

# Intrusion detection system in cloud computing by utilizing VTR-HLSTM based on deep learning

Valavan Woothukadu Thirumaran<sup>1</sup>, Nalini Joseph<sup>1</sup>, Umarani Srikanth<sup>2</sup>

<sup>1</sup>Department of Computer Science and Engineering, Bharath Institute of Higher Education and Research, Chennai, India

<sup>2</sup>Department of Computer Science and Engineering, Panimalar Engineering College, Chennai, India

## Article Info

### Article history:

Received Dec 2, 2023

Revised Dec 29, 2023

Accepted Jan 11, 2024

### Keywords:

Cloud computing

Deep learning

Hierarchical long short-term memory

Intrusion detection system

Variance threshold

## ABSTRACT

Cloud computing (CC) is a rapidly developing IT approach with intrusion detection system being a crucial tool for safeguarding virtual networks and machines from potential threats, thereby mitigating security concerns in the cloud environment. The intrusion detection system (IDS) system demands significant improvements, primarily based on optimizing performance and bolstering security measures. This research aims to implement an IDS in cloud computing utilizing deep learning (DL) method. The DL model is a promising technique and is widely used to detect intrusions. The implemented hierarchical long short-term memory (HLSTM) method's performance is evaluated for feature selection through variance threshold-based regression (VTR) on two IDS network datasets: Bot-IoT and network security lab-knowledge discovery and data mining (NSL-KDD). This paper concludes the use of an intrusion detection network resulting in high security and performance. Moreover, the implemented method on the NSL-KDD and Bot-IoT datasets obtains respective accuracies of 99.50% and 0.995. It is compared with the existing methods namely, ensemble ID model for CC utilizing DL, LeNet, fuzzy deep neural network with a Honey Bader algorithm for privacy-preserving ID, and improved metaheuristics with a fuzzy logic-based IDS for cloud security, and beluga whale-tasmanian devil optimization based on deep convolutional neural network (CNN) with TL, chronological slap swarm algorithm-based deep belief network (DBN), and dragonfly improved invasive weed optimization-based Shepard CNN.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Valavan Woothukadu Thirumaran

Department of Computer Science and Engineering, Bharath Institute of Higher Education and Research  
Chennai, India

Email: wtvalavan@gmail.com

## 1. INTRODUCTION

An intrusion detection system (IDS) is a system that monitors traffic in the network to detect potentially dangerous activity [1]. An IDS is a technology that monitors the network traffic and detects malicious traffic or any type of assault, as well as issues alarms [2]. Cloud computing (CC) technology allows users to access, store, and maintain their data in new and potentially easier ways [3], [4]. The client computing paradigm has inherent issues with security, privacy, service availability, and other problems [5]. The cloud's resources and data connections are becoming increasingly challenging to manage, as the number of intrusions increase [6]. One of the techniques for preventing malicious attacks on cloud computing is intrusion detection. The cloud intrusion detection systems (CIDSs) in research focus on the detection and analysis of network traffic, specifically within a cloud environment. They are designed to identify malicious attack behaviors, prevent any potential damage, and ensure the security and reliability of cloud

computing [7]. Attacks against service applications take many different forms in a cloud network including state and protocol attacks, volumetric denial-of-service (DoS) attacks, and attacks involving encrypted or malicious inputs. The security and confidentiality of the system's network are compromised when threats or intrusions are introduced [8]. On platforms that need continuous security monitoring, the network IDS is proven to be helpful in detecting these attacks. It is understood that intrusion detection is crucial for safeguarding any system against malicious activities, and maintaining the system's security so that it may function as intended [9]. IDS is a security tool that tracks the entire network's activity and identifies any malicious activity that is taking place [10]. The most common function of an IDS is to monitor risks and capture intruders before they cause serious damage to the network. The CC, a shared full network of resources and largely scalable internet-based innovation enables several clients to share a collection of resources [11]. The IDS's neural organizations are adaptive and impressive due to their strong ability to express information effectively for a visual demonstration. However, the time taken to run the model on a big dataset is very high [12]. Recent data analysis shows an incredible number of security issues in the virtual network layer of cloud computing. Several earlier researches have used deep learning (DL) to detect attack traffic, combining DL and shallow learning for network intrusion detection system (NIDS) on network security lab-knowledge discovery and data mining (NSL-KDD) and KDD'99, as well as detecting NSL-KDD datasets using DL sparse autoencoder and soft-max regression [13]. The cloud platform is now provided with a variety of IDS tools and techniques that are used to identify attacks on the cloud infrastructure [14]. Even though the cloud service employs authentication technique to safeguard data information of the user to some level, some user data is saved during the storage process. This is because the current research on CC service technology is still at a relatively rudimentary stage in terms of technological advancements. Softmax single classifiers are frequently used in traditional deep learning-based ID methods for categorization in the output layer [15]. Due to their impact on data dimensions, the training model frequently exhibits a poor detection rate of attacks in the presence of large amounts of high-dimensional data, which has an impact on the overall detection efficiency. When deep learning is used to create an IDS, higher-dimensional features are extracted from original data to improve the classification model [16]. Raj and Pani [17] implemented a beluga whale-tasmanian devil optimization (BWTDO)\_DeepCNN-TL model, which was BWTDO based on deep CNN with TL. This implemented method was utilized to develop a high-performance IDS based on a fuzzy classifier in an environment cloud. The implemented hybrid meta-heuristic method combined Matusita Distance and Fisher's score for feature selection, while DeepCNN with TL was utilized in the classification model. The implemented method increased the performance of cloud-based IDS and achieved an accurate detection of network attacks. However, an efficient detection by the implemented BWTDO\_DeepCNN-TL method further required increased outlier detection techniques and clustering incorporation. Salvakkam *et al.* [18] implemented ensemble intrusion detection model for cloud computing utilizing deep learning (EICDL) to detect intrusions effectively. This implemented method utilized NSL-KDD, UNSW-NB15, and KDDcup 1999 datasets to detect cloud computing intrusion. The implemented method improved the both accuracy and detection rate by employing a combination of classification and learning models. However, the implemented method did not possess the adequate ability to detect prior unknown attacks.

