Enhancing Cloud Security: A Novel Intrusion Detection System Using Deep Learning Algorithms

Vijay Kumar Gandam
Department of Computer science and Technology,
Research Scholar
Chaitanya Deemed to be University
Hyderabad, India

E. Aravind, PhD
Department of Computer science and Technology,
Research Supervisor
Chaitanya Deemed to be University
Hyderabad, India

ABSTRACT
The intrinsic qualities of Cloud Computing (CC), including scalability and adaptability, have led to its adoption by several sectors. Nevertheless, cloud providers continue to face substantial challenges related to security, even though these benefits are undeniable. Unauthorized entrée, data breaches, and insider threats are some of the new dangers that CC introduces. Attackers find cloud systems appealing due to their common infrastructure. Tackling these security concerns requires the inclusion of strong security systems. Intrusion Detection Systems (IDS) are one such method that is essential for protecting cloud environments and networks. IDS keep tabs on every system and network activity. A lot of people have been looking at ways to improve IDS performance using ML and DL techniques as of late. Machine learning and deep learning algorithms have proven themselves capable of sifting through mountains of data and producing reliable forecasts. Using these methods, IDS can adjust to new threats, find past attacks, and cut down on false positives. This paper presents a new intrusion detection system (IDS) model that incorporates DL methods such as the Morlet Wavelet Kernel Function. An MLSTM classifier is suggested for the purpose of identifying breaches in the IoT-Cloud setting. Jarratt-Butterfly optimization algorithm (JBOA) selects the relevant features to increase classification accuracy. The suggested model is tested using known methodologies in terms of various parameters using the comprehensive intrusion dataset BoT-IoT. Through the use of simulations, the results prove that the suggested research classical outperforms the state-of-the-art models.

Keywords
Intrusion Detection System; Morlet Wavelet Kernel Function Long Short-Term Memory; Jarratt-Butterfly optimization algorithm; Intrusion Detection System; Cloud Computing; Deep Learning.

1. INTRODUCTION
Because it's a cheap approach to set up and run their own system resources, Cloud computing is now used by many enterprises and startups [1]. Cloud computing has a number of challenges, including location awareness, low latency, geo-location, and mobility support. Cloud model for delivering computing services and applications over the Internet; it allows for increased mobility, flexibility, location service awareness, and low latency [2-3]. Cloud computing encounters numerous safety and security concerns as a result of its implementation in various locations with inadequate security measures. For instance, smart devices can be subject to numerous cyber-attacks that compromise their data privacy, including man-in-the-middle and port scan attacks [4]. The proliferation of internet-enabled gadgets is a direct result of the pervasiveness of the internet in contemporary life. One example is the increasing prevalence of internet of things (IoT) gadgets in people's everyday lives. However, a number of researchers are discussing potential solutions to these growing difficulties [5-6]. Finding, verifying, and stopping unofficial access to a computer network or internetwork is the job of intrusion detection, a technique utilized in cloud and IoT security measures.

Due to the tremendous improvements in data skill, there are major disputes over network confidentiality that need to be handled. IDS are so crucial for protecting networks [7]. Intrusion detection systems are grouped into various distinct methods. Active and inactive are the two primary groups. Newly emerging threats are insurmountable for traditional active IDS systems [8]. Due to the large sum of components and characteristics of this sort of network, one of the primary challenges in detecting intrusions is finding and distinguishing between normal and abnormal connections. Locating and determining the method of intrusions is a common use case for IDS [9]. In order to accomplish intrusion detection in real-time, the researchers looked into various element selection methodologies in depth [10]. A compelling argument for educating the efficiency and precision of classification algorithms is to decrease the sum of features by selecting only the most crucial ones.

It is common practice to employ machine learning algorithms for attack detection; these algorithms also guide network managers toward the best course of action when responding to attacks [11]. Nevertheless, the majority of these conventional ML approaches require a comprehensive feature extraction and selection procedure, and they belong to the shallow learning class [12]. Due to the large sum of components and characteristics of this sort of network, one of the primary challenges in detecting intrusions is finding and distinguishing between normal and abnormal connections. When an intrusion occurs, IDS are often employed to find out where and how it happened [13]. The classifier is the heart of an IDS; it uses a detection algorithm to tell the difference between normal and intrusion-related activity. In networks of cloud, where there are many devices, it can be very difficult to implement a classifier with an accurate discovery technique [14].

