Deep Learning Technique to Identify the Malicious Traffic in Fog based IoT Networks

Akshata Deshmukh, Tanuja Pattanshetti

***Abstract**: The network of devices known as the Internet of Things (IoT) consists of hardware with sensors and software. These devices communicate and exchange data through the internet. IoT device-based data exchanges are often processed at cloud servers. Since the number of edge devices and quantity of data exchanged is increasing, massive latency-related concerns are observed. The answer to these issues is fog computing technology. Fog computing layer is introduced between the edge devices and cloud servers. Edge devices can conveniently access data from the fog servers. Security of fog layer devices is a major concern. As it provides easy access to different resources, it is more vulnerable to different attacks. In this paper, a deep learning-based intrusion detection approach called Multi-LSTM Aggregate Classifier is proposed to identify malicious traffic for the fog-based IoT network. The MLAC approach contains a set of long short-term memory (LSTM) modules. The final outcomes of these modules are aggregated using a Random Forest to produce the final outcome. Network intrusion dataset UNSW-NB15 is used to evaluate performance of the MLAC technique. For binary classification accuracy of 89.40% has been achieved using the proposed deep learning-based MLAC model.*

***Keyword**: Internet of Things, fog computing, security, deep learning.*

I. INTRODUCTION

The Internet of Things (IoT) has turn into a more accepted and promising technology by providing numerous opportunities for various industrial systems and different home automation applications. The rapid progress of IoT has contributed to the creation of an enormous quantity of data and increased requirement for various new resources and services [1]. The primary task of cloud computing is to address these issues. However, traditional cloud computing comes up with challenges like low capacity, network failure and high latency [2]. IoT applications require low latency with location awareness and mobility support. To conquer these limits, a new idea that favors combination of IoT and Fog computing is proposed. Fog computing is a technology, which is developed by CISCO. It transfers the services and data to the network edge from the cloud. It is established on distributed computing. The main reason for developing it is to handle the immense amount of data generated from the speedily increasing amount of IoT devices with minimum latency. Fog Computing technology is greatly virtual domain which offers network services between cloud data centers and devices that are situated at the edge of the network, computing services and storage[3]. It is normally treated as an enhanced version traditional cloud computing. In [4] author defines fog computing comprehensively. The definition is based on technologies and challenges that form the fog, with more stress on few major properties. Cloud, fog and End-user devices, collaboratively make a service model consisting of three layers. The attributes of FC introduce new security and privacy challenges. We cannot directly use the existing privacy and security solutions that are available for cloud computing on fog computing because it has some dissimilar properties. There is little work on privacy and security issues of fog computing as the technology are still in its initial stage. Among various available security solutions, the Intrusion Detection System is the most widely accepted system to provide protection from the attacks [5]. Intrusion detection systems identify the intruders and provide security to the network and data by denying access to unauthorized users. It monitors and alerts the security team in various potentially harmful situations [6]. In past research, many of IDS based on Artificial Intelligence (AI) methods like Deep Learning (DL) and Machine Learning (ML) have been proposed. However, proposing effective and efficient attack detection systems based on AI methods is still an area of research. Recent advancements in cyber-attacks have led to the need for enhanced IDS. In this paper, a DL based IDS that uses LSTM for intermediate classification and Random Forest for aggregation is proposed. UNSW-NB15 dataset constructed in IoT environment is used for performance evaluation of Multi-LSTM Aggregate Classifier (MLAC) IDS model as it contains modern IoT attacks.

Manuscript received on 30 June 2022 | Revised Manuscript received on 08 July 2022 | Manuscript Accepted on 15 July 2022 | Manuscript published on 30 July 2022.

* Correspondence Author

Akshata Deshmukh*, Department of Computer Engineering, College of Engineering, Pune (Maharashtra), India. Email: deshmukhaa20.comp@coep.ac.in

Dr. Tanuja Pattanshetti*, Department of Computer Engineering, College of Engineering, Pune (Maharashtra), India. Email: **trp.comp@coep.ac.in**

II. MATERIALS AND METHODS

There are many intrusions detection system model have been proposed using artificial intelligence. M. Pahl and F. Aubet [7] proposed an is IDS for anomalies in IoT systems. Iterative reducing in balanced manner, hierarchies-based clustering and K Means. Performance of model is evaluated by self-generation of a dataset namely DS2OS in an IoT environment. Proposed model achieved 96.3% accuracy according to their experimental results. Proposed model gives overall high performance but for comparison prediction results according to class are not used. Pajouh HH [8] designed an anomaly detection system. In the proposed system they used two-tier classification models. The proposed model implemented.