Sreelatha *et al.* [19] implemented extended equilibrium deep TL (EEDTL) classification which was utilized to increase the cloud-based computing environment security. The introduced method with sandpiper-based feature selection was employed to develop an efficient cloud IDS. The security system was built to recognize attacks on virtual networks in the cloud at the lowest possible cost. This method determined the suitable features and classified the intrusion with high accuracy and less FAR. Nonetheless, this method had a limited set of features that allowed for fast learning and hence provided a better IDS for the security of cloud data. Selvapandian and Santhosh [20] implemented a LeNet-based IDS which for the environment of IoT multi-cloud. The introduced DL-based IDS method addressed neural network-based IDS. It made use of the dataset of NSL-KDD to provide an improved performance. This method gave a high convergence rate and made it easy to compute inputs. The implemented IDS method increased the accuracy of detection by increasing the efficiency of training. However, the implemented method needed to improve the detection of multi-class attacks effectively in the cloud environment. Jain *et al.* [21] implemented a fuzzy deep neural network with a honey bader algorithm for privacy-preserving ID (FDNN-HBAID) for cloud environment. This method established the scheme ID method with a blockchain-enabled privacy-preserving scheme. As a result, the suggested FDNN-HBAID with the dataset of NSL-KDD had the potential to ensure privacy and security in the cloud infrastructure. Nonetheless, the real-world utilization of shared data was then constrained as the IDS system required to be properly integrated and trusted. Karuppusamy *et al.* [22] implemented a chronological slap swarm algorithm-based deep belief network (chronological SSA-based DBN) which was used to detect suspicious intrusions in the cloud environment. The implemented Chronological SSA-based deep belief network (DBN) was enhanced by combining the chronological concept with slap swarm algorithm. This method was deployed to detect the intrusion by disclosing the fitness

function which provided a minimal error rate as the optimum solution. Nonetheless, the implemented Chronological SSA-based DBN method needed improvement in its ability to detect unknown attacks. Sathiyadhas and Antony [23] proposed dragonfly improved invasive weed optimization-based shepard convolutional neural networks CNN (DIIWO-based ShCNN) method. The proposed DIIWO-based ShCNN method was used to detect attackers and intruders, and mitigate threats in the cloud paradigm with Bot-IoT and KDD cup datasets. The characteristics and features of DA were inherited by IIWO to improve the rate of convergence and enhance the detected effectiveness. But, the implemented method had computation problems with imbalanced datasets. Alohali *et al.* [24] suggested improved metaheuristics with a fuzzy logic-based IDS for cloud security (IMFL-IDSCS) method for intrusion detection. The IMFL-IDSCS method deployed ECOA-FS categorization whilst parameter optimization of JSSO being utilized for ID. The ECOA-FS method was deployed for subset feature selection to increase the performance of classification. The JSSO was combined with the adaptive neuro fuzzy inference system (ANFIS) method for intrusion classification which was further classified into various classes. As a result, the proposed method reduced computational burden and improved the rate of classification. But, the proposed IMFL-IDSCS method had security issues. The main contributions of this study is given as follows:

- To implement variance threshold-based regression (VTR)-hierarchical long short-term memory (HLSTM) method for effective cloud IDS based on the feature selection and categorization methods to increase the cloud environment’s security. The implemented approach addresses useful features and categorizes network attacks by employing novel techniques.
- The filtering techniques resemble pearson correlation, Ridge regression, and variance thresholding are employed for selecting the features, and the HLSTM is used for process of classification.
- A novel method is introduced for detecting cloud computing intrusion utilizing NSL-KDD and Bot-IoT datasets. The implemented method improves the accuracy of detection under lesser computation time.

The rest of this paper is structured as follows: The implemented model is described in section 2. A description of the feature selection and classification methods is given in section 3. The results and discussion are described in section 4, and the conclusion of this paper is given in section 5.

**2. PROPOSED METHOD**

In this proposed work, an IDS is utilized in a cloud computing environment. This method involves the steps of dataset, pre-processing, feature selection, DL model, and evaluation. Figure 1 represents the proposed block diagram of the introduced IDS.

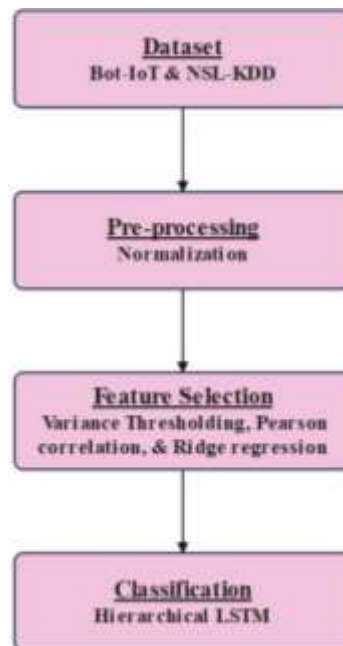


Figure 1. Block diagram of the implemented IDS method

## 2.1. Dataset

The IDS is an application or device that monitors and detects attacks and malicious activities in the IoT network. The two datasets employed for the suggested intrusion detection using deep learning model are NSL-KDD and Bot-IoT. These databases are explained.

### 2.1.1. Bot-IoT Dataset

The dataset of Bot-IoT is used for intrusion detection in IoT networks. The dataset of Bot-IoT is generated by the Cyber Range Lab of UNSW Canberra in a realistic network environment. In this dataset, botnet and normal traffic are combined for the network environment, and the complete dataset contains 72 million records. 364,562 records are used for training the dataset, whereas 243,043 records are used in testing the dataset. This dataset includes four kinds of attacks: reconnaissance, distributed DoS (DDoS), theft, and DoS [25].

