieves 97.84% accuracy, 96.18% precision, 98.47% recall, and 97.31% $F1$-score. RF achieves 97.68% accuracy, 94.62% precision, 99.81% recall, and 97.15% $F1$-score. LR achieves 75.87% accuracy, 71.68% precision, 64.72% recall, and 68.02% $F1$-score. KNN achieves 98.73% accuracy, 98.26% precision, 98.55% recall, and 98.40% $F1$-score. SVM achieves 75.89% accuracy, 72.39% precision, 63.40% recall, and 67.59% $F1$-score. Accuracy is only a basic measure in evaluating a model but proved to be good when a balanced dataset is used. In this work, the dataset used is imbalanced as the number of normal users is 60.92% and malicious users are 39.08%; also we used an 80 : 20 train and test split. So, there may be a chance of larger false positives (FP) than false negatives (FN). In such cases, it is better to account for the other performance metrics like precision, recall, and $F1$-measure. The recall does only consider false negatives and true positives (TP) and thus recall may be high. Precision does only consider FP and TP;

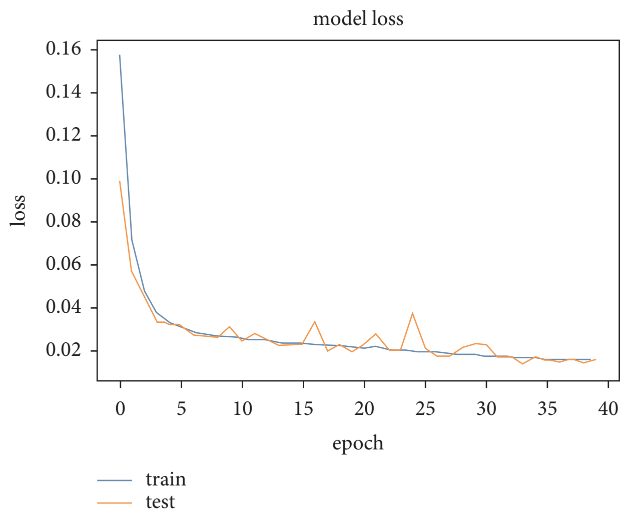

Figure 8: Model loss for DNMLP using loss vs. epoch graph.

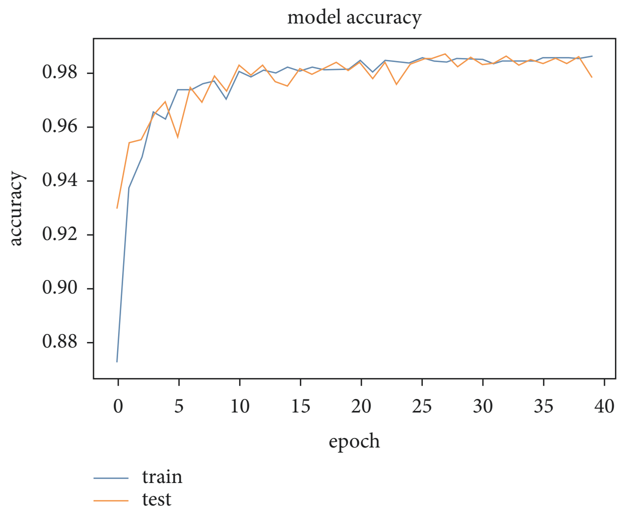

Figure 9: Model accuracy for LSTM using accuracy vs. epoch graph.

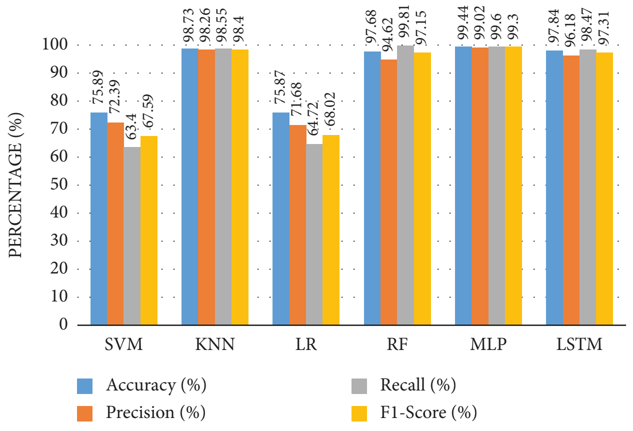

Figure 11: Comparison of models based on accuracy, precision, recall, and $F1$-score.