In recent times, scholarly investigations have demonstrated that intelligent learning techniques like ML, learning may accomplish network security tasks and have multiple practical applications [15]. To sum up, current procedures still have some flaws that need fixing, even if numerous NN-based intrusion detection methods have been suggested recently and boast about achieving a high performance rate [16]:

- While past studies dealt with DDoS attacks in the cloud, our suggested model is capable of handling any kind of attack.
- The Host-based IDS is the target of the majority of the published approaches. We prefer Network-based IDS over Host-based IDS due of their faster reaction time. In addition, network-based IDS can monitor a whole network segment, OS notwithstanding, without requiring any changes to the current infrastructure.
- Traditional feature selection approaches, such as wrapper methods, are used by most of the presented

methods. A classifier that isn't as sensitive causes inaccurate detection since traditional feature selection methods miss a number of sensitive features. Instead, we employ filter methods that are simpler, take up less room, and are far quicker.

The main contribution of the research work includes:

- With the use of a DL-based ID framework, specifically MLSTM, to detect intrusions in the IoT during cloud-based IoT data broadcast.
- To propose an efficient and optimal mechanism for parameter assortment with the help of JBOA to lessen the presentation of the classification.
- The experimental analysis shows that the proposed model is tested with publicly available dataset.

The related works is untaken in Section 2, the projected perfect is explained in Section 3. The validation of projected model with existing procedures by using dataset is given in Section 4. Finally, the conclusion is obtainable in Section 5.

2. RELATED WORKS

In terms of convergence, security between different IoT devices, and communication speed, this study presents the (BABCN) procedure for intrusion detection, which was proposed by Laassar, I., et al., [17] and uses binary networks. The BABCN method's foundational depth-first search structure equations enhance the artificial bee. The results obtained NSL-KDD dataset show that the projected method improves classification and has a decent capacity to intrusions.

By analyzing the sequence of system calls, Chaudhari, A., et al. [18] present a new intrusion detection framework that can identify both known and undiscovered threats. Using a combination of LSTM and anomaly detection methods based on system call frequency, the framework examines the system call arrangements of virtual machines. We test the suggested architecture on the ADFA-LD dataset, which stands for the Australian Defence Force Academy-Linux Dataset. We acquired the maximum accuracy of 97.2% and the lowest false positive rate of 2.4% using our projected outline when compared to the existing frameworks

A IDS based on a genetic algorithm and multilayer perceptron (MLP) networks is suggested by Ziheng, G. E., and Jiang, G. [19]. To maximize the linkage-related weights and biases, the MLP employs the genetic algorithm. Because of this, it can reliably distinguish between typical and unusual packets of network data. The proposed technique was tested in the Matlab simulator with the KDD cup dataset. According to the consequences. After comparing the proposed method to others, it was shown to be far more accurate. Furthermore, the suggested approach showed excellent specificity and sensitivity in identifying both typical and non-standard packets of network traffic.To prevent the influence of attacks, Polepally, V., et al., [20] create a unique IDS framework employing cloud data. In this case, the incursions are discovered using the spark architecture. In order to remove artifacts and noise from the incoming data, pre-processing is used. Slave nodes then carry out the feature extraction and fusion. The ExpSSA algorithm, which is a proposed method, is used to perform the feature fusion. For effective intrusion detection, the fused characteristics are taken into account in a deep-stacked autoencoder (Deep SAE). Deep SAE is trained using the modified ExpSSA to tune optimal weights. The suggested ExpSSA is a hybrid of the exponential weighted moving average (EWMA) and the squirrel search algorithm (SSA). The suggested ExpSSA-based Deep SAE outperformed competing methods in terms of accuracy, finding rate (0.846), besides false positive rate (FPR).

For the sake of intrusion detection besides secure cloud data storage, Preethi et al. [21] suggested an MDBGRNN-ID-SCESOA, which stands for multi-scale bidirectional gated recurrent neural network with optimal encryption scheme. As a first step in data preparation, we use Domain Transform Filtering (DTF) to tokenize, reduce dimensions, and do semantic analysis using the KDD CUP 99 and DS2OS datasets. In order to separate intrusion data from non-intrusion data, MDBGRNN is then used. In addition, a two-way encryption method that combines Elliptical Curve with the Sine Cosine Egret Swarm Optimization Algorithm (ECC-SCESOA) improves data security with little computing overheads. An effective method of concealing sensitive content is developed through steganography to ensure the security of encrypted data while it is at rest in the cloud. Accuracy, specificity, sensitivity, execution time, memory utilization, and Matthews correlation coefficient (MCC) are some of the performance evaluation measures that show how effective MDBGRNN-ID-SCESOA is. Significant improvements in computing efficiency and data security are revealed by comparing with previous methodologies. Offering a potential path for safeguarding sensitive data in cloud surroundings, this complete solution tackles important security concerns in cloud computing.

A hyper-automation processes in the IIoT is presented by Souri, A., et al., [22] and is based on Trees Detection algorithm. The design is able to predict harmful attacks. Depending on factors such as network traffic, computation time, malicious behaviors, and types of assaults, the suggested architecture employs a priority-based feature approach in conjunction with Analysis of Variance (ANOVA) to determine the most suitable features. The next step is to run experiments with the technical data sets NSL-KDD and UNSW-NB15. Optimisation of large-scale cyber-attack systems for key hyper-automation processes in an IIoT context is effectively achieved by the suggested design, according to the simulation findings, which outperform existing case studies and prediction models.