### 2.1.2. NSL-KDD dataset

The dataset of NSL-KDD is generated by utilizing the KDDcup99 dataset which is the most popular intrusion dataset. The problem with the KDDcup99 dataset is that it contains duplicate records in training and testing data and has bias classifiers [26]. This problem is resolved in the NSL-KDD dataset. This dataset has a total of 125,973 records, out of which 67,343 are benign, and 58,630 are anomaly samples. One of the most often used in the IDS framework evaluation is the NSL-KDD dataset. Apart from the normal network traffic, the NSL-KDD consists of 4 kinds of intrusions they are which are, U2R, DoS, Probe, and R2L.

## 2.2. Pre-processing

Pre-processing is a process that involves cleaning data and converting the actual dataset into a suitable format. Normalization is a pre-processing method that is mainly deployed in the preparation of data parts for the models of DL. By using the basic scale and avoiding perverting variance in the original value ranges, the normalization process attempts to control the numerical data values in the dataset (i.e., 0 and 1) without losing any data. Through creating a new value that manages resource data ratios and basic distribution, normalization is efficient in protecting the values of all the models' numerical columns.

Furthermore, the method of changing the samples of training data called min-max normalization is employed in this research. The datasets contain minimal and maximal values for each characteristic. This approach is particularly deployed to normalize the data by providing every element within [0,1] is normalized, which improves the efficiency of categorization for every data element. The Min-Max normalization is numerically represented in (1).

$$i = \frac{n_i - Min}{Max - Min} \quad (1)$$

Where, the numeric feature for the  $i$ th sample is denoted as  $n_i$ , Min and Max are the minimum and maximum values of numeric features. Then, the pre-processed output is provided for feature selection as input.

## 3. RESEARCH METHOD

The DL model is a promising technique, widely used for intrusion detection. The implemented HLSTM method's performance is evaluated for feature selection through VTR on two IDS network datasets: NSL-KDD, and Bot-IoT. A technique for choosing pertinent features and eliminating redundant or unnecessary ones is provided by the VTR. Concentrating on the most useful features and decreasing the dimensionality of the input space increases the model's efficiency. The model's ability to be more significantly generalized is enhanced by the combination of HLSTM and VTR. The model's adaptability to new intrusion patterns is heightened by selecting features with higher relevance, thereby reducing the risk of overfitting in training data noise.

### 3.1. Feature selection

After pre-processing, the output of the pre-processed data is fed into as input for feature selection. Feature selection is a process that includes the necessary or significant features and removes the irrelevant features in the dataset. Filtering techniques that resemble pearson correlation, ridge regression, and variance thresholding are used in the proposed implementation to select the features. By utilizing any of these feature selection methods, the necessary features are selected for the IDS.

**3.1.1. Variance thresholding**

Variance thresholding is a feature selection technique which removes low-variance features from the dataset that are not highly significant for modeling purposes after the pre-processing step. To further reduce the redundant features, variance thresholding is used after pearson correlation. This filter-based technique is applied to eliminate features with variance values below a predetermined threshold. This is because the features with a reduced variance transmit less information as the variation is inversely correlated with the predictive power. As a result, a threshold of 0.0001 is used, and the variance of each feature determined, while the variance thresholding value is further used in the person correlation to select the features [27].

**3.1.2. Pearson correlation**

In this phase, a feature optimization approach is used to reduce the input dimension by selecting an optimal feature subset. Pearson’s correlation coefficient filter method is deployed to find the discriminative features in the feature set with the value of variance thresholding. It produces a value of correlation coefficient in the range of [-1, 1] and assesses the similarity comparable to the attributes or qualities of the dataset [28]. It shows that when value of correlation coefficient is equal to 1 it is an entirely positive correlation, and when equal to -1 it is an entirely negative correlation. This demonstrates that there is a strong correlation between features and the value with a high coefficient, and vice versa. In (2) mathematically expresses the pearson’s correlation coefficient where ‘σ<sub>X</sub>’ and ‘σ<sub>Y</sub>’ denote ‘X’ and ‘Y’ standard deviations, respectively, (XY) is a covariance measure for ‘X’ and ‘Y’, and the expected value of ‘E’ is ‘E(X)’.

$$\rho_{x,y} = \frac{(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X-\mu_X)(X-\mu_Y))}{\sigma_X \sigma_Y} = \frac{E(XY)-E(X)E(Y)}{\sqrt{E(X^2)-E^2(X)}\sqrt{E(Y^2)-E^2(Y)}} \tag{2}$$

**3.1.3. Ridge regression**

Ridge regression is a method of model tuning that is utilized to evaluate any data that suffers from multicollinearity. By adding a L<sub>2</sub> norm penalty to the negative of the log-likelihood function, Ridge regression is intended to address the overfitting issue with logistic regression [29]. In (3) mathematically expresses the Ridge regression model:

$$\theta = arg\ min\{-L(\theta|D) + P(\theta; \lambda)\} \tag{3}$$

Where, the log-likelihood function is denoted as  $L(\theta|D) = \sum_{i=1}^n \{y_i \log[f(X_i^i)] + (1 - y_i) \log[1 - f(X_i^i)]\}$ , the sigmoid function is denoted as  $f(x) = \exp(x) / (1 + \exp(x))$ , the penalty term is denoted as  $P(\theta; \lambda) = \lambda \sum_{j=1}^p \theta_j^2$ , the intercept term of unknown coefficient is  $\theta_0$ , and the vector is denoted as  $\theta = (\theta_1, \theta_2, L, \theta_p)^t$ . In (6) measures the feature's importance where the feature is more significant with higher absolute magnitude of the coefficient. Here, the Ridge regression’s error rate should be lower than the selected features of pearson correlation.

**3.2. Classification**

The DL HLSTM method is deployed after feature selection for classification in this scenario. Generally, LSTM detects attacks in cloud computing by classifying malicious data from various stages into the respective classification systems. LSTM neural network identifies long-term dependencies between the time steps of sequence data from the dataset. The LSTM layers have input, word embedding (WE), Softmax, and output classification layers.

**3.2.1. LSTM**

The LSTM neural network identifies long-term dependencies between the time steps of sequence data from the dataset. Regardless of the window size, the LSTM networks are qualified to be utilized to detect attack patterns that repeat within a sequence of long packets [30]. Figure 2 shows a basic LSTM cell that forms the foundation of the LSTM architecture. It contains three gates to control the data flow from one cell state to another. The LSTM neural network method which is represented in (4) demonstrates the forward calculation method.