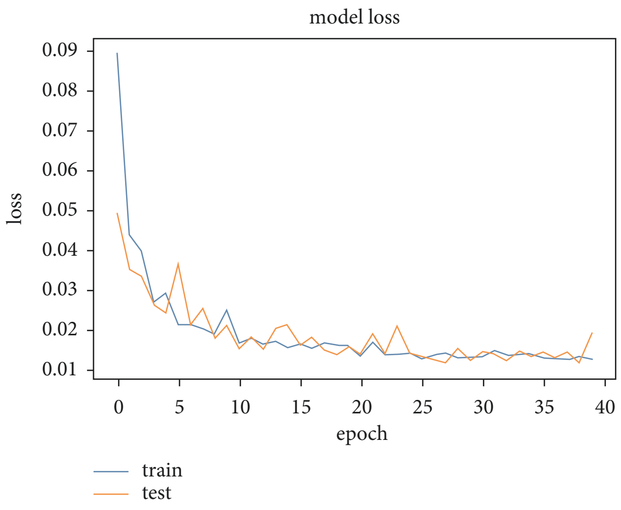

Figure 10: Model loss for LSTM using loss vs. epoch graph.

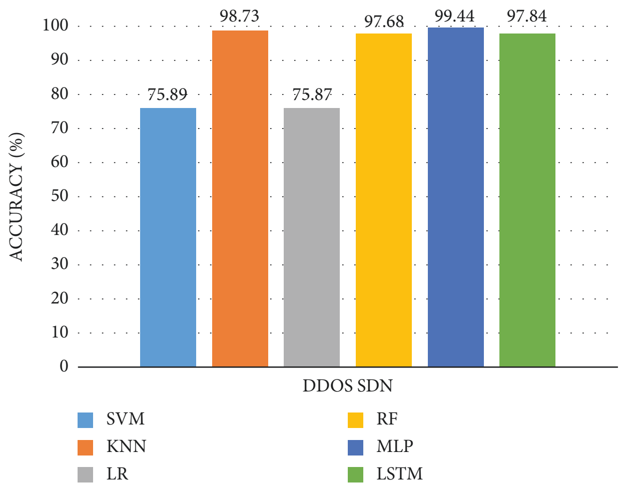

Figure 12: Comparison for best model selection using accuracy parameter.

Table 2: Network simulation setup for analysing device behavior detection time at fog.

Sl. no.	Parameter	Value
1	Number of cloud node	1
2	Number of fog nodes	1–10
3	Number of IoT devices	10–100
4	Average behavior detection time for DNMLP for 10 samples	0.0004 sec
5	Average behavior detection time for LSTM for 10 samples	0.001 sec
6	Number of simulation runs	10

it may suffer from low value. Here $F1$-score, a harmonic mean of precision and recall, will have its importance in deciding the performance of the model and it is evident by the results showing the highest value of the $F1$-score (99.30%) with DNMLP.