An strategy to improve cloud intrusion detection using CNN is proposed by Ali, S. Y., et al., [23]. Cloud computing security presented its own set of unique issues, which our deep learning architecture reacted to. The CNN-based intrusion finding scheme demonstrated in this study takes benefit of the network's capacity to autonomously learn hierarchical attributes from raw data, as opposed to conventional IDS schemes that depend on signature- or rule-based approaches. Gathering broad and representative information from cloud systems, including both normal network traffic and other forms of attacks, is an important part of this study. Using these datasets, the CNN is taught the natural patterns of normal behavior and how to spot anomalies intrusion. The suggested system maintains its adaptability to changing threats by regularly retraining with new data to update its expertise. Extensive tests are used to evaluate the CNN-based intrusion detection scheme, comparing its performance to that of established approaches. The results demonstration that the CNN-based methodology is more IDS methods, suggesting that it could be a good choice for cloud computing intrusion detection systems.

3. PROPOSED METHODOLOGY

An IoT network detection approach that is automated is presented in this work. Flow data acquired by sensors is fed into feature engineering algorithm techniques in our suggested model. Methods from feature engineering, including feature imbalance and feature selection, will be active. Recursive Feature Elimination besides Principal Component Analysis are two feature selection methods that can improve model accuracy, decrease training time, and eliminate overfitting, among other data concerns. To find out how each deep learning model performs and how long it takes to

run, we will run a number of them. Here is the study work flow diagrammed in Figure 1.

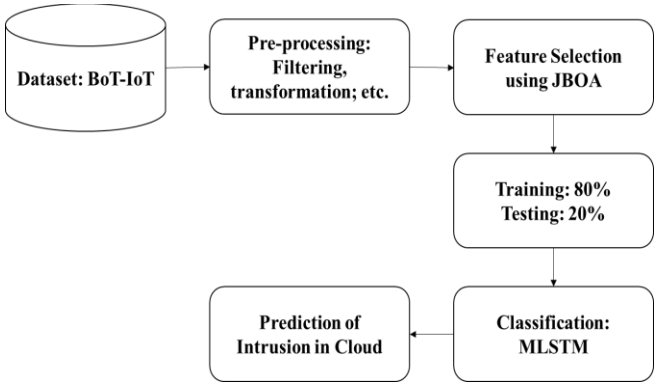

Figure 1: Workflow of the Research Work

a. Description of Bot-IoT Dataset

A fresh set of development data In the experiment, Bot-IoT is utilized to identify simulated assaults through the use of the IoT network [24]. Information gathered from the IoT, the Cyber Range Lab at UNSW Canberra, regular traffic patterns, and botnet traffic patterns induced by different kinds of attacks are all part of the collection. To generate a useful dataset with detailed traffic statistics, a realistic testbed was utilized. The presentation of the machine learning representations was enhanced by adding and labeling additional features. The extraction of features was supported by three subcomponents: investigative analysis, networking structure, and simulated IoT services.

The Internet of Things system may collect weather data in real-time and use it to change the settings. While one smart device controls the lights, another provides information about the fridge's temperature and cooling system. When they sense movement, these lights will automatically switch on. A smart thermostat that can change the temperature on its own and an Internet of Things smart door that takes probabilistic input are also on the list. Characteristics of attacks on the data are detailed in Table 1.

With the use of targets, an IoT system may better categorize network data as either safe or harmful, facilitating the separation of the two. The BoT-IoT dataset aims to capture the following types of data:

i. The benign category includes typical, lawful, and non-malicious Internet of Things (IoT) network operations.

ii. A attacks using TCP can make a network unavailable to legitimate users by overwhelming it with requests;

iii. DDoS that are focused on UDP: these overwhelm networks with service outages.

iv. DDoS attacks that are HTTP-based: these overwhelm web servers with requests, causing them to become unresponsive or unavailable.

v. Attacks that are TCP-based: these take advantage of vulnerabilities in the TCP stack to exhaust network and device resources. - Attacks that are focused on UDP: these overwhelm targets with packets, causing disruptions outages.

vi. Attacks that are HTTP-based: these overwhelm web requests, causing them to become unresponsive or unavailable.

vii. Keylogging: secretly keeping track of keystrokes on an infected device, with the intention of stealing sensitive information.

viii. Data capture: illegally taking data from infected Internet of Things networks or devices.

.