The gates are classified as the output, forget, and input gates. To provide a determination and control of the information flow, all three gates rely on the activation of sigmoid  $\sigma$ . The determining of whether a certain piece of information needs to be kept or forgotten in a given data sample is called the forget gate. The current input signal  $X_t$ , and the previous output sequences  $y_{t-1}$  in the cell state  $C_{t-1}$  are interpreted by this gate to produce an output  $f_t$  in the interval between 1 and 0, where 1 represents entirely remembering

the information and 0 represents fully forgetting it. The information included in the current cell state  $C_t$  is determined by multiplying the output by the activation layer tanh input gate. Similarly, the output gate flow of the proportion of details  $y_t$  in  $C_t$  at the cell's result is determined by combining the LSTM cell's output with the result of another activation layer tanh. The LSTM cell of three gate operations to generate outcome  $y_t$  in state of cell  $C_t$  is represented by the following in (4).

$$\begin{aligned}
 f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
 i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
 \tilde{C}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\
 C_t &= f_t C_{t-1} + i_t \tilde{C}_t \\
 O_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
 h_t &= O_t \tanh(C_t)
 \end{aligned} \tag{4}$$

Where, the terms for matrix and bias are respectively  $W$  and  $b$ , whereas  $f$  is the forgetting gate,  $i$  is the gate of input, and  $o$  is the gate of output. The tangent activation functions in a hyperbolic order as the tanh is sigmoid. The usage of this formula demonstrate that the LSTM solves the issue of long-term RNN dependence. As a consequence, DDoS attacks are possible to be identified and detected using the derived functions, which are then used to set up and detect a neural network prediction method depending on the LSTM. Next, by applying input data architecture from the NSL-KDD and Bot-IoT datasets and optimizing the weights of the LSTM neural network using the H-LSTM neural network, the predicted accuracy is increased. Better results are obtained by using the hierarchical algorithm, which helps to further explain the parameters.

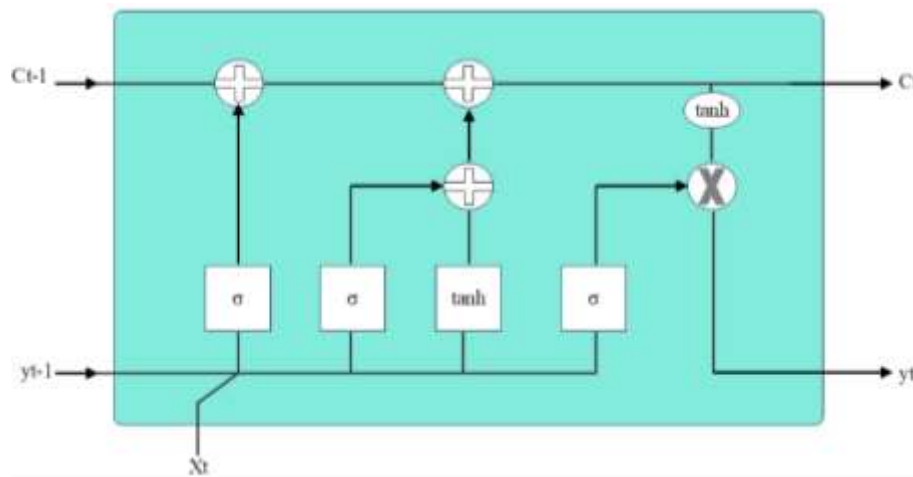


Figure 2. LSTM cell

### 3.2.2. Hierarchical LSTM

This hierarchical method increases the detection accuracy and produces a robust performance in the classification process. The model of convolutional LSTM takes a vector income  $V = v_1, v_2, \dots, v_n$  and predicts the vector outcome  $U = u_1, u_2, \dots, u_n$  [31]. The (5) and (6) are used to calculate the token in sequence of softmax function.

$$\text{the } S(U|V) = \prod_{t \in [1, n_u]} S(u_t | v_1, v_2, \dots, v_t, u_1, u_2, \dots, u_{t-1}) \tag{5}$$

$$S(U|V) = \prod_{t \in [1, n_u]} \frac{\exp(f(p_{t-1}, e_{u_t}))}{\exp(\sum_{\hat{u}} f(p_{t-1}, e_{\hat{u}}))} \tag{6}$$

Where,  $p_{t-1}$  denotes the output of LSTM, whereas  $f(p_{t-1}, e_{\hat{u}})$  denotes the activation function. The complex sequence is learned at various levels with the hierarchical LSTM. According to (7), the sequence vector is generated at the recurrent network's first layer.

$$p_t^s = LSTM(p_{t-1}^s, e_t^s) \quad (7)$$

Where,  $p_{t-1}^s$  denotes the hidden vector, while  $e_t^s$  denotes the world level embedding. In order to make the explanation less complicated, the operation of LSTM is described with the function. In (8) is used to create the document vector at the 2<sup>nd</sup> layer of the recurrent network. Here,  $p_{t-1}^d$  denotes the hidden vector, and embedding at the sentence level in the LSTM model. The sentence structure and word-level structure are retained by the vector document.

$$p_t^d = LSTM(p_{t-1}^d, e_t^d) \quad (8)$$

HLSTM is used in the implemented cloud computing method to improve the computation efficacy and ID accuracy. The number of records are used to create a grayscale image, where the Bot-IoT dataset comprises 43 features, while the dataset of NSL-KDD has 40 features. For every record, a 7×7 image is possible to be created, the first layer (7,1) is encoded by the HLSTM model as a (64) shape column vector. Furthermore, a column vector 7 along the form (7,64) encodes every 7 column vectors to create a complete image, and finally, the forecast is attained with a connection of the complete layer.