Naive Bayes and the factor related to K-Nearest Neighbor algorithm. To reduce dimension linear discriminant analysis and PCA is used. The NSL-KDD dataset is used. Dangerous attacks like R2L and U2R have high detection results according to their proposed model but for binary classification the model gives very low detection rate. Nour Moustafa and Slay J [9] designed an Intrusion Detection System, which is carried out using expectation-maximization clustering, logistic regression, artificial neural network, decision tree and Naive Bayes. They have used KDD99 and UNSW-NB15 datasets. The Gain ratio and Pearson's correlation n coefficient strategy is used for selection of features. According to proposed, DT gives 85.56% accuracy for UNSW-NB15 and ANN gives 97.04% accuracy for KDD99 dataset. PajouhHH [10] proposed an anomaly recognition system for Internet of Things networks. The proposed model uses two tier classification and two layer dimension. For reduction of dimension linear discriminate analysis and PCA is used. For calculation of classification certainty factor Naive Bayes (NB) and K-nearest neighbor (KNN) is used. Main focus of the model is low frequency attacks like U2R and R2L. NSL-KDD dataset is used to achieve experimental results. Unified IDS (UIDS) has been proposed [11] by V Kumar for IoT environment. Detection system proposed by this model is based on gateway nodes and cluster heads to protect against various attacks in IoT. NB-15 dataset is implemented for evaluation. Overall accuracy of the system is improved but the clustering mechanism that is used to form the group of sensors is not disclosed by this model. Prabhat Kumar proposed [12] anomaly-based IDS for IoT network using fog computing. The model proposes to decentralize the existing security mechanism at cloud to fog nodes. For evaluation of model various ML classifiers like Decision Tree (DT), RandomForest and KNN are employed. Performance of the proposed system is evaluated using dataset DS2OS. Proposed system outperforms in detection low FAR and accuracy using Random Forest. IDS system for IoT network is proposed by Diro [13] using fog computing. The DL approach proposed is based on distributed ID systems. The results obtained after experiments claim that accuracy of parallel distributed architecture is higher the accuracy obtained by centralized model. The NSL-KDD dataset is used for training and implementation of IDS. Detection rate of model is 93.66% and FAR is 4.97%. But NSL-KDD contains very outdated attacks. Kambourakis G proposed [14] (DENDRON) which is a rule based intrusion detection system. According to proposed methodology, detection results are generated using DT and Genetic Algorithms are used for classification. KDD99, NB-15 and NSL-KDD datasets are used for experimentation. According to their system, detection rate is minimum for Worms, DoS and Shellcode and attacks in UNSW-NB15dataset. S Prabavathy and K.Sundarkantham designed a detection system [15] with fog computing. The proposed system is consisting of two levels. In the first level, attack is detected at local fog nodes, and in the second level, IoT system state is summarized at cloud. NSL-KDD dataset is used for implementation and achieved accuracy of 97.36%. Detection rate according to authors is 25% faster for fog based implementation than implementation at cloud. Prabhat Kumar [16] proposed distributed IDS with the help of fog computing technology for protection of IoT network, where they used ensemble Gaussian naive Bayes, KNN and XGBoost at first stage, then used Random Forest for final classification. UNSW-NB15 dataset and DS2OS dataset are implemented for performance evaluation proposed model. Yanxia Sun presented [17] IDS using feature selection technique. They applied feature reduction method based on filters using XGBoost algorithm. ML approaches like, DT, L Regression, kNN, Support Vector Machine and Artificial Neural Network are applied on full and reduced dataset. They implemented both multiclass and binary classification methods. UNSW-NB15 dataset is used for implementation. Authors claim that XGBoost algorithm used for feature selection increase testing accuracy for some algorithm. A deep learning model that comprises two stages is proposed [18] by Khan. The model uses auto encoder in first stage and soft max classifier in second stage. The model is trained using feature representation in first stage and result of first stage is used as an input to second stage for abnormal traffic detection. KDD99 dataset and UNSW-NB15 dataset are used for evaluation of system. Kaiyuan proposed [19] Network Intrusion Detection System (NIDS). In proposed system, One-Side Selection method is used for reduction of noisy data values form classes [20]. Deep learning model that consists of Convolutional Neural Networks and Bi-directional LSTM is used to conduct predictions, where CNN is used for selection of spatial attributes and Bi-LSTM is used for selection of temporal attributes. NSL-KDD and NB-15 datasets are used for evaluation. 77.16% accuracy for UNSW-NB15 dataset and 83.58% accuracy for NSL-KDD dataset have been achieved. Being comparatively a newer technology, fog computing is less explored area of Research. Fog computing has some distinct characteristics from cloud computing, making the cloud security solutions insufficient for it hence there exists an abundant scope of research in security of fog computing. When the size of data increases extensively, traditional ML algorithms are inefficient as compared to DL algorithms. However, the past research works indicate that the deep learning approaches for security of fog computing are under-explored.

III. THEORY

In the first stage of Multi-LSTM based aggregate classifier (MLAC) technique six LSTM models that are combination of single layer LSTM and multilayer stacked LSTM are used. These LSTM models have different parameters to learn about numerous patterns of network traffic that are related with attack and clean network cases. "Fig. 1." Gives the basic idea of steps involved in MLAC technique. The diagram consists of four components fog layer, stack of lstm models, classifier model and finally the outcome of the system. Fog layer is the part where network traffic data is received and processed. From the network traffic data important features and parameters are taken by 6 different LSTM models that generate the confidence rates over certain outcome for the category of traffic. These confidence rates are further processed by the next component – classifier model which can be any machine learning classification algorithm that would lead to the next component of the system i.e. outcome. It has two possibilities, one is that the traffic being classified as attack or another of being classified as normal.

