International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Cloud Security Redefined: Intrusion Detection System Powered by Deep Learning

Vijay Kumar Gandam[1], Dr. E. Aravind*[2]

Submitted: 05/11/2024 **Revised:** 15/12/2024 **Accepted:** 23/12/2024

Abstract: Cloud Computing (CC) provides unparalleled scalability and adaptability, making it integral across industries. However, these benefits come with significant security challenges, such as unauthorized access, data breaches, and insider threats. The shared infrastructure of cloud systems is especially attractive to malicious actors. Addressing these vulnerabilities necessitates robust security mechanisms, with Intrusion Detection Systems (IDS) being a key solution. IDS monitor network and system activities to detect potential intrusions. The integration of machine learning (ML) and deep learning (DL) has enabled IDS to adapt to emerging threats while minimizing false alarms. This study proposes an innovative IDS model incorporating the Morlet Wavelet Kernel Function and an MLSTM (Modified Long Short-Term Memory) classifier. The Jarratt-Butterfly Optimization Algorithm (JBOA) is utilized for feature selection to enhance classification accuracy. Tested on the comprehensive BoT-IoT dataset, the model demonstrates superior performance compared to existing techniques.

Keywords: *Cloud Computing, Deep Learning, Intrusion Detection Systems, Jarratt-Butterfly Optimization, Long Short-Term Memory, Morlet Wavelet Kernel.*

1. Introduction

Due to its cost-effectiveness in establishing and managing system resources, cloud computing has become widely adopted by businesses and startups [1]. Despite its advantages, cloud computing faces several challenges, including mobility support, low latency, geo-location, and location awareness. This model delivers computing services and applications over the Internet, enabling enhanced flexibility, mobility, and service awareness while maintaining low latency [2-3]. However, the deployment of cloud computing across various regions with inconsistent security protocols introduces significant safety and security risks. For example, smart devices are vulnerable to numerous cyber-attacks, such as man-in-the-middle and port scanning attacks, which jeopardize data privacy [4]. The growing prevalence of internet-enabled devices, driven by the widespread integration of the Internet in modern life, has led to a surge in the adoption of Internet of Things (IoT) devices. Despite their benefits, these devices face rising security challenges, prompting researchers to explore potential solutions [5-6]. Intrusion detection plays a critical role in cloud and IoT security by identifying, validating, and preventing unauthorized access to computer networks or systems. With the significant advancements in data technologies, addressing critical disputes over network confidentiality has become essential. Intrusion Detection Systems (IDS) play a vital role in safeguarding networks [7]. These systems are broadly categorized into active and passive methods. However, traditional active IDS often struggle to combat newly emerging threats [8]. One of the key challenges in intrusion detection lies in identifying and differentiating between normal and abnormal connections within complex networks with numerous components and characteristics. IDS are commonly employed to pinpoint and analyze intrusion methods [9]. To achieve real-time intrusion detection, researchers have extensively explored various feature selection methodologies [10]. A strong case exists for enhancing the efficiency and accuracy of classification algorithms by reducing the number of features and focusing solely on the most critical ones.

1 Research Scholar, Department of Computer Science and Engineering, Chaitanya Deemed to be University – 500075, Hyderabad, India
ORCID ID : 0000-0001-8101-2166
**2 Research Supervisor, Department of Computer Science and Engineering, Chaitanya Deemed to be University – 500075, Hyderabad.*
ORCID ID : 0009-0009-8584-0087
** Corresponding Author Email: gandamvijaykumarphd@email.com*

Machine learning algorithms are commonly utilized for detecting attacks and assisting network administrators in determining the most effective response strategies [11]. However, most traditional ML approaches rely on exhaustive feature extraction and selection processes, and they fall under the category of shallow learning [12]. A major challenge in intrusion detection lies in identifying and distinguishing between normal and anomalous network connections, given the complexity and vast number of components in such networks. When an intrusion occurs, IDS are crucial for determining its origin and method [13]. At the core of any IDS is the classifier, which employs detection algorithms to differentiate between normal and intrusion-related activities. In cloud networks, characterized by numerous interconnected devices, developing a classifier with high accuracy for detecting intrusions poses significant difficulties [14].