Target	Category	Count
Benign	BENIGN	9654
Attack	DDoS TCP	19,547,60
Attack	DDoS UDP	18,965,10
Attack	DDoS HTTP	19,71
Attack	DoS-TCP	12,35,897
Attack	DoS UDP	20,69,491
Attack	DoS HTTP	29,607
Keylogging	Key logging	109
Data theft	Data -theft	118
-	Total	73,370,443

Table 1. Bot-IoT dataset

b. Data Preprocessing

An integral part of building models is the pre-processing of data. In order to progress the suggested model, we used the following pre-processing methods throughout the process. Data cleansing includes data filtering, data conversion, and missing data checks in the pre-processing phase. Data filtration involves finding and removing null and duplicate values. One step in data transformation is format conversion, which may involve going from a categorial to a numerical format, among others. Data can be cleaned up and made ready for analysis with the help of several Python programs [25].

c. Feature Selection using JBOA

For nonlinear equations, one of the significant enhancements to Newton's approach is Jarratt's method, which may be expressed as:

$$\begin{cases} y_n = x_n - \frac{2f(x_n)}{3f'(x_n)} \\ x_{n+1} = x_n - \left(\frac{3f'(y_n)+f'(x_n)}{6f'(y_n)-2f'(x_n)}\right)\frac{f(x_n)}{f'(x_n)} \end{cases} \quad (1)$$

An integral part of building models is the pre-processing of data. In order to progress the suggested model, we used the following pre-processing methods throughout the process. Data cleansing includes data filtering, data conversion, and missing data checks in the pre-processing phase. Data filtration involves finding and removing null and duplicate values. One step in data transformation is format conversion, which may involve going from a categorial to a numerical format, among others.

Data can be cleaned up and made ready for analysis with the help of several Python programs [25].

that is $(x_n), f'(x_n)$, and $f'(y_n)$. Thus, Jarratt's method because its equals $2^{3-1} = 4$, and hence it is optimal. In addition, Jarratt's

method has been the subject of a great deal of research, with numerous proposals for enhancements.

With a fourth- method approaches four significant digits, or the number of correct decimals after each iteration, and hence multiplies by four. This nonlinear equation example should help to illustrate the concept of Jarratt's technique. Think about $f(x) = \cos(x) - x$. The equation is a=0:739085133215. We set the initial solution x_0=1:7.

As table, the approximation of the root advances by a factor of four with each repetition, until approaching the precise root. But, there are a few drawbacks to Jarratt's method that are common to iterative approaches: problems with divergence, local optima trapping, and beginning value selection

d. Butterfly optimization algorithm (BOA)

The scent and texture of each fragrance in BOA is unique. The BOA stands apart from other metaheuristic algorithms due to its smell, which is determined in the following way::

$$f = cI^a (2)$$

where f reflects the intensity of the scent, which in turn indicates how other butterflies rate the scents, Among the several sensory modalities that distinguish odor, "c" stands for fragrance. Parameter a's value is related to the butterfly's aroma. We can pretend that all butterflies have the same scent if we suppose that a=1. Since each butterfly has the same threshold for olfactory perception, there can be no aroma absorption. So, it's easy to get to one optimal solution, which is typically the global one. On the other hand, when a=0, no other butterfly will be able to detect the scent that one butterfly is producing.

For optimal solution finding, the BOA algorithm mimics the flight patterns of butterflies, which are characterized by the following essential features:

1. Butterflies can entice one another through the scent they release.
2. The butterflies will either flit about at random or converge around the one with the strongest scent.
3. The number of stimuli a butterfly is exposed to is pretentious by the goal function.

All metaheuristic algorithms have three stages: initialization, iteration, and finalization. BOA is no exception. At the outset, the algorithm specifies the goal space of possible solutions. Furthermore, the values of BOA parameters are also assigned. After that, in order to optimize, the algorithm generates a starting population of butterflies. Since the quantity of butterflies does not fluctuate during the BOA fixed memory size is assigned to them to store their data. Iteration is the following process in BOA. The algorithm is iterated several times. At each iteration, the fitness value of every butterfly in the key space is calculated. At their respective sites, the butterflies produce aromas according to Equation (2). The algorithm can toggle between a global search and a local search. As part of their global search, butterflies aim to land on the optimal solution, or butterfly with the highest fitness value. One way to express the global search equation is as (3):

$$x_i^{t+1} = X_i^t + (r^2 \times g - x_i^t) \times f_i (3)$$

where X_i^t represents the key vector x_i of butterfly in repetition t, while g* is the greatest solution for the current repetition. f_i characterizes the butterfly, and r is a random sum between 0 besides 1.

$$x_i^{t+1} = X_i^t + \left(r^2 \times x_j^t - x_k^t\right) \times f_i (4)$$

where X_i^t and x_j^t are butterflies space. Thus, Equation (24) performs a local walk.