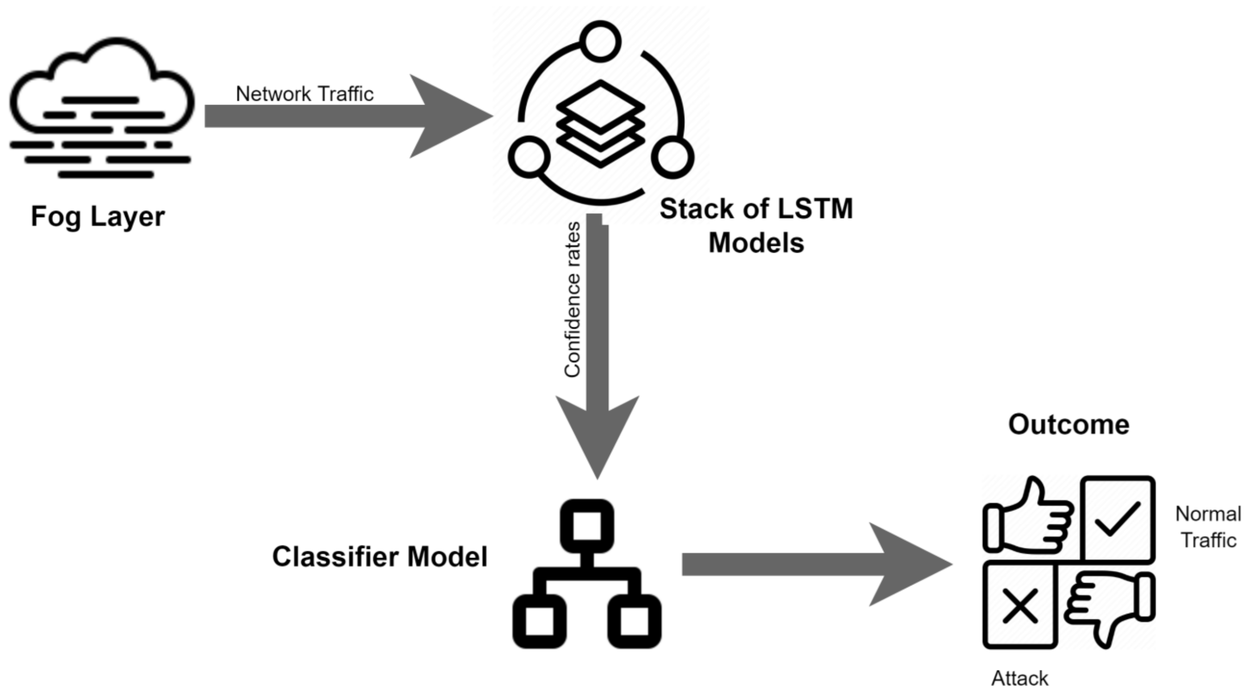

Fig. 1. Basic structure of MLAC Technique

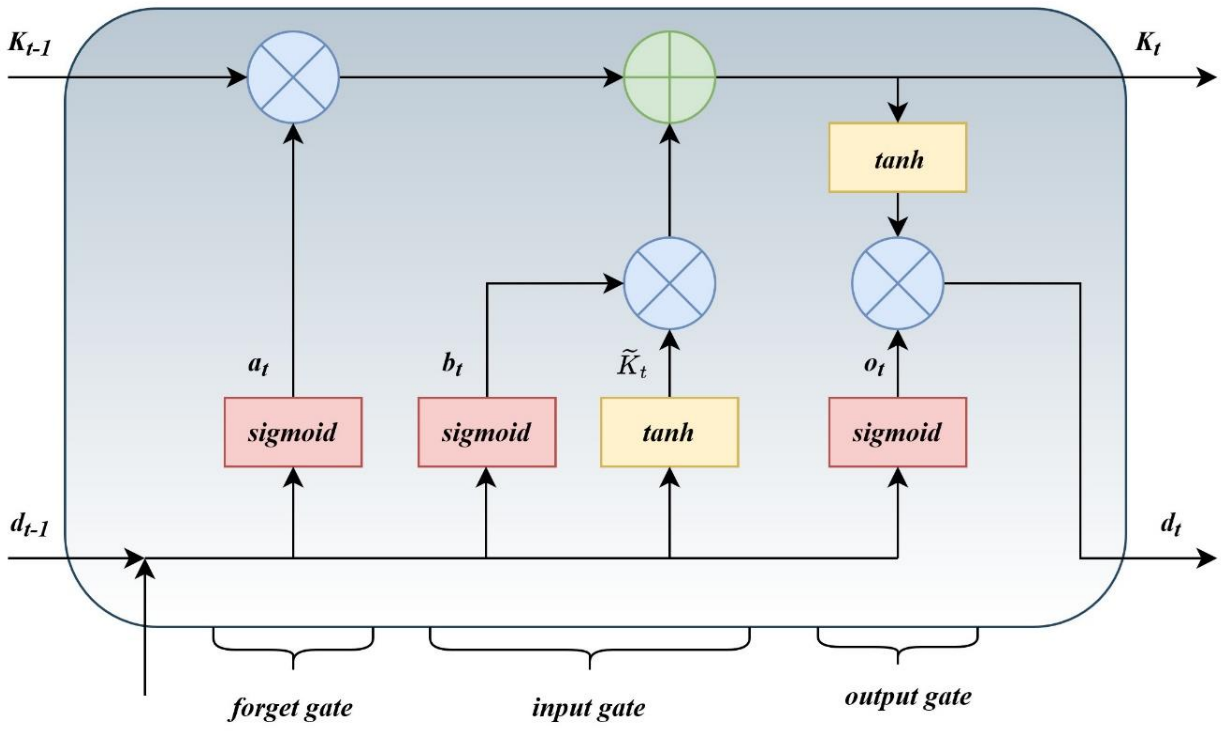

Fig.2. LSTM cell structure

A. Long Short-Term Memory Model

LSTM is a kind of artificial recurrent NN commonly used in deep learning field. Unlike, a regular feed forward neural network, LSTM consist of feedback connections also. "Fig. 2." shows the basic structure that includes the four components of an LSTM cell. A simple LSTM unit (cell) consists of four parts: Output Gate, Cell State, Input Gate, and Forget Gate which are discussed in brief below:

1) Forget Gate

This is the first step in the LSTM cell, wherein the decision over retention of information of the previous timestamp is made. In this decision, the cell forgets the information or retains it. Values from preceding hidden state *d(t-1)* and current input *y(t)* pass through sigmoid function. Then, values between range 0 and 1 are generated by sigmoid function.