Recent scholarly researches has shown that advanced learning techniques, such as machine learning (ML), are highly effective in addressing network security challenges and offer a range of practical applications [15]. Despite the introduction of numerous neural network (NN)-based intrusion detection methods that claim to deliver high performance, existing approaches still exhibit certain shortcomings that require resolution [16]:While past studies dealt with DDoS attacks in the cloud, our suggested model is capable of handling any kind of attack.

Most published approaches focus on Host-based IDS. However, we prioritize Network-based IDS due to their superior response times. Network-based IDS can oversee an entire network segment, independent of the operating system, without necessitating modifications to the existing infrastructure.
Traditional feature selection methods, such as wrapper techniques, dominate existing approaches. These methods often result in less sensitive classifiers, leading to inaccurate detection by overlooking critical features. In contrast, we utilize filter-based methods, which are simpler, consume less storage, and operate significantly faster.

The primary contributions of this research are as follows:

- Utilization of a Deep Learning-based Intrusion Detection (ID) framework, specifically MLSTM, to detect intrusions in IoT environments during cloud-based IoT data transmission.
- Proposal of an efficient and optimal mechanism for parameter selection using JBOA to improve the performance of the classification process.
- Experimental analysis demonstrating that the proposed model is validated using publicly available datasets

The related works are discussed in Section 2, the proposed approach is detailed in Section 3, the validation of the proposed model with existing methods using the dataset is provided in Section 4, and the conclusion is presented in Section 5.

2. Related Works

This Paper introduces the Binary Artificial Bee Colony Network (BABCN) procedure for intrusion detection, as proposed by Laassar, I., et al. [17], which utilizes binary networks. The method's depth-first search structure equations enhance the performance of the artificial bee algorithm. Experimental results using the NSL-KDD dataset demonstrate that the proposed method improves classification accuracy and exhibits a strong ability to detect intrusions, particularly in terms of convergence, security between IoT devices, and communication speed.

Chaudhari, A., et al. [18] propose a novel intrusion detection framework that analyzes the sequence of system calls to detect both known and unknown threats. The framework combines Long Short-Term Memory (LSTM) and anomaly detection methods based on system call frequency to examine the system call sequences of virtual machines. The proposed architecture is tested on the ADFA-LD dataset (Australian Defence Force Academy-Linux Dataset). The results demonstrate a maximum accuracy of 97.2% and the lowest false positive rate of 2.4%, outperforming existing frameworks.

Ziheng, G. E., and Jiang, G. [19] propose an Intrusion Detection System (IDS) based on a genetic algorithm and multilayer perceptron (MLP) networks. The MLP uses the genetic algorithm to optimize the linkage-related weights and biases, enabling reliable differentiation between normal and abnormal network data packets. The proposed method was tested in the Matlab simulator using the KDD Cup dataset, showing superior accuracy compared to other methods. The technique also demonstrated excellent specificity and sensitivity in identifying both normal and unusual network traffic packets.

To counteract the effects of attacks, Polepally, V., et al. [20] introduce a novel IDS framework leveraging cloud data and Spark architecture. Pre-processing is used to eliminate artifacts and noise from incoming data, while feature extraction and fusion are performed by slave nodes. The proposed ExpSSA algorithm, a hybrid of the exponential weighted moving average (EWMA) and the squirrel search algorithm (SSA), is used for feature fusion. These fused features are then fed into a deep-stacked autoencoder (Deep SAE) trained using the modified ExpSSA to optimize weights. The ExpSSA-based Deep SAE outperforms existing methods in terms of accuracy, detection rate (0.846), and false positive rate (FPR).