BOA uses the probability value p, which is typically among 0 and 1, to toggle between common global search and intense local search. The iteration phase continues until a termination criterion is satisfied. Nevertheless, there are a few methods that the standards are established; for example, by computing CPU time, going beyond a particular number of iterations, or attaining a particular error rate. In the last step, we choose the option that maximizes fitness.

e. Jarratt-Butterfly optimization algorithm (JBOA)

BOA is an effective optimization procedure that has found use in many different contexts. But the NFL theorem states that no procedure can solve every problem perfectly. In addition, when solving NSE, BOA could get stuck at local optima or encounter divergence issues. Thus, JBOA's presentation in solving the optimal solution for BGRU has enhanced by integrating BOA and Jarratt's methods.

Each new version of BOA uses Jarratt's technique. The first step is to consider the BOA's top-ranked spot for the butterfly as a potential site. In most circumstances, the Jarratt approach enhances the butterfly location by using the candidate site as input. In most cases, the Jarratt approach enhances butterfly location and is fed the candidate site. At last, the results from Jarratt's methodologies are evaluated against the potential locations, and the most fit one is selected. Because of convergence, Jarratt's approach can generate more precise solutions with less repetition. Consequently, JBOA's search strategy for solving NSE is enhanced. At the end of each iteration, JBOA implements the alterations noted in the red box1. The position of Jarratt's method (X_(n+1)) and the location of the BOA butterfly x_bj^(t.) are compared using fitness values. In the end, the best solution is the one that finds the appropriate position based on fitness measurements.

4. CLASSIFICATION USING MLSTM

The features that were selected are then input into LSTM in order to determine the type of incursion. One DL design that uses an artificial recurrent neural network (RNN) to represent time series data is Long Short-Term Memory (LSTM). LSTM features feedback connections amid hidden units linked to discrete time steps, unlike traditional feed-forward neural networks. This allows for the learning of long-term sequence dependencies and the prediction of a transaction label from a series of past transactions. In order to address the problem of vanishing and exploding gradients that might arise during the training of conventional RNNs, LSTMs were created. The construction of the LSTM unit is shown in Fig. 2.

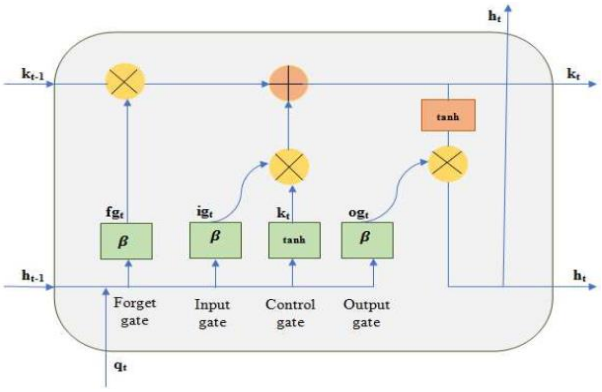

Figure 2: Construction of LSTM.

Each LSTM layer is made up gates: an input gate (ig_t), forget gate (fg_t), and an output gate ((og_t)). The three gates regulate the incoming and outgoing data flows into and out of the cell, which stores the values of random time intervals. Conversely, traditional LSTM training methods suffer from a slower convergence issue and fail to find the error function due to gradient descent becoming stuck in local minima with bias and weight selection that is random. A derivative-free approach to optimizing complicated problems is the Morlet Wavelet Kernel Function (MWKF). Traditional LSTM uses a MWKF in place of the original LSTM's random selection of gate weights and biases. When compared to basic LSTM, this integration performs better for data categorization and non-linear function approximation. MWKFLSTM is the name given to this MWKF-based weight and bias assortment in LSTM. The equations for ig_t, fg_t ,and og_t are given below:

$$fg_t = \beta(w_{fg}(h_{t-1}, q_t) + b_{fg}) \quad (5)$$

$$ig_t = \beta(w_{ig}(h_{t-1}, q_t) + b_{ig}) \quad (6)$$

$$og_t = \beta(w_{og}(h_{t-1}, q_t) + b_{og}) \quad (7)$$

Where β indicates the sigmoid function. The terms w_{fg},w_{ig},and w_{og} are the weight matrices of f_{g_t},i_{g_t} ,and o_{g_t} respectively. The terms b_{fg} , b_{ig} ,and b_{og} are the bias matrices of f_{g_t} , i_{g_t} and o_{g_t}correspondingly. The preceding output at $(t-1)^{th}$ timestamp is represented as h_{t-1} , andqt characterizes the present input vector at time stamp t. Every LSTM gate has a weight and bias value between zero and n-1 that are randomly generated. To improve the classifier's detection performance, we use MWKF to find the best fit weight and bias values for the LSTM network, rather than picking them at random. The expression for the wavelet basis function can be given by taking a function, H(w), with a scale factor of z and a translation factor of x.

$$\beta_{z,x}(w) = \frac{1}{\sqrt{|z|}}\psi\left(\frac{w-x}{z}\right) \quad (8)$$