If value closer to 1 is generated, then forget gate retains part of old output, else it forgets the old output. Cell uses this value *a(t)* for point-by-point multiplication.

$$a_t = \sigma\left(W_a.\left[d_{t-1}, yt\right] + v_a\right)$$

Where, t= timestamp,
a_t = forget gate at timestamp t,
y_t = input value,
d_{t-1} = hidden state at previous stage,
W = calculated weight matrix values joining input gate and forget gate.

2) Input Gate

Input gate updates cell status by performing following operations.

First, preceding hidden state *d(t-1)* and current input *y(t)* pass through second sigmoid function. Then, values between 0 and 1 are generated. Here, 0 indicates important value and 1 indicates irrelevant value. Next, same values from current state and preceding hidden state pass through *tanh* function. Then, *tanh* vector creates vector *(K ~ (t))* with values between -1 and 1. Values generated by activation function are used for point-by-point multiplication.

$$b_t = \sigma\left(W_b.\left[d_{t-1}, y_t\right] + v_b\right)$$

$$K_t = \tanh\left(W_k.\left[d_{t-1}, y_t\right] + v_k\right)$$

where, *t*= timestamp,
b_t= input gate at timestamp *t*,
W_b= calculated weight matrix values for sigmoid operator joining input gate and output gate,
v_b=bias vector with respect to W_b
K_t =value calculated by *tanh*,
W_K= calculated weight matrix value for *tanh* operator joining state information of cell and network output,
v_K=bias vector at timestamp t with respect to W_K

3) Cell State

Cell state operation decides whether to update cell state to new state or to keep previous cell state as it is. The forget vector a(t) is multiplied with preceding cell state K(t-1), if the output is zero, then data in the cell state gets released. Next, point-to-point addition is performed on output value from input vector; this updates the cell state to new cell state K(t).

$$K_t = \tanh\left(W_K.\left[d_{t-1,}y_t\right] + v_K\right)$$

$$K_t = a_t * K_{t-1} + b_t * K_t$$

where t= timestamp,
K_t=information of cell state,
a_t= forget gate at timestamp t,
K_{t-1}=timestamp at previous stage,
b_t=input gate at timestamp t,
K_t =value calculated by *tanh.*

4) Output Gate

The role of this component is to determine the value of the next unknown state based on the current state and preceding hidden state. This hidden state is further used for prediction purposes. The values from preceding hidden state *d(t-1)* and current input *y(t)* pass through third sigmoid function. Then, cell state generates new cell state which passes through *tanh* function. Point by point multiplication is performed on outputs from both the functions. Then the final value is generated, which is used to decide what value hidden state should contain. Prediction is dependent on hidden state. Finally, new hidden state *d(t)* and new cell state *K(t)* are supplied to succeeding time step.

$$c_t = \sigma\left(W_c.\left[d_{t-1}, y_t\right] + v_t\right)$$

$$d_t = c_t * \tanh(K_t)$$

where, t= timestamp,
c_t=output gate at timestamp t,
W_c= calculated weight matrix values for output gate,
v_c=bias vector with respect to $W_{c,}$
d_t=output of lstm

So, the forget gate regulates which appropriate data from the preceding steps is desired. The input gate determines what appropriate data can be supplied from current step, and the next hidden state is finalized by output gate.

B. Stacked LSTM

Normally, an LSTM network model contains only one layer. An LSTM model that contains multiple LSTM layers is called as stacked LSTM. In stacked LSTM model, upper layer does not provide single value output to lower layer, but it provides a sequence output. It allows greater model complexity. Each LSTM memory cell needs 3D array as input and after processing it gives 2D array as output. For stacked LSTM, it is configured such that it gives 3D array as output which will be further used as an input to subsequent layer. By growing the number of layers in LSTM network uplifts its accuracy and performance to learn and recognize various complex sequential patterns. "Fig. 3." shows structure of stacked LSTM

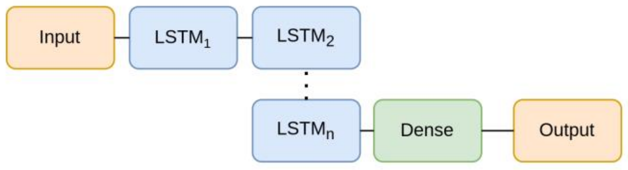

Fig. 1. Stacked LSTM

IV. DATASET

The famous network dataset that resembles the network traffic incoming at fog layer, called UNSW-NB15 is used for training and testing the model. It has real modern normal and synthetic abnormal network traffic. It represents 9 attack families according to behavior of attack**.** "Fig. 4." shows distribution of attacks and "Table-I." shows types of attacks in UNSW-NB15.