To test the scalability issue, the simulation environment is set as a three-tier architecture where 1 cloud server is connected to multiple fog nodes. In the last layer, we assume that there are 10–100 IoT devices connected to the nearest fog nodes. For example, if 1 fog node is there and 10 IoT devices are there, then 10 IoT devices directly connect to the fog node; however, if the number of fog nodes is greater than 1 then the number of fog nodes equally divides the number of IoT devices for providing required service. So, if there are 2 fog nodes then 1 fog node will give service to 5 IoT devices. In this scenario, we assume that an IoT device communicates with the fog node and 1 sample is generated (row). This sample is then processed at the fog node for behavior prediction (normal/attack). The average behavior detection time from the above experiment using DNMLP is found to be 0.0004 seconds for 10 IoT devices/samples and for LSTM it shows an average of 0.001 seconds for 10 IoT devices/samples. From this experiment, we have tested if the number of IoT devices increases concerning the number of fog nodes then what the impact on behavior detection time is. Behavior detection time (BDT) is the time to detect the number of IoT devices as normal or attacker by the fog nodes using DNMLP or LSTM. Table 2 shows the parameters and values for the network simulation.

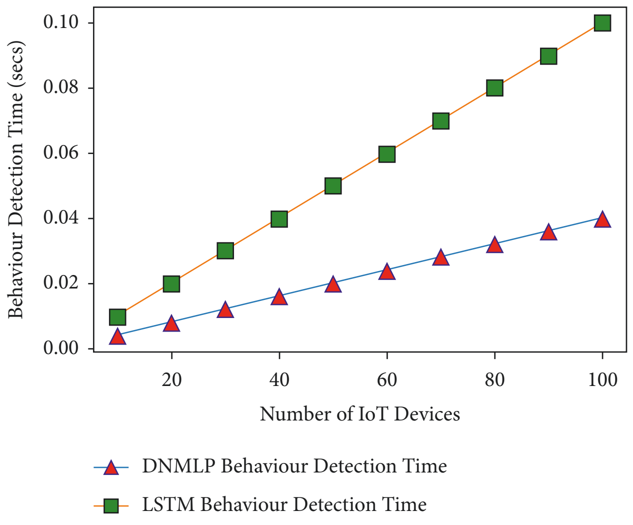

Figure 13: Comparison of behavior detection time for DNMLP and LSTM with 1 fog node in the network.

From Figures 13–17, it is observed that when the number of IoT devices increases in the network then the behavior detection time also increases for all the IoT devices. Figure 13 shows the result when the number of fog nodes in the network is 1 and the IoT devices increase from 10 to 100. From this figure, it is observed that DNMLP shows less behavior detection time than LSTM. The average behavior detection time of 100 IoT devices for DNMLP is found to be 0.022 secs and the average behavior detection time of LSTM is found to be 0.055 secs. Figure 14 shows the result when the number of fog nodes in the network is 3 and the IoT devices increase from 10 to 100. From this figure, it is observed that DNMLP shows less behavior detection time than LSTM. The average behavior detection time of 100 IoT devices for DNMLP is found to be 0.0073 secs and the average behavior detection time of LSTM is found to be 0.018 secs. Figure 15 shows the result when the number of fog nodes in the network is 5 and the IoT devices increase from 10 to 100. From this figure, it is observed that DNMLP shows less behavior detection time than LSTM. The average behavior detection time of 100 IoT devices for DNMLP is found to be

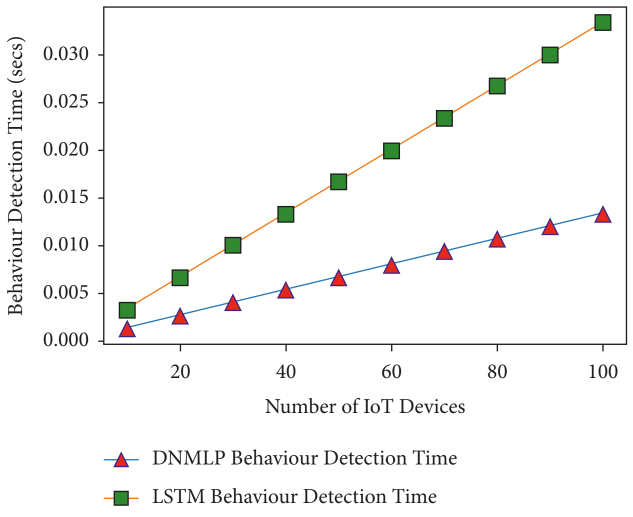

Figure 14: Comparison of behavior detection time for DNMLP and LSTM with 3 fog nodes in the network.

0.0044 secs and the average behavior detection time of LSTM is found to be 0.011 secs. Figure 16 shows the result when the number of fog nodes in the network is 7 and the IoT devices increases from 10 to 100. From this figure, it is observed that DNMLP shows less behavior detection time than LSTM. The average behavior detection time of 100 IoT devices for DNMLP is found to be 0.003142 secs and the average behavior detection time of LSTM is found to be 0.007857 secs. Figure 17 shows the result when the number of fog nodes in

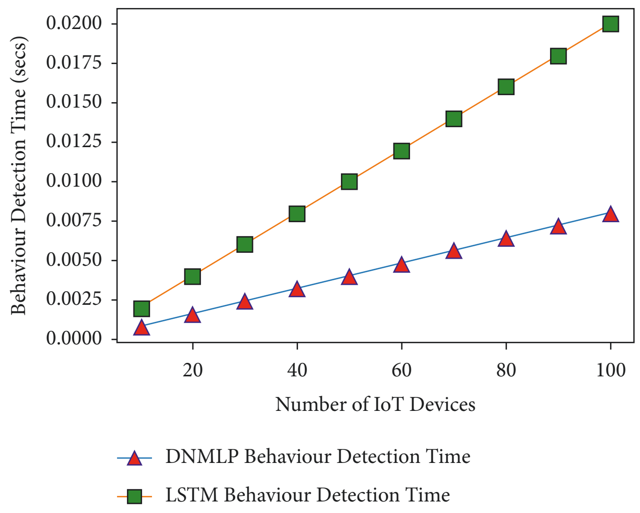

Figure 15: Comparison of behavior detection time for DNMLP and LSTM with 5 fog nodes in the network.

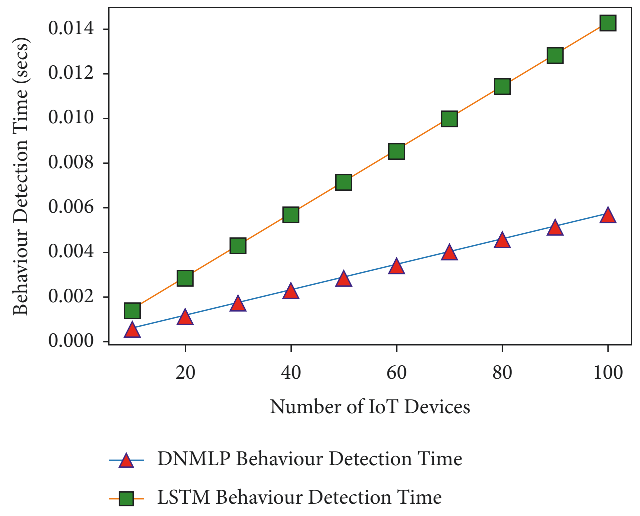

Figure 16: Comparison of behavior detection time for DNMLP and LSTM with 7 fog nodes in the network.

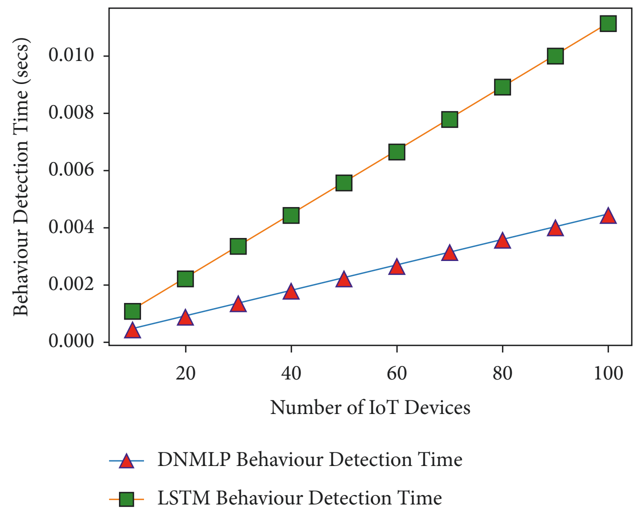

Figure 17: Comparison of behavior detection time for DNMLP and LSTM with 9 fog nodes in the network.

the network is 9 and the IoT devices increase from 10 to 100. From this figure, it is observed that DNMLP shows less behavior detection time than LSTM. The average behavior detection time of 100 IoT devices for DNMLP is found to be 0.0024 secs and the average behavior detection time of LSTM is found to be 0.0061 secs. From all these results it also concluded that when the fog nodes in the network increase the behavior detection time reduces.

7. Conclusion

In this paper, we designed a DI-ADS scheme to detect DDoS attacks for fog-based IoT systems using DLM. Firstly, the network is set with three layers: IoT device layer, fog layer, and cloud layer, and then DNMLP, LSTM DL models, RF, LR, KNN, and SVM ML models are evaluated to predict the best model with high accuracy to be deployed at the fog nodes. The result shows 99.44% accuracy using the DNMLP model and hence the fog layer in the network is deployed with DNMLP where each fog node is enabled with DNMLP. It performs binary classification into two classes 1 and 0 as attacker and normal devices, respectively, and sends the device behavior to the cloud for an update. Then, the cloud sends the attacker information to the IoT device layer where each device knows about the attacker device status in the neighbourhood. Further communication with these attacker nodes is decided by the individual IoT devices by verifying the current behavior status. This model will be a better scheme for securing the fog layer from DDoS attacks. In future, we will implement the same scheme for attack detection using new DLMs, hybrid models, and unsupervised learning like Deep Belief Networks (DBNs) by training the fog nodes with an increased size of the dataset and also using newer datasets along with multiclass classification can be performed to detect particular attacks.

In the current work, the DNMLP model shows better accuracy compared to other considered models which have implementation level limitations in obtaining the performance metrics. In the time ahead it would be apposite to train the DL models by altering the batch size and learning rates, and increasing epoch number helps in achieving a better performance benchmark. It is also possible to get better results by refining the process of data preprocessing. Beyond this, we could take on heuristic algorithms for optimizing the DL models.

Data Availability

Data will be available on request basis and will be provided by first author of this manuscript.

Consent

Not Applicable.

Conflicts of Interest

There are no conflicts of interest.

Authors' Contributions

All authors contributed equally to this work.

Acknowledgments

There is no research funding associated with this research work. However, all authors contributed equally to 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, Article ID 74571, 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] 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 (IEEE Cat. No. 03EX795)*, pp. 190–193, IEEE, Darmstadt, Germany, 2003.

[8] B. S. 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.

[9] K. Lakshmanna, R. Kaluri, N. Gundluru et al., "A review on deep learning techniques for iot data," *Electronics*, vol. 11, no. 10, p. 1604, 2022.

[10] P. Ray, R. Kaluri, T. Reddy, K. Lakshmanna, and K. Praveen, "Contemporary developments and technologies in deep learning–based iot," in *Deep Learning for Internet of Things Infrastructure*, CRC Press, Boca Raton, FL, USA, 2021.

[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, 2020.

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

[14] K. Bhushan and Deepali, "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, Bangalore, India, May 2017.

[15] 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, 2019.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] P. Prabhat 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.

[21] 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.

[22] 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.

[23] S. Askar, "Deep learning and fog computing: a review," 2021, https://ideas.repec.org/a/aif/journl/v5y2021i6p197-208.html.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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 Cham, New York, NY, USA, 2022.

[29] 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.

[30] 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.

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

[32] R. Kaluri and P. R. Ch, “Optimized feature extraction for precise sign gesture recognition using self-improved genetic algorithm,” *International Journal of Engineering and Technology Innovation*, vol. 8, no. 1, pp. 25–37, 2018.