To represent a multidimensional wavelet purpose, tensor product theory states that one can take many functions besides multiply them together.

$$\psi(w) = \prod_{i=1}^{n}\psi(w_i) \quad (9)$$

Equation (9), in where n is the sum of functions, and w_i is variable of the i-th one- function, allows one to design a kernel function.

$$K(w, \bar{w}) = \prod_{i=1}^{n}\beta\left(\frac{w_i - \bar{w}_i}{z}\right) \quad (10)$$

This study uses the MVKF to build the function. This is the MVKF function:

$$\beta(w) = \cos(1.75w)exp\left(-\frac{w^2}{2}\right) \quad (11)$$

The WK purpose accumulated since the MVKF

$$K(w, \bar{w}) = \prod_{i=1}^{n}\left[cos\left(1.75 \times \frac{w_i - \bar{w}_i}{z}\right)exp\left(-\frac{(w_i - \bar{w}_i)^2}{2z^2}\right)\right] \quad (12)$$

Where $w_i$describes the standards of the three gates' weights. The approximation qualities of the wavelet function over to the MWKF by virtue of its construction utilizing the wavelet function's kernel. Therefore, LSTM will perform better in classification tasks when a MW function is used as the kernel function. Each of the three gates' weight values undergoes the aforementioned kernel computations. The calculation for the bias values gates is also executed in the same manner. In addition to the output, the terms for the cell state and applicant cell state are

$$\mathcal{K}_t{}^{\omega} = \tanh(w_k[h_{t-1}, q_t] + b_k) \quad (13)$$

$$k_t = f_{g_t} * k_{t-1} + ig_t * k_t \quad (14)$$

$$h_t = og_t * \tanh(k_t) \quad (15)$$

Where k_t besides k_{t-1} represents the new and preceding cell states of $t$and alsot $-$ 1. The term $\mathcal{K}_t{}^{\omega}$characterizes candidate cell public at t, and *symbolizes the multiplication of vectors.

5. RESULTS AND DISCUSSION

This experiment was carried out using a DELL laptop running Windows 10 with 16 GB of RAM besides an Intel Core i5-10210U processor in order to apply and assess the suggested method on the dataset under consideration. Several libraries, including matplotlib (version 3.3.2), numpy (version 1.19.2), pandas (version 1.1.3), scikit-learn (version 0.23.2), keras (version 2.6.0), besides tensor flow (version 2.6.0), are implemented using Spyder Python (version 3.8).

We use the following metrics to measure the efficacy of the projected system: Accuracy, Precision, Recall, F1-Score, True Positive Degree, and False Positive Degree. The ratio of correctly sum of records or counts is used to measure accuracy, as shown in Equation (16):

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \quad (16)$$

The accuracy rate of abnormal instance predictions relative to the total sum of abnormal instance predictions is called precision, and it is calculated using Equation (17):

$$Precision = TP/TP + FP \quad (17)$$

According to Equation (18), recall is the proportion of correctly estimated abnormal cases to the entire sum of actual abnormal instances:

$Recall = TP/TP + FN$ (18)

According to Equation (19), the F1 Score provides a Precision besides Recall for evaluating the scheme's accuracy:

$F1Score = 2((Precision * Recall)/(Precision + Recall))$ (19)

f. Validation Analysis of Proposed Model

Table 2 offers the investigational study of proposed classicalwith existing techniques in terms of diverse metrics.

Method	ACC	Recall	Precision	F1-Score
XGBoost	93.12	94.21	92.31	93.42
DBN	95.75	95.78	93.08	94.30
CNN	**96.51**	96.94	94.29	95.19
RNN	97.61	97.04	95.14	96.51
LSTM	98.44	98.39	97.66	97.93
MLSTM	99.22	99.51	98.48	98.58

Table 2: *Validation study of proposed perfect with existing procedures*

In the proposed perfect with existing procedures, the XGBoost technique accuracy as 93.12 also recall of 94.21 and precision as92.31alsof1-score as 93.42. Then the DBN technique accuracy as 95.75 also recalls of 95.78 and precision as 93.08 also the f1-score as 94.30 correspondingly. Then the CNN technique accuracy as 96.51 also recall of 96.94 and precision as 94.29 also the f1-score as 95.19 correspondingly. Then the RNN technique accuracy as 97.61 also recalls of 97.04 and precision as 95.14 also the f1-score as 96.51 correspondingly. Then the LSTM technique accuracy as 98.44 also recalls of 98.39 and precision as 97.66 also the f1-score as 97.93 correspondingly. Then the MLSTM technique accuracy as 99.22 also recalls of 99.51 and precision as 98.48 also the f1-score as 98.58 correspondingly.