There are total 49 features. It has two labeled features, first is attack cat which has name of category of attacks which can be used for multiclass classification. Second is Label, where 0 represents normal and 1 represent attack records. It can be used for binary classification. There are total 82332 rows in testing set and 175341 rows in training set, which are further explained in "Table-II". Older network intrusion dataset KDDCUP99 dataset has huge number of redundant records and many missing records which affect detection results and can change behavior of data. NSLKDD which is an improved version of KDDCUP99 addresses issues in it. But, NSLKDD dataset has many outdated attacks, to address this, a new dataset UNSW-NB15 is created.

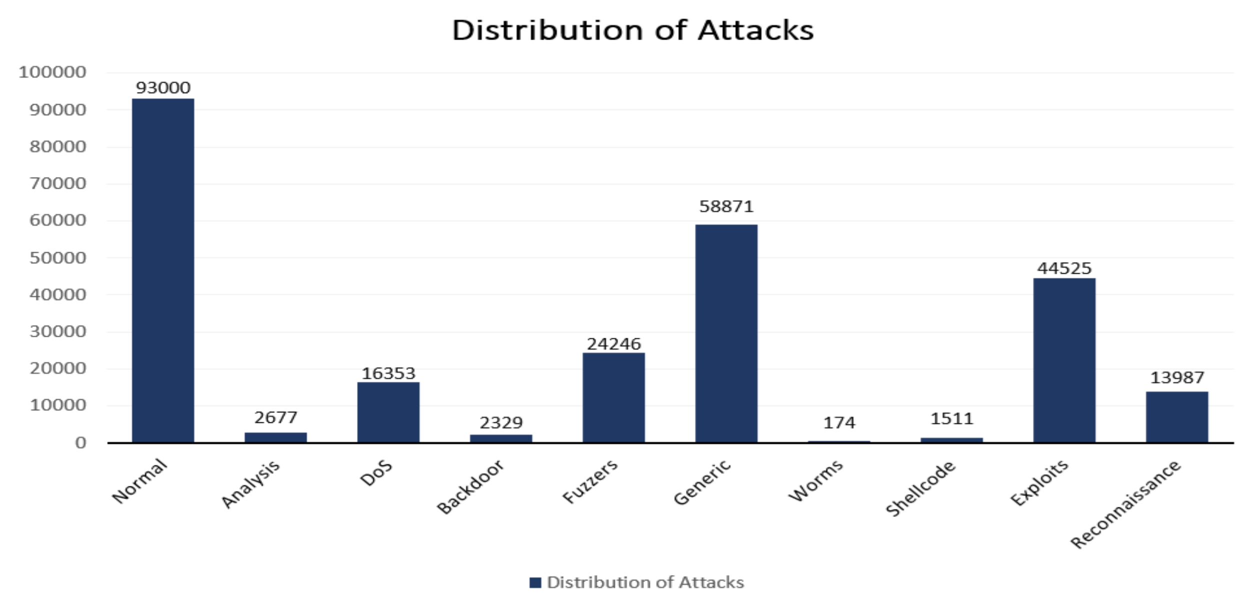

Fig. 4. Attack distribution in NB-15 dataset

Table-I: Types of attacks in UNSW-NB15

Type of Attack	Description
Analysis	Analysis is an intrusion which enters into the web based applications through web scripts like HTML files, ports like port scans and emails like spam.
DoS	In denial-of-service (DoS) attack, attacker shuts down the targeted system or network by interrupting its normal operation to make it unavailable to its intended users.
Backdoor	It is the method of avoiding a standard authentication. In this attack, hackers use a specific type of malware to avoid normal authentication procedures to gain access to a target system.
Fuzzers	In the Fuzzers attack, attacker tries to find security vulnerabilities in a network, an operating system, or an application by injecting it with large quantity of arbitrary data to break down the system.
Generic	Generic is a type of collision attack that works opposed to all block ciphers
Worms	In this attack, a malicious piece of code is used, which spreads through a computer network and attacks the targeted computer.
Shellcode	It is a variety of malware where the attacker controls the targeted machine by penetrating a small code that starts through a shell.
Exploits	Exploit is a piece of code intended to find and take advantage of vulnerability, security flaw, or a glitch which cause an unintended actions on a network system or a host device.
Reconnaissance	Reconnaissance attack collects information regarding a target network or a target system to escape from its security controls.

Table-II. Total instances in Train and Test sets

Class	Instances in Training Set	Instances in Testing Set
Analysis	2000	677
Normal	56,000	37,000
DoS	12,264	4089
Backdoor	1746	583
Fuzzers	18,184	6062
Generic	40,000	18,871
Worms	130	44
Shellcode	1133	378
Exploits	33,393	11,132
Reconnaissance	10,491	3496
Total	175,341	82,332

V. CALCULATION

Multi-LSTM based aggregate classifier technique comprises of two main phases namely, the confidence computation phase and final classification phase.