Since the LSTM's success is typically dependent on the hierarchy that recognizes its learning to other layers, the HLSTM method is scalable. In a cloud computing environment, every layer completes a task before passing it on to the subsequent layer. In this approach, a pipeline where every layer calculates local data and delivers activity to the LSTM's upper layer is created. The cloud computing design at every LSTM layer performs to minimize the computing load on the cloud, whilst the deep learning contributing to the established construction of a global model. Pseudocode 1 represents a pseudo-code of HLSTM in Cloud system for intrusion detection. The Table 1 shows the notation description.

#### Pseudocode 1. HLSTM in Cloud system for intrusion detection

```
# Load dataset (X, y) where X is input data and y is labels
# Variable Threshold-based Ridge regression for feature selection
def ridge_regression_feature_selection(X, y, threshold):
    selected_features = []
    # Iterate over features
    for feature in range(X.shape[1]):
        # Apply Ridge regression
        ridge_model = Ridge(alpha=1.0)
        ridge_model.fit(X[:, feature].reshape(-1, 1), y)
        # Check the coefficient magnitude against the threshold
        if np.abs(ridge_model.coef_[0]) > threshold:
            selected_features.append(feature)
    return selected_features
# Set the threshold based on your requirements
threshold_value = 0.1
# Select features using Ridge regression
selected_features = ridge_regression_feature_selection(X, y, threshold_value)
# Use the selected features for further analysis or model training
X_selected = X[:, selected_features]
# Hierarchical LSTM model
def hierarchical_lstm_model(input_shape):
    model = Sequential()
    # First LSTM layer
    model.add(LSTM(units=50, return_sequences=True, input_shape=input_shape))
    # Second LSTM layer
    model.add(LSTM(units=50, return_sequences=True))
    # Third LSTM layer
    model.add(LSTM(units=50))
    # Output layer
    model.add(Dense(units=1, activation='sigmoid'))
    # Compile the model
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
# Train the hierarchical LSTM model
input_shape = (X_selected.shape[1], 1) # Adjust input shape based on selected features
model = hierarchical_lstm_model(input_shape)
model.fit(X_selected, y, epochs=10, batch_size=32)
# Evaluate the model on a test set
test_predictions = model.predict(X_test[:, selected_features])
```

Table 1. Notation description

Symbol	Description
$n_i$	numeric feature for the $i$ th sample
$Min$	minimum values
$Max$	maximum values
$\sigma_X$	standard deviations 'X'
$\sigma_Y$	standard deviations 'Y'
$(XY)$	covariance measure for 'X' and 'Y'
$E(X)$	expected value of 'E'
$L(\theta D) = \sum_{i=1}^n \{y_i \log[f(X_i^i)] + (1 - y_i) \log[1 - f(X_i^i \theta)]\}$	log-likelihood function
$f(x) = \exp(x) / (1 + \exp(x))$	sigmoid function
$P(\theta; \lambda) = \lambda \sum_{j=1}^p \theta_j^2$	penalty term
$W$ and $b$	matrix and terms for the bias
$f$	forgetting gate
$i$	input gate
$o$	output gate
$V = v_1, v_2, \dots, v_n$	vector income
$U = u_1, u_2, \dots, u_n$	vector outcome
$p_{t-1}$	output of LSTM
$f(p_{t-1}, e_{\hat{u}})$	activation function
$p_{t-1}^s$ and $p_{t-1}^d$	hidden vectors
$e_t^s$	world level embedding

## 4. RESULTS AND DISCUSSION

### 4.1. Evaluation parameters

For the assessment of fault diagnosis, six parameters which are accuracy, specificity, sensitivity, recall, precision, and f-measure are used [32]-[34]. Here, the number of normal instances is denoted as  $N$ , the number of correctly predicted normal instances is denoted as  $FN$ , the number of fault instances is denoted as  $P$ , the number of correctly predicted fault instances is denoted as  $TP$ , the incorrectly predicted fault samples is denoted as  $FP$ .

Accuracy: it is the proportion of accurate predictions to all input samples which is calculated using in (9).

$$Accuracy = \frac{TP+TN}{(P+N)} \times 100 \quad (9)$$

Specificity: it is the probability of negative test results, conditioned on the proportion of truly being negative. It is calculated using in (10).

$$Specificity = \frac{TN}{(FP+TN)} \quad (10)$$

Sensitivity: the sensitivity of a model is a measure of its ability to predict true positives in each of the categories that is accessible. It is calculated using in (11).

$$Sensitivity = \frac{TP}{(TP+FN)} \quad (11)$$

Recall: it measures the wide range of accurate predictions. It is evaluated by dividing the total of true positive predictions by the sum of  $TP$  and  $FN$  predictions. It is calculated using in (12).

$$Recall = \frac{TP}{(TP+FN)} \quad (12)$$

Precision: the precision measures the percentage of actual data records versus the expected data records. The performance of the classification model is greater if the precision is higher. It is calculated using in (13).

$$Precision = \frac{TP}{(TP+FP)} \quad (13)$$

F-measure: it is a single static that combines recall and precision to capture both features. It is calculated using in (14).



$$F\text{-measure} = \frac{(2 * Precision * Recall)}{(Precision + Recall)} \tag{14}$$

**4.2. Result**

Tables 2 and 3 display the enhanced performance analysis of specificity, F1-score, sensitivity, precision, accuracy, as well as the selected feature selection. Tables 2 and 3 represent the simulation results of HLSTM by varying the classifiers for the respective datasets of Bot-IoT and NSL-KDD.