For intrusion detection and secure cloud data storage, Preethi et al. [21] propose the MDBGRNN-ID-SCESOA model, which stands for Multi-Scale Bidirectional Gated Recurrent Neural Network with Optimal Encryption Scheme. The model uses Domain Transform Filtering (DTF) for data preparation, including tokenization, dimensionality reduction, and semantic analysis with KDD CUP 99 and DS2OS datasets. MDBGRNN is employed to distinguish between intrusion and non-intrusion data. A two-way encryption method combining Elliptical Curve and the Sine Cosine Egret Swarm Optimization Algorithm (ECC-SCESOA) enhances data security with minimal computational overhead. Additionally, a steganography-based approach is used to protect encrypted data while it is stored in the cloud. Performance evaluation metrics such as accuracy, specificity, sensitivity, execution time, memory utilization, and Matthews correlation coefficient (MCC) demonstrate significant improvements in computing efficiency and data security, highlighting the effectiveness of MDBGRNN-ID-SCESOA in securing cloud environments.

Souri, A., et al. [22] present a hyper-automation process for Industrial Internet of Things (IIoT) based on a Trees Detection algorithm, designed to predict malicious attacks. The architecture utilizes a priority-based feature selection approach alongside Analysis of Variance (ANOVA) to identify the most relevant features based on network traffic, computation time, malicious behaviors, and attack types. Experiments with NSL-KDD and UNSW-NB15 datasets show that the proposed design effectively optimizes large-scale cyber-attack systems for IIoT processes, outperforming existing models.

3. Proposed Methodology

This work presents an automated IoT network detection approach. In our proposed model, flow data collected from sensors is processed through feature engineering algorithms. The feature engineering methods, such as feature imbalance handling and feature selection, are integrated into the model. Recursive Feature Elimination (RFE) and Principal Component Analysis (PCA) are two feature selection techniques used to enhance model accuracy, reduce training time, and prevent overfitting, among other data-related issues.To evaluate the performance and runtime of each deep learning model, we will run multiple models and analyze their results. The study workflow is illustrated in Figure 1.

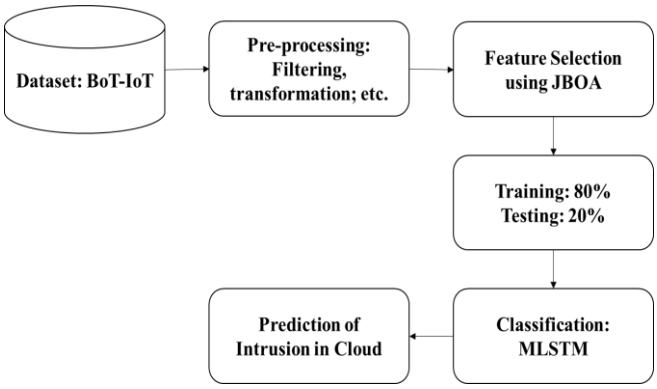

Figure 1: Workflow of the Research Work

a. Description of Bot-IoT Dataset

In this experiment, a fresh set of development data is utilized, with Bot-IoT employed to identify simulated attacks on the IoT network [24]. The data collection consists of information gathered from IoT devices, the Cyber Range Lab at UNSW Canberra, regular traffic patterns, and botnet traffic patterns induced by various attack types. A realistic testbed is used to generate a comprehensive dataset with detailed traffic statistics. To enhance the machine learning model's performance, additional features are added and labeled. Feature extraction is supported by three subcomponents: investigative analysis, network structure, and simulated IoT services.

The IoT system is capable of collecting real-time weather data and adjusting settings accordingly. For instance, one smart device controls the lighting, while another monitors the fridge's temperature and cooling system. These lights automatically turn on when

motion is detected. The system also includes a smart thermostat that adjusts the temperature independently and an IoT-enabled smart door that responds to probabilistic input. The characteristics of attacks on this data are detailed in Table 1.