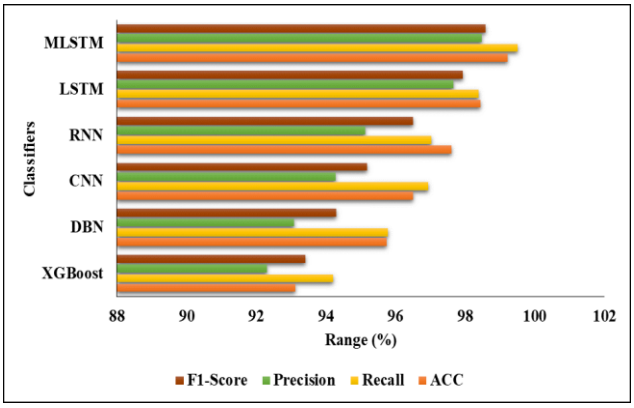

Figure 3: Graphical Description of proposed perfect with existing performances

Algorithm	Training Accurancy	Test Accuracy	Training Time(s)
XGBoost	0.9101	0.9144	244
DBN	0.9319	0.9382	229
CNN	0.9407	0.9423	324
RNN	0.9594	0.9534	251
LSTM	0.9600	0.9653	260
MLSTM	0.9743	0.9777	236

Table 3: Timing Analysis

The XGBoost technique has a training accuracy of 0.9101, a testing accuracy of 0.9144, and a training time of 244 in the timing analysis of various techniques. Subsequently, the DBN technique yielded training accuracy of 0.9319, testing accuracy of 0.9382, and training time of 229 in accordance. Subsequently, the CNN technique yielded training accuracy of 0.9407, testing accuracy of 0.9423, and training time of 324 in accordance. Subsequently, the RNN technique yielded training accuracy of 0.9594, testing accuracy of 0.9534, and a corresponding training time of 251. Next, the training accuracy and testing accuracy of the LSTM technique are 0.9600 and 0.9653, respectively, and the training time is 260. Next, the training accuracy and testing accuracy of the MLSTM technique are 0.9743 and 0.9777, respectively, and the training time is 236 respectively.

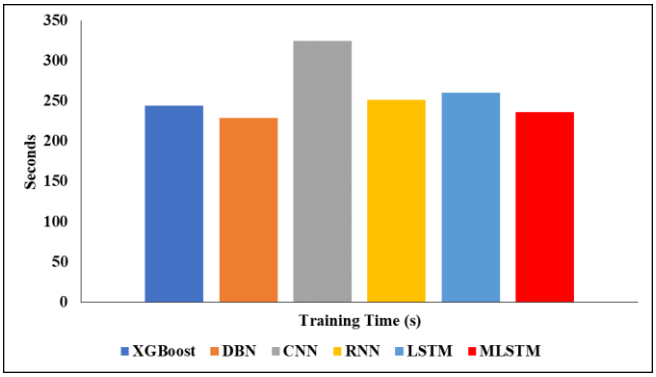

Figure 4: Visual Representation of the proposed model for timing analysis

6. CONCLUSION

In order to detect breaches in the IoT-Cloud infrastructure, this paper proposes a new MLSTM. To help the MLSTM choose features, pre-processing and JBOA are provided. Running simulations on the BoT-IoT dataset allows us to verify the models' efficacy. The MLSTM is able to notice all types of network threats with a better degree of precision. When associated to state-of-the-art methodologies graphed in the literature, MLSTM offers a greater detection accuracy. When compared to other approaches on the same dataset, the model achieves the best throughput ratio, lowest false alarm rate, and shortest delay. The enhanced efficiency of MLSTM's data packet transfer from cloud-based IoT devices is demonstrated by their higher performance. Following normalcy classification, the data was safely stored on the cloud. But, there are convinced restrictions on the system, such as the reliability of the data used to create the input besides output. In a similar vein, the data imbalance is a significant problem with the dataset that needs fixing in the next stage.

7. REFERENCES

[1] Al-Ghuwairi, A. R., Sharrab, Y., Al-Fraihat, D., AlElaimat, M., Alsarhan, A., & Algarni, A. (2023). Intrusion detection in cloud computing based on time series anomalies utilizing machine learning. Journal of Cloud Computing, 12(1), 127.

[2] Mohamed, D., & Ismael, O. (2023). Enhancement of an IoT hybrid intrusion detection system based on fog-to-cloud computing. Journal of Cloud Computing, 12(1), 41.

[3] Samunnisa, K., Kumar, G. S. V., & Madhavi, K. (2023). Intrusion detection system in distributed cloud computing:

Hybrid clustering and classification methods. Measurement: Sensors, 25, 100612.

[4] Attou, H., Guezzaz, A., Benkirane, S., Azrour, M., & Farhaoui, Y. (2023). Cloud-based intrusion detection approach using machine learning techniques. Big Data Mining and Analytics, 6(3), 311-320.

[5] Tariq, M., & Suaib, M. (2023). A review on intrusion detection in cloud computing. International Journal of Engineering and Management Research, 13(2), 207-215.