**Table 2. HLSTM’s simulation results by various classifiers for the dataset Bot-IoT**

	Methods	Precision	Accuracy	F1-score	Specificity	Recall	Sensitivity
Selected feature selection results	CNN	0.933	0.903	0.964	0.926	0.902	0.952
	RNN	0.922	0.902	0.959	0.945	0.951	0.910
	LSTM	0.921	0.925	0.976	0.984	0.899	0.965
	HLSTM	0.99	0.995	0.987	0.992	0.987	0.982
Actual feature selection results	CNN	0.923	0.915	0.956	0.934	0.881	0.916
	RNN	0.943	0.943	0.914	0.925	0.947	0.956
	LSTM	0.953	0.922	0.967	0.945	0.937	0.927
	HLSTM	0.966	0.993	0.978	0.956	0.962	0.974

In comparison to classifiers selected feature selection, those features on the dataset of Bot-IoT have the highest performance metrics. Table 2 displays the outcomes of simulations using the Bot-IoT dataset and the HLSTM algorithm by employing various classifiers. The implemented HLSTM is compared to the recurrent neural network (RNN), CNN, and LSTM in terms of sensitivity, precision, accuracy, F1-score, and specificity. The graphical representation of Bot-IoT’s disease classification performed through the selected and actual feature selection is shown in Figures 2 and 3, simultaneously. In comparison with the other classifiers, the results obtained reveal that the implemented HLSTM obtains the superior values with an accuracy of 0.995, precision of 0.99, sensitivity of 0.982, specificity of 0.992, recall of 0.987, and f1-score of 0.987. Figures 3 and 4 represent the quantitative analysis of the actual and selected feature selection of various categories for the datasets Bot-IoT and NSL-KDD, simultaneously.

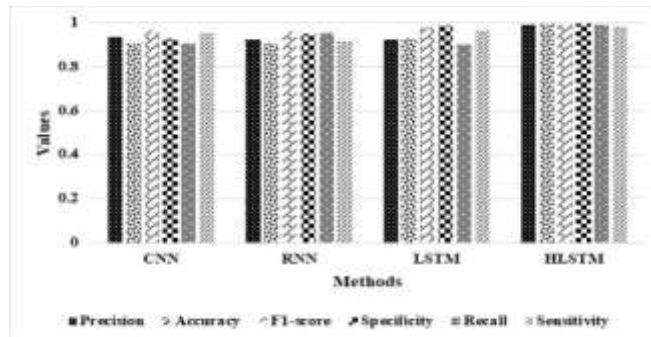


Figure 2. HLSTM optimization performance selected feature selection

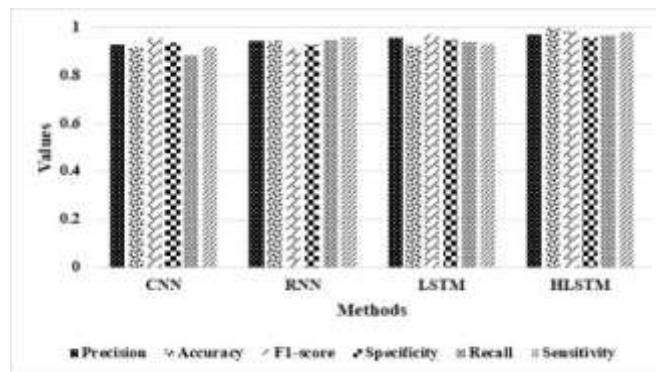


Figure 3. HLSTM optimization performance actual feature selection

Table 3. HLSTM’s simulation results by various classifiers for the dataset of NSL-KDD

	Methods	Precision (%)	Accuracy (%)	F1-score (%)	Specificity (%)	Recall (%)
Selected Feature selection results	CNN	91.50	89.24	94.75	92.56	90.35
	RNN	88.96	92.47	93.20	94.99	95.23
	LSTM	93.27	94.31	96.60	94.84	93.22
	HLSTM	99.73	99.50	99.46	99.62	99.70
Actual Feature selection results	CNN	89.32	87.42	94.42	90.54	92.10
	RNN	94.78	93.25	93.27	91.80	94.15
	LSTM	98.42	95.71	96.78	96.61	92.58
	HLSTM	98.00	98.95	98.53	98.73	97.50

The NSL-KDD dataset’s classifiers display the superior performance metrics in contrast to classifiers that use feature selection. The outcomes of HLSTM’s simulations on the NSL-KDD with different classifiers are represented in Table 3. The suggested HLSTM is compared to the RNN, CNN, and LSTM in terms of recall, precision, F1-score, specificity, and accuracy.

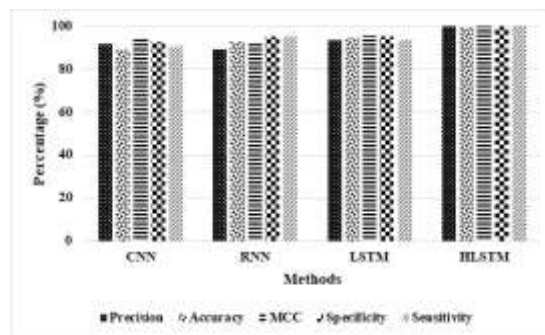


Figure 4. HLSTM optimization performance selected feature selection

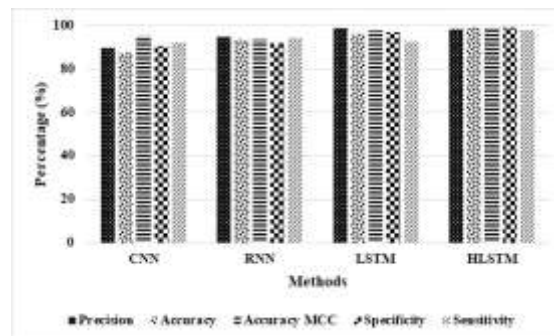


Figure 5. HLSTM optimization performance actual feature selection

Figure 4 displays the graphical representation of HLSTM’s optimization performance in selected feature selection. Figure 5 shows the graphical representation of the actual feature selection of HLSTM’s optimization performance. In comparison with the other classifiers, the results obtained reveal that the suggested HLSTM obtains the greatest values with a 99.50% of accuracy, 99.73% of precision, 99.70% of recall, 99.46% of f1-score, and 99.62% of specificity.

In this study, the FRIEDMAN test is selected to prove the difference among the categorization methods (CNN, RNN, LSTM, and HLSTM). The models are ranked independently for each fold in the database using the FRIEDMAN test. For example, the Bot-IoT dataset assigns a rating of one to the model that performs the best, two to the second best, and so on. The null hypothesis is rejected since Table 4 shows that p-value is under accuracy [35], [36]. In such a way, at least one categorization method varies from other categorization methods over all datasets (Bot-IoT and NSL-KDD). Table 4 shows the statistics of FRIEDMAN test for five-fold cross-validation. It is hence evident that classification model’s outcomes are different by means of recall, F1-measure, accuracy, and precision.

Table 4. Obtained results of the FRIEDMAN test

FRIEDMAN test	Precisior	Recall	F1-measure	Accuracy
F-statistic	1.256	1.075	1.684	1.423
P-value	0.984	0.587	0.947	0.853

**4.3. Comparative analysis**

In this section, the implemented method’s results are analyzed by utilizing the parameters of precision, accuracy, f1-score, recall, sensitivity, and specificity. The comparative analysis of the existing and proposed models on the datasets of Bot-IoT is shown in Table 5. This comparative analysis contrasts the suggested model with the existing approaches on the Bot-IoT dataset. These results are shown in Table 3 which demonstrates well that the developed approach performs better than the previous models: BETDO\_CNN-TL [13], chronological SSA-based DBN [18], and DIIWO-based ShCNN [19]. With the values of 0.99, 0.95, 0.992, 0.987, 0.982, and 0.987 in the respective terms of precision, f1-score, specificity, recall, accuracy, sensitivity, the suggested HLSTM outperforms the Bot-IoT dataset.

The introduced model’s outcomes are analyzed based on the parameters of recall, precision, f1-score, and accuracy. The comparative analysis of the existing and presented approach on the NSL-KDD dataset is presented in Table 6. In this evaluation, the results of the presented approach on the NSL-KDD dataset is compared with that of the existing approaches. According to the measures of accuracy, precision, recall, and f1-score, the results displayed in Table 4 indicate that the suggested method outperformed all other models namely, EICDL [14], LeNet [16], FDNN-HBAID [17], and IMFL-IDSCS [20]. The implemented HLSTM achieves better accuracy on the dataset of NSL-KDD with values of 99.73%, 99.50%, 99.46%, and 99.70% in terms of f1-score, accuracy, precision, and recall, simultaneously.

Table 5. Comparative analysis of implemented method using Bot-IoT dataset

Authors	Methods	Precision (%)	Accuracy (%)	Recall (%)	F1-score (%)	Sensitivity (%)	Specificity (%)
Boukhalifa <i>et al.</i> [13]	BETDO_Deep CNN-TL	0.926	0.924	0.924	0.927	N/A	N/A
Salvakkam <i>et al.</i> [18]	Chronological SSA-based DBN	N/A	0.976	N/A	N/A	0.982	0.930
Sathiyadhas and Sreelatha <i>et al</i> [19]	DIIWO- based ShCNN	0.950	N/A	N/A	N/A	0.955	0.948
Proposed method	VTR-HLSTM	0.99	0.995	0.987	0.987	0.982	0.992

Table 6. Comparative analysis of implemented method using dataset NSL-KDD

Authors	Methods	Precision (%)	Accuracy (%)	F1-score (%)	Recall (%)
Krishnaveni <i>et al.</i> [14]	EICDL	82.60	91.71	N/A	78.09
Gao [16]	LeNet	94.41	96.28	N/A	97.51
Raj and Pani. [17]	FDNN-HBAID	99.39	99.39	99.39	99.39
Selvapandian and Santhosh. [20]	IMFL-IDSCS	92.03	99.31	81.08	78.25
Proposed method	VTR-HLSTM	99.73	99.50	99.46	99.70

**4.4. Discussion**

The suggested VTR-HLSTM’s individual performance, as well as its comparative study with the previously introduced methods namely, BETDO\_Deep CNN-TL [13], Chronological SSA-based DBN [18], and DIIWO- based ShCNN [19] EICDL [14], LeNet [16], FDNN-HBAID [17], and IMFL-IDSCS [20] as given in section 4.3 are discussed in this section. The existing method BWTDO\_DeepCNN-TL [17], has limitations such as the requirement to enhance the outlier detection techniques as well as the clustering incorporation for an efficient detection. Furthermore, the EICDL [18] model does not possess an adequate ability to detect prior unknown attacks. The EEDTL [19] method has only a limited set of features which allowed for fast learning so as to provide better IDS for the security of cloud data. LetNet [20] method needs to improve the detection of multi-class attacks effectively in the cloud environment while the IMFL-IDSCS [24] method has security issues. Therefore, the VTR-HLSTM method is recommended for an effective cloud IDS based on the feature selection and categorization method employed to increase cloud security in the environment. The main goal of the HLSTM with variance threshold is to increase the security and performance of the cloud computing IDS. By utilizing the Bot-IoT and NSL-KDD datasets, this novel method is recommended for detecting cloud computing intrusion. This new ID model for cloud computing using VRT-HLSTM is introduced after a careful investigation into the drawbacks of the existing IDS

techniques. It ensures a secure data transfer in the cloud computing architecture with the implemented ID method which avoids end-user security issues. But employing this nuanced methodology, the introduced approach addresses the advantageous features and categorizes the network attacks. The two datasets utilized for Intrusion Detection using the deep learning model are Bot-IoT and NSL-KDD. First, the pre-processing of data is a crucial step that is carried out to ensure the robust performance of the introduced method. This step involves cleaning the data and converting the actual dataset into a suitable format. Next, feature selection which is a process that includes the selection of the necessary or significant features alongside the removal of the irrelevant features in the dataset using Pearson correlation and variance thresholding methods is carried out. Finally, the HLSTM method executes the phase of classification using the most significant and pertinent features following the selection of the optimal features. Here, the VTR assists to improve data privacy, whereas the HLSTM method detects the intrusion in CC. Additionally, the proposed VTR-HLSTM method is also capable of recognizing the vector attacks. The experimental outcomes accomplish attack detection with the involvement of the proposed VTR-LSTM, therefore establishing that it is more robust than the existing methods. The performance of VTR-HLSTM is validated on both the Bot-IoT and NSL-KDD datasets. In comparison to the existing methods such as BETDO\_Deep CNN-TL [13], Chronological SSA-based DBN [18], and DIIWO-based ShCNN [19] EICDL [14], LeNet [16], FDNN-HBAID [17], and IMFL-IDSCS [20], the VTR-HLSTM achieves 0.995% and 99.50% of accuracy on the respective Bot-IoT and NSL-KDD datasets. The VTR-LSTM exhibits finer performance guaranteeing reliability, strength and accuracy of the proposed model over the previous methods.