Using targets, an IoT system can more effectively categorize network data as either safe or harmful, helping to separate normal from malicious traffic. The BoT-IoT dataset is designed to capture the following types of data:

i. The benign category includes typical, lawful, and non-malicious Internet of Things (IoT) network operations, representing regular, safe activity within the IoT network.

ii. TCP-based attacks can render a network unavailable to legitimate users by overwhelming it with excessive requests, causing congestion and disrupting normal communication

iii. UDP-based DDoS attacks overwhelm networks by flooding them with traffic, leading to service outages and disruptions in network availability.

iv. HTTP-based DDoS attacks overwhelm web servers with an excessive number of requests, causing them to become unresponsive or unavailable, effectively disrupting access to the targeted websites or services.

v. TCP-based attacks exploit vulnerabilities in the TCP stack to exhaust network and device resources, potentially leading to system crashes or slowdowns. UDP-based attacks flood the target with excessive packets, causing network congestion and disruptions, resulting in outages and degraded performance.

vi. HTTP-based attacks overwhelm web servers with a high volume of requests, causing them to become unresponsive or unavailable, disrupting access to websites and online services.

vii. Keylogging involves secretly recording keystrokes on an infected device, with the intent of stealing sensitive information such as passwords, personal data, or financial details.

viii. Data capture refers to the illegal collection of data from compromised IoT networks or devices, often with the aim of stealing confidential or proprietary information.

.

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 essential aspect of building models is the pre-processing of data. To enhance the proposed model, the following pre-processing methods were implemented throughout the process:

- Data Cleansing: This involves data filtering, data transformation, and checking for missing values during the pre-processing phase.
- Data Filtering: Identifies and removes null and duplicate values to ensure data consistency.

Data Transformation: Includes format conversions, such as transforming categorical data into numerical formats, among others, to standardize the data for analysis.

These steps ensure that the data is cleaned and prepared for analysis, improving the quality and reliability of the model. Various Python tools and libraries were utilized to perform these pre-processing tasks [25].

c. Feature Selection using JBOA

Jarratt's method is a significant enhancement of Newton's method for solving nonlinear equations. It is a fourth-order iterative method designed to improve convergence efficiency. The method can be expressed mathematically as follows:

Let the nonlinear equation be

$f(x)=0$

f(x)=0.

The iterative steps for Jarratt's method are:

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

An essential step in building effective models is the pre-processing of data, which ensures the data is clean, structured, and ready for analysis. To enhance the proposed model, the following pre-processing methods were applied:

- Data Cleansing: This step focuses on ensuring the dataset is free of errors and inconsistencies:
- Data Filtering: Involves identifying and removing null values and duplicates to improve data quality.
- Missing Data Checks: Ensures that any gaps in the data are addressed to maintain completeness.
- Data Transformation: This process converts the data into formats suitable for analysis:
- Format Conversion: Includes transforming categorical data into numerical formats or other necessary conversions to make the data compatible with machine learning algorithms.

By employing these pre-processing techniques, the data becomes better structured and optimized for training the model, ultimately improving its accuracy and reliability.

Python programs and libraries were instrumental in cleaning and preparing the data for analysis [25].

In addition to data pre-processing, Jarratt's method offers an efficient approach for solving nonlinear equations. This method, due to its convergence rate of 23−1=42^{3-1} = 423−1=4, is considered optimal**.** The process evaluates f(xn)f(x_n)f(xn), f′(xn)f′(x_n)f′(xn), and f′(yn)f′(y_n)f′(yn), resulting in significant advancements in numerical solutions.

Key Features of Jarratt's Method:

- Convergence: The method approaches four significant digits (or correct decimals) per iteration, effectively multiplying accuracy by four with each repetition.
- Application Example: Consider the nonlinear equation f(x)=cos⁡(x)−xf(x) = \cos(x) - xf(x)=cos(x)−x. The root of this equation is approximately a=0.739085133215a = 0.739085133215a=0.739085133215. Using Jarratt's method with an initial guess of x0=1.7x_0 = 1.7x0=1.7, the approximation of the root advances consistently by a factor of four with each iteration until it converges to the exact solution.