A. Confidence Computation

The first phase of the system consists of a stack of LSTM models to learn the patterns in network traffic (both attack and normal scenarios). These LSTM models are combination of single layer LSTM models and stacked LSTM models. "Table-III" shows the variants of LSTM used in MACL for experimentation purposes. The input to this phase is the network traffic data incoming from the edge layer devices. In an experimentation environment to train the model, a benchmark dataset UNSW NB-15 is used as the network traffic data. Output of this phase is the confidence rate of the selected class. This confidence rate is supplied to the next phase i.e. Final Classification.

$$cr_i = LSTM_i\left(D\right)$$

where cr_i is the confidence rate calculated for ith LSTM model and D is the dataset. Algorithm for the confidence Computation phase is as follows,

Algorithm: Phase I: Confidence Computation

```
1   Input: Dataset Output: Confidence rates For each LSTM variant do
2       add desired number of layers
3       set unit_size for each layer
4       activation function = sigmoid //since we using binary classification approach
5       Train the model using the training dataset
6       (confidence)iß Compile and execute the model on the test dataset
7   end
```

Table-III. Variants of LSTM models used in MLAC

Models	Number of Layers	Size of Layer
LSTM *M-1	1	{10}
LSTM M-2	1	{20}
LSTM M-3	2	{10,10}
LSTM M-4	2	{20,10}
LSTM M-5	3	{10,10,10}
LSTM M-6	3	{20,10,5}

*M: Model

B. Final Classification

A classifier is included in the last step; it delivers an aggregated final judgment based on the results of the preceding stage. The classifier gets the collection of D' values that are confidence rates of LSTM models for all the classes in the dataset and produces the output. The following information is the input given to the classifier (as computed during the computation of confidence):

$$D' = \bigcup_{i=0}^{n} (cr)_i$$

where, *i* is a particular LSTM model, n is total count of LSTM models and *(cr)*$_i$ is corresponding LSTM model's confidence rate. and n is total count of LSTM models. The classifier assigns the actual label by associatively analyzing the correlation between the confidences ratings derived from LSTMs. In other words, the classifier analyses the output space created by LSTMs to determine if the packet is harmless or trying to launch an attack. Random Forest algorithm is used for classification.

Algorithm for the Classification phase is as follows,

Algorithm: Phase II: Classification

```
     Input: Confidence rates
     Output: Outcome and its evaluation
1    cfds ← Combine the confidence rates obtained from phase I
2    outcome ←classifier(cfds) //Train and fit the classification
     model on cfds
3    Evaluate the performance based on outcome
4    end
```

"Fig. 5." shows the flow diagram of steps that are followed in experimentation to obtain the results for performance evaluation of MLAC technique. On both training and test set of the dataset the phase I and phase II of MLAC technique is applied. Initially, all the categorical object type of features in both datasets are encoded into numeric features. Then the dependent and independent of each dataset are separated into different sets. Each set is further normalized and reshaped according to input of LSTM model. The dependent and independent sets of train dataset are given as input to each variant of LSTM model after configuring each of it for specific epoch, unit_size, layer count etc. All the parameters for each LSTM model are different except the activation function which is set to sigmoid as it is most suitable for binary classification based applications. In execution of each LSTM variant, the best model was saved into a file. Thus, 6 files of different LSTM models are obtained. From each model the confidence rates are obtained for every instance. Hence for every instance 6 confidence rates are obtained, which are used to build a new dataset to train the final