[6] Kavitha, C., Gadekallu, T. R., K, N., Kavin, B. P., & Lai, W. C. (2023). Filter-based ensemble feature selection and deep learning model for intrusion detection in cloud computing. Electronics, 12(3), 556.

[7] Attou, H., Mohy-eddine, M., Guezzaz, A., Benkirane, S., Azrour, M., Alabdultif, A., & Almusallam, N. (2023). Towards an intelligent intrusion detection system to detect malicious activities in cloud computing. Applied Sciences, 13(17), 9588.

[8] Lin, H., Xue, Q., Feng, J., & Bai, D. (2023). Internet of things intrusion detection model and algorithm based on cloud computing and multi-feature extraction extreme learning machine. Digital Communications and Networks, 9(1), 111-124.

[9] Maheswari, K. G., Siva, C., & Priya, G. N. (2023). An optimal cluster based intrusion detection system for defence against attack in web and cloud computing environments. Wireless Personal Communications, 128(3), 2011-2037.

[10] Maheswari, K. G., Siva, C., & Nalinipriya, G. (2023). Optimal cluster based feature selection for intrusion detection system in web and cloud computing environment using hybrid teacher learning optimization enables deep recurrent neural network. Computer Communications, 202, 145-153.

[11] Vashishtha, L. K., Singh, A. P., & Chatterjee, K. (2023). HIDM: A hybrid intrusion detection model for cloud based systems. Wireless Personal Communications, 128(4), 2637-2666.

[12] Srilatha, D., & Thillaiarasu, N. (2023). Implementation of Intrusion detection and prevention with Deep Learning in Cloud Computing. Journal of Information Technology Management, 15(Special Issue), 1-18.

[13] Alzughaibi, S., & El Khediri, S. (2023). A cloud intrusion detection systems based on dnn using backpropagation and pso on the cse-cic-ids2018 dataset. Applied Sciences, 13(4), 2276.

[14] Alheeti, K. M. A., Lateef, A. A. A., Alzahrani, A., Imran, A., & Al_Dosary, D. (2023). Cloud Intrusion Detection System Based on SVM. International Journal of Interactive Mobile Technologies, 17(11).

[15] Bingu, R., & Jothilakshmi, S. (2023). Design of intrusion detection system using ensemble learning technique in cloud computing environment. International Journal of Advanced Computer Science and Applications, 14(5).

[16] Wankhade, N., & Khandare, A. (2023). Optimization of deep generative intrusion detection system for cloud computing: challenges and scope for improvements. EAI Endorsed Transactions on Scalable Information Systems, 10(6).

[17] Laassar, I., Hadi, M. Y., Arifullah, H. R., & Khan, F. S. (2024). Proposed algorithm base optimisation plan for feature selection-based intrusion detection in cloud computing. Indonesian Journal of Electrical Engineering and Computer Science, 33(2), 1140-1149.

[18] Chaudhari, A., Gohil, B., & Rao, U. P. (2024). A novel hybrid framework for cloud intrusion detection system using system call sequence analysis. Cluster Computing, 27(3), 3753-3769.

[19] Ziheng, G. E., & Jiang, G. A Novel Intrusion Detection Mechanism in Cloud Computing Environments based on Artificial Neural Network and Genetic Algorithm. Telecommunications and Radio Engineering.

[20] Polepally, V., Jagannadha Rao, D. B., Kalpana, P., & Nagendra Prabhu, S. (2024). Exponential Squirrel Search Algorithm-Based Deep Classifier for Intrusion Detection in Cloud Computing with Big Data Assisted Spark Framework. Cybernetics and Systems, 55(2), 331-350.

[21] Preethi, B. C., Vasanthi, R., Sugitha, G., & Lakshmi, S. A. (2024). Intrusion detection and secure data storage in the cloud were recommend by a multiscale deep bidirectional gated recurrent neural network. Expert Systems with Applications, 255, 124428.

[22] Souri, A., Norouzi, M., & Alsenani, Y. (2024). A new cloud-based cyber-attack detection architecture for hyper-automation process in industrial internet of things. Cluster Computing, 27(3), 3639-3655.

[23] Ali, S. Y., Farooq, U., Anum, L., Mian, N. A., Asim, M., & Alyas, T. (2024). Securing cloud environments: a Convolutional Neural Network (CNN) approach to intrusion detection system. Journal of Computing & Biomedical Informatics, 6(02), 295-308.

[24] The Bot-Iot Dataset; IEEE: Piscataway, NJ, USA, 2019; Volume 5.

[25] Fan, C.; Chen, M.; Wang, X.; Wang, J.; Huang, B. A Review on Data Preprocessing Techniques toward Efficient and Reliable Knowledge Discovery From Building Operational Data. Front. Energy Res. 2021, 9, 652801