Limitations of Jarratt's Method:

Despite its effectiveness, Jarratt's method shares some challenges common to iterative methods:

- Divergence: The method may fail to converge if the initial guess is poorly chosen.
- Local Optima Trapping: It can become stuck in a local optimum rather than finding the global solution.
- Initial Value Sensitivity: The accuracy and success of the method heavily depend on selecting an appropriate starting point.

These limitations have driven extensive research on Jarratt's method, resulting in various proposed enhancements to improve its stability and applicability.

Through its efficiency and widespread applicability, Jarratt's method serves as a powerful tool in solving nonlinear equations, complementing the robust data preparation methods used in model building.

d. Butterfly optimization algorithm (BOA)

The scent and texture of each fragrance in the Butterfly Optimization Algorithm (BOA) are unique, serving as a defining characteristic that sets it apart from other metaheuristic algorithms. In BOA, the "smell" is a key parameter used to guide the search process, and it is determined in the following way:

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

The parameter f represents the intensity of the scent, which indicates how other butterflies perceive and rate the fragrance. Among the various sensory modalities that define odor, c symbolizes the fragrance. The value of parameter a is associated with the butterfly's aroma. If we assume a=1 all butterflies can be considered to have the same scent. In this scenario, as all butterflies share the same threshold for olfactory perception, there is no absorption of the aroma. This simplifies the process, making it easier to converge toward a single

optimal solution, usually the global one. Conversely, when a=0, no other butterfly can detect the scent produced by a specific butterfly.

To achieve an optimal solution, the BOA algorithm replicates the unique flight patterns of butterflies, which are defined by the following key characteristics.

1. Butterflies attract one another through the release of their distinctive scent.
2. They either move randomly or converge around the butterfly emitting the strongest scent.
3. The intensity of stimuli a butterfly perceives is influenced by the objective function.

All metaheuristic algorithms generally follow three stages: initialization, iteration, and finalization, and the Butterfly Optimization Algorithm (BOA) adheres to this structure. Initialization**:** In this stage, the algorithm defines the solution space to explore potential solutions. The parameters of the BOA are also initialized. Subsequently, the algorithm generates an initial population of butterflies. Since the number of butterflies remains constant throughout the algorithm, each butterfly is allocated a fixed memory size to store relevant data. Iteration**:** This phase involves repeated execution of the algorithm. During each iteration, the fitness value of every butterfly in the solution space is computed. Butterflies generate scents at their respective positions based on Equation (2). The algorithm dynamically alternates between global and local searches. In the global search phase, butterflies aim to converge on the optimal solution, which corresponds to the butterfly with the highest fitness value. The global search process can be represented mathematically, as shown in Equation (3).

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

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.

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

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

The Butterfly Optimization Algorithm (BOA) incorporates a probability value p, typically ranging between 0 and 1, to switch dynamically between a broad global search and a focused local search. The iteration phase continues until a predefined termination criterion is met. These criteria can be set in several ways, such as measuring CPU time, reaching a specific number of iterations, or achieving a predefined error threshold.In the final step, the solution that yields the highest fitness value is selected as the optimal outcome. This ensures that the algorithm identifies the most effective solution within the defined parameters.

e. Jarratt-Butterfly optimization algorithm (JBOA)

The Butterfly Optimization Algorithm (BOA) is a versatile optimization method applied in various contexts. However, the "No Free Lunch" (NFL) theorem asserts that no single optimization method can excel at solving every problem. Additionally, when addressing Nonlinear System Equations (NSE), BOA may encounter challenges such as getting stuck in local optima or facing divergence issues. To address these limitations, the JBOA method enhances BOA by integrating it with Jarratt's techniques.