## 5. CONCLUSION





This research paper recommends a variance threshold-based regression feature selection using the (VTR-HLSTM) method to increase the performance of cloud-based IDS. The recommended VTR-HLSTM method covers various component steps including the pre-processing of data dimensions using normalization, feature selection using Pearson correlation and variance thresholding, along with hierarchical LSTM-based classification. The experimental outcomes display the VTR-HLSTM's robustness over the other methods based on the metrics of recall, f1-score, precision, accuracy, sensitivity, and specificity. Accordingly, this recommended method is used by IDS for an effectual ID in cloud computing. The proposed method possesses an improved accuracy for detection with less computation time. On the Bot-IoT and NSL-KDD datasets, the implemented method obtains the highest accuracy of 0.995% and 99.50% values, respectively. In the future, deep learning and feature optimization approaches will be used to develop more efficient IDS in the cloud network.

## REFERENCES





- [1] M. Ramasamy, and P.V. Eric, "A novel classification and clustering algorithms for intrusion detection system on convolutional neural network," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 5, pp. 2845-2855, 2022, doi: 10.11591/eei.v11i5.4145
- [2] J. Jose, and D.V. Jose, "Deep learning algorithms for intrusion detection systems in the Internet of things using CIC-IDS 2017 dataset," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 1, pp.1134-1141, 2023, doi: 10.11591/ijece.v13i1.pp1134-1141.
- [3] N. O. Ogwara, K. Petrova, and M. L. Yang, "Towards the development of a cloud computing intrusion detection framework using an ensemble hybrid feature selection approach," *J. Comput. Networks Commun.*, vol. 2022, p. 5988567, Feb. 2022, doi: 10.1155/2022/5988567.
- [4] K. G. Maheswari, C. Siva, and G. Nalinipriya, "A hybrid soft computing technique for intrusion detection in web and cloud environment," *Concurrency Comput. Pract. Exper.*, vol. 34, no. 22, p. e7046, Aug. 2022, doi: 10.1002/cpe.7046.
- [5] P. Singh and V. Ranga, "Attack and intrusion detection in cloud computing using an ensemble learning approach," *Int. J. Inf. Technol.*, vol. 13, no. 2, pp. 565-571, Apr. 2021, doi: 10.1007/s41870-020-00583-w.
- [6] C. Kavitha, M. Saravanan, T. R. Gadekallu, K. Nirmala, B. P. Kavin, and W. -C. Lai, "Filter-based ensemble feature selection and deep learning model for intrusion detection in cloud computing," *Electronics*, vol. 12, no. 3, p. 556, Jan. 2023, doi: 10.3390/electronics12030556.
- [7] W. Wang, X. Du, D. Shan, R. Qin, and N. Wang, "Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1634-1646, 2022, doi: 10.1109/TCC.2020.3001017
- [8] K. Samunnisa, G.S.V. Kumar, and K. Madhavi, "Intrusion detection system in distributed cloud computing: hybrid clustering and classification methods," *Meas.: Sens.*, vol. 25, p. 100612, Feb. 2023, doi: 10.1016/j.measen.2022.100612.
- [9] Farhana, K., Rahman, M. and Ahmed, M.T., 2020. An intrusion detection system for packet and flow based networks using deep neural network approach. *International Journal of Electrical & Computer Engineering (2088-8708)*, 10(5).
- [10] Javadpour, P. Pinto, F. Ja'fari, and W. Zhang, "DMAIDPS: a distributed multi-agent intrusion detection and prevention system for cloud IoT environments," *Cluster Comput.*, vol. 26, no. 1, pp. 367-384, Feb. 2023, doi: 10.1007/s10586-022-03621-3.
- [11] S. Krishnaveni, S. Sivamohan, S. Sridhar, and S. Prabhakaran, "Network intrusion detection based on ensemble classification and feature selection method for cloud computing," *Concurrency Comput. Pract. Exper.*, vol. 34, no. 11, p. e6838, May 2022. doi: 10.1002/cpe.6838.
- [12] G. Nagarajan and P. J. Sajith, "Optimization of BPN parameters using PSO for intrusion detection in cloud environment," *Soft Comput.*, Jun. 2023, doi: 10.1007/s00500-023-08737-1.

- [13] A. Boukhalfa, A. Abdellaoui, N. Hmina, and H. Chaoui, "LSTM deep learning method for network intrusion detection system," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 3, p.3315, 2020, doi: 10.11591/ijece.v10i3.pp3315-3322.
- [14] S. Krishnaveni, S. Sivamohan, S.S. Sridhar, and S. Prabakaran, "Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing," *Cluster Comput.*, vol. 24, no. 3, pp. 1761-1779, Sep. 2021, doi: 10.1007/s10586-020-03222-y.
- [15] X. Wang, "A collaborative detection method of wireless mobile network intrusion based on cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2022, p. 1499736, Oct. 2022, doi: 10.1155/2022/1499736.
- [16] J. Gao, "Network intrusion detection method combining CNN and biLSTM in cloud computing environment," *Comput. Intell. Neurosci.*, vol. 2022, p. 7272479, Apr. 2022, doi: 10.1155/2022/7272479.
- [17] M.G. Raj, and S.K. Pani, "Hybrid feature selection and BWTDO enabled DeepCNN-TL for intrusion detection in fuzzy cloud computing," *Soft Comput.*, Jun. 2023, doi: 10.1007/s00500-023-08573-3.
- [18] D.B. Salvakkam, V. Saravanan, P.K. Jain, and R. Pamula, "Enhanced quantum-secure ensemble intrusion detection techniques for cloud based on deep learning," *Cognit. Comput.