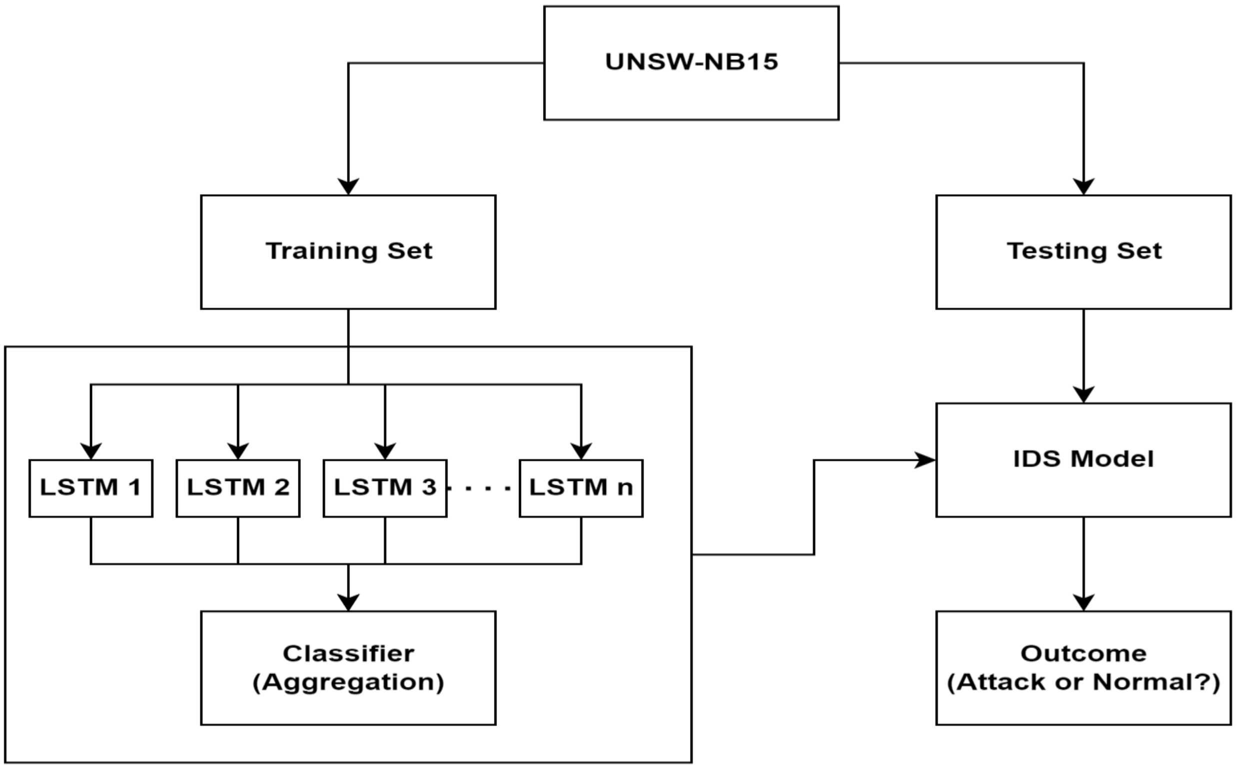

Fig. 5. MLAC Flow Diagram

classifier that aggregates these rates into single outcome, the test dataset is then used to fit all the 6 models and obtain the confidence rates. Again the confidence rates of instances in test dataset are combined to form dataset and final outcome is predicted based on it using classification model. The final outcome is binary in nature i.e. it specifies whether certain network request is a normal traffic or is attempting an attack to exploit the network.

VI. RESULTS AND DISCUSSIONS

The MLAC approach, which is based on LSTM models and a Random Forest classifier, was evaluated on the benchmark network security dataset UNSW-NB15. The trials are run on a Windows 10 machine with 16 GB of RAM and 8 Core i9 CPUs. The outcomes of the models are compared. Python 3.7 is used to write all scripts for data extraction as well as learning activities. Tensorflow4 serves as deep learning platform. Before analyzing the results of LSTM and the MLAC technique, four classifiers were applied to the original dataset, that is: 1) DT; 2) KNN where k=3; 3) Multinomial; and 4) BernoulliNB. Scikitlearn is used to implement these classification techniques. "Table 4" illustrates the experiment outcomes.

Table -IV: Comparison with ML algorithms

Classifier	CA (%)	Precision (%)	Recall (%)	F1-Score (%)
Decision Tree	86.23	82.34	95.46	88.42
KNN	78.51	74.61	92.42	82.57
MultinomialNB	75.23	85.77	65.95	74.56
BernoulliNB	74.88	84.92	66.11	74.35
MLAC	**89.40**	**89.98**	95.14	**92.49**

A. LSTM

In the initial stage of research, a set of LSTMs were trained and tested to see how well they detect IoT cyber intrusions using network traffic. Six distinct LSTMs models with varying architectures are trained for the MLAC approach. The epoch for deep learning training is set as 100 and batch size is set as 32.

According to the findings, the MLAC method surpasses the classifiers listed in the "Table-IV." Furthermore, the performance of various LSTMs varies with batch size. It is also worth to note that the most accurate model, LSTM Model1, has an accuracy rate of 86.61% (for batch size = 32).

B. MLAC Approach

Similarly to the prior part, the effectiveness of the MLAC model is tested, while a Random Forest aggregates the result of LSTMs. The best accuracy rate achieved is 89.40 %. Precision is obtained around 89.98 %. In terms of F1-Score, the MLAC technique achieves 92.49 %. It can be shown that the MLAC approach surpasses single LSTMs and the other classification algorithms listed in the "Table-V" in terms of high performance. The table compares the accuracy of the MLAC technique to that of many other methods proposed in the literature. Evaluation parameters include accuracy rate, precision, recall, and F1-Score.

Table-V: Comparison with other approaches

Approach Name	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)
L Regression	79.59	73.32	98.94	84.22
SVM	62.42	60.91	88.58	71.18
KNN	83.18	79.15	94.30	86.06
DT	88.13	83.91	96.47	90.00
ANN	86.71	81.54	98.06	89.04
AlexNet	73.89	79.61	72.73	76.01
CNN-BiLSTM	77.16	82.63	79.91	81.25
CNN	74.61	81.01	75.65	78.24
LeNet-5	71.11	78.87	71.13	74.80
BiLSTM	72.24	79.52	72.43	75.81
MLAC approach	**89.40**	**89.98**	95.14	**92.49**

1) Precision

It not only supports binary classification, but also multiclass classification problems. Precision is basically the quantification of correctly classified instances in testing phase. To calculate it, we take the ratio of True Positives (T.P.) and addition of False Positives (F.P.) and True Positives (T.P.).

$$Precision = \frac{T.P.}{(F.P. + \text{T.P.})}$$

2) Recall

It measure the correctness of identified True Positives i.e. how accurately the learning model is able to classify the relevant data. It is alternatively called as Sensitivity or True Positive Rate(TPR). It is calculated using False Negatives and True Positives (T.P.).

$$Recall = \frac{T.P.}{(F.P. + T.N.)}$$

3) Accuracy

It is the simplest and most accepted metric to measure the efficiency of model. It uses one additional value than Recall and Precision namely, True Negatives (TN). To calculate accuracy formula stated below is used.

$$Accuracy = \frac{T.N. + T.P.}{(T.P. + F.N. + F.P. + T.N.)}$$

4) F1-Score

F1-Score is a balanced evaluation metric that is also called as F-Measure. It combines recall and precision into a single metric that captures both features. It is obtained by calculating the harmonic mean (H.M.) of recall and precision.

$$F1 = 2.\frac{T.P.}{\frac{1}{2}(F.N.+F.P.)+T.P.}$$

Because it is mean of both metrics recall and precision, it indicates that recall and precision are given equal consideration:

If both recall and precision are higher, a model will have high F1 score. If both recall and precision are poor, a model will have a lower F1 score. If one of recall and precision is lower and other is higher, a model will receive a moderate F1 score.

VII. CONCLUSION

The MLAC technique aims to harness the strengths of deep learning and machine learning technologies in order to build a robust security solution for fog networks. To ensure the maximum benefit from deep learning part, ensembles of LSTM models are implemented and their results are aggregated to obtain better robustness. UNSW-NB15 network security dataset is used for effectiveness evaluation of MLAC model. Since the system is intended to be deployed on the fog layer, the system's complexity is kept as minimal as possible considering the relatively lower computation capability. As a result, the MLAC model addresses both the demand for efficiency and decreased complexity.

More optimizations to the system, such as feature selection or advanced DL and ML classification algorithms, will be explored in the future to extend the work and improve its efficiency even more.

REFERENCES

1. SenguptaJ. Sushmita R. and Das S. B. A comprehensive survey on attacks, security issues and blockchain solutions for iot and iiot. Journal of Net. and Comp. App., 149:102481, 2020. [CrossRef]
2. GhofraneFersi. Fog computing and iot in one building block: a survey and an overview of interacting techs. The Cluster Computing, 24, 12 2021. [CrossRef]
3. B. Flavio and Milito R. Fog computing and its role in the iot. The Proceedings of the MCC workshop on Mobile Cloud Computing, 08 2012.
4. L. Vaquero and L.Rodero M. Finding your way in the fog: Towards comprehensive def. of fog comp. The HP Laboratories Technical Report, 44, 01 2014. [CrossRef]
5. Lei S. Choudhary N. Kumar V. Maglaras L. MukherjeeM., MatamR. and Ferrag M. A. Sec. and privacy in fog comp.: Challenges. IEEE Acc, PP, 09 2017.
6. Dr.WangandDr.BasarSmys S. Hybrid ids for iot. J. of ISMAC, 2:190–199, 09 2020. [CrossRef]
7. Pahl M-O, Aubet F-X All eyes on you: distributed multi dimensional IoT microservice anomaly detection, pp 72–80. IEEE
8. Pajouh HH, Dastghaibyfard G, Hashemi S Two-tier network anomaly detection model: a machine learning approach. J Intell Inf Syst 48(1):61–74 [CrossRef]
9. Moustafa N, Slay J The evaluation of network anomaly detec tion systems: statistical analysis of the UNSW-NB15 data set and the comparison with the kdd99 data set. Inf Secur J Glob Perspect 25(1–3):18–31 [CrossRef]
10. Pajouh HH, Javidan R, Khayami R, Ali D, Choo KKR A two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. IEEE Transactions on Emerging Topics in Computing
11. Kumar V, Das AK, Sinha D UIDS: a unifed intrusion detection system for IoT environment. Evol Intell. https://doi.org/10.1007/s12065-019-00291-w
12. Prabhat Kumar, Gupta, G.P. & Tripathi, R. Design of Anomaly-Based Intrusion Detection System Using Fog Computing for IoT Network. Aut. Control Comp. Sci. 55, 137–147 (2021). [CrossRef]
13. Diro AA, Chilamkurti N (2018) Distributed attack detection scheme using deep learning approach for internet of things. Future Gener Comput Syst 82:761–768 [CrossRef]
14. Papamartzivanos D, Mármol FG, Kambourakis G Dendron: genetic trees driven rule induction for network intrusion detection systems. Future Gener Comput Syst 79:558–574 [CrossRef]
15. Prabavathy S, Sundarakantham K, Shalinie SM Design of cognitive fog computing for intrusion detection in internet of things. J Commun Netw 20(3):291–298 [CrossRef]
16. Kumar, Prabhat & Gupta, Govind & Tripathi, Rakesh. . 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.
17. Kasongo, S.M., Sun, Y. Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. J Big Data 7, 105 (2020). [CrossRef]
18. Khan FA, Gumaei A, Derhab A, Hussain A A novel two-stage deep learning model for efcient network intrusion detection. IEEE Access 7:30373–30385 [CrossRef]
19. K. Jiang, W. Wang, A. Wang and H. Wu, "Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network," in IEEE Access, vol. 8, pp. 32464-32476, 2020, doi: 10.1109/ACCESS.2020.2973730. [CrossRef]
20. Hasan, Mahmudul & Islam, Md & Islam, Ishrak & Hashem, M.M.A. (2019). Attack and Anomaly Detection in IoT Sensors in IoT Sites Using Machine Learning Approaches. 100059. 10.1016/j.iot.2019.100059. [CrossRef]

AUTHOR PROFILE

Akshata Deshmukh, received the B. Tech degree in Computer Science and Engineering from Amravati University, India in Summer 2017. She is pursuing Master in Technology in Computer Engineering stream from College of Engineering Pune She is interested in Deep Learning, Cloud Computing and Cyber Security. She is member of IEEE student chapter since 2011.

Tanuja Pattanshetti, is currently working as assistant professor at College of Engineering, Pune, India. She instructs in areas, including big data, software engineering, and cloud computing. Her research interests include Big Data, Data preprocessing frameworks, Software engineering, and Machine Learning. Various research works have been published by her in the research domains like Data preprocessing and Big data. She is a member of Indian Society for Technical Education (ISTE).