In JBOA, each iteration incorporates Jarratt's method to refine solutions. Initially, the best position identified by BOA for the butterfly is treated as a candidate site. Jarratt's method then optimizes this candidate site, often improving the butterfly's location. The results of Jarratt's method are compared against BOA's candidate positions, and the location with the highest fitness value is chosen.Due to its convergence capabilities, Jarratt's method enables more accurate solutions with fewer iterations, thus improving JBOA's efficiency in solving NSE. At the end of each iteration, JBOA incorporates updates, comparing the fitness of Jarratt's refined position Xn+1 with the BOA butterfly position $xbj(t)$x . Ultimately, the solution that achieves the best fitness value is selected as the optimal outcome. This integration significantly enhances the search strategy and solution accuracy of BOA.

4. Classification using MLSTM

The selected features are then passed into a Long Short-Term Memory (LSTM) model to classify the type of incursion. LSTM is a deep learning architecture based on artificial recurrent neural networks (RNNs) that is particularly adept at handling time-series data. Unlike conventional feed-forward neural networks,

LSTM introduces feedback connections between hidden units across discrete time steps. This architecture enables the model to learn long-term sequence dependencies effectively, making it capable of predicting transaction labels based on a sequence of previous transactions.LSTMs were designed to overcome challenges such as vanishing and exploding gradients, which are common in the training of traditional RNNs.The construction of the LSTM unit is shown in Fig. 2.

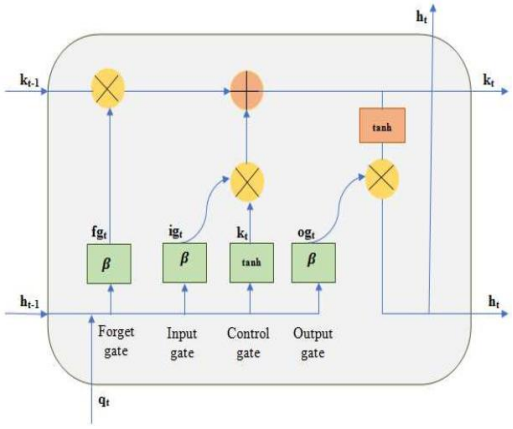

Figure 2: Construction of LSTM.

Each LSTM layer comprises three critical gates: the input gate(ig_t), forget gate (fg_t), and an output gate ((og_t)). These gates manage the flow of information into and out of the cell, which is responsible for storing values over random time intervals. The input gate controls which values are updated, the forget gate decides which information to discard, and the output gate determines what part of the cell state is output at a given time step.However, traditional LSTM training methods often face slower convergence and struggle with optimizing the error function due to gradient descent becoming trapped in local minima caused by random initialization of biases and weights. To address this, the Morlet Wavelet Kernel Function (MWKF) is introduced as a derivative-free optimization approach for complex problems.MWKF replaces the random selection of gate weights and biases in traditional LSTM, improving its performance in data classification and non-linear function approximation tasks. The integration of MWKF with LSTM results in a model known as MWKFLSTM. This approach leverages the MWKF for efficient weight and bias initialization, leading to superior accuracy and faster convergence compared to standard 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 termsb_{fg},b_{ig}, andb_{og} are the bias matrices off_{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.

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

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

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

In the MWKFLSTM model, the weights of the three gates are represented by wi which determines the standards of the input, forget, and output gate weights. The Morlet Wavelet Kernel Function (MWKF) is employed to approximate these weights, leveraging the wavelet function's kernel properties. The construction of MWKF provides enhanced approximation capabilities, which, when applied to LSTM, result in superior performance for classification tasks.The weight values for each gate (wi) are processed through the kernel computations of the MWKF, ensuring better initialization and optimization. Similarly, the calculation of bias values for the gates is carried out using the same kernel computations, ensuring consistency in the optimization process. In addition to the output, the terms for the cell state and applicant cell state are

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

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

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

Where k_{t} besides $k_{\mathrm{t-1}}$ represents the new and preceding cell states of t and also $t-1$. 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 conducted using a DELL laptop running Windows 10, equipped with 16 GB of RAM and an Intel Core i5-10210U processor. The suggested method was applied and evaluated on the dataset in question. The experiment utilized 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), and tensorflow (version 2.6.0). The programming environment used was Spyder Python (version 3.8).

To measure the efficacy of the proposed system, we use the following metrics: Accuracy, Precision, Recall, F1-Score, True Positive Rate, and False Positive Rate. Accuracy is calculated as the ratio of correctly predicted records to the total number of records, 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. It is calculated using the following 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:

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

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

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

- ***Validation Analysis of Proposed Model***

Table 2 offers the investigational study of proposed classical with 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 method compared to existing procedures, the performance metrics for each technique are as follows: 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.

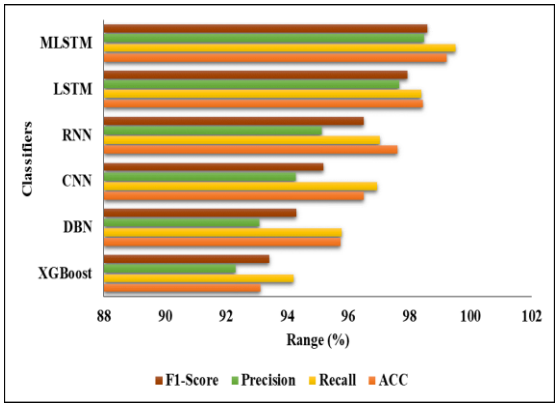

Figure 3: Graphical Description of proposed perfect with existing performances

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.

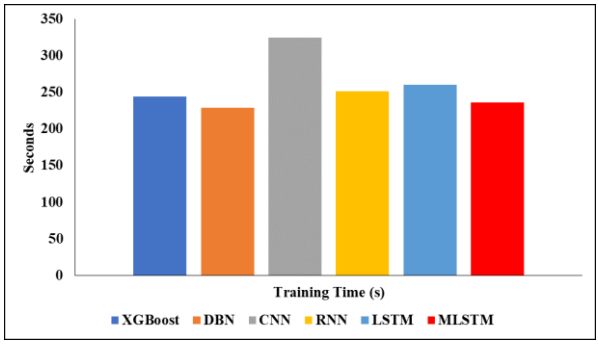

Figure 4: Visual Representation of the proposed model for timing analysis

6. Conclusion

This paper proposes a novel MLSTM model for detecting breaches in IoT-Cloud infrastructure. To enhance the feature selection process, pre-processing techniques and the JBOA algorithm were utilized. Simulations conducted on the BoT-IoT dataset demonstrate the efficacy of the model. The MLSTM achieves a high degree of accuracy in detecting various types of network threats. When compared to state-of-the-art methodologies presented in existing literature, the MLSTM outperforms them in detection accuracy. The model also exhibits the best throughput ratio, lowest false alarm rate, and minimal delay when tested against other approaches using the same dataset. Additionally, the performance of the MLSTM is further validated by its superior efficiency in transferring data packets between cloud-based IoT devices.After the classification of normalcy, the data was securely stored in the cloud. However, some limitations exist within the system, such as the reliability of the input and output data used to train the model. Additionally, the issue of data imbalance in the dataset remains a significant challenge that will be addressed in future stages of the research.

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] Sanz González, R, Luque Juárez, J, M.ª, Martino, L, Liz Rivas, L, Delgado Morán, J, J, & Payá Santos, C, A. (2024) Artificial Intelligence Applications for Criminology and Police Sciences. International Journal of Humanities and Social Science. Vol. 14, No. 2, pp. 139-148. https://doi.org/10.15640/jehd.v14n2a14

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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).

[16] 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).

[17] 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).

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

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

[26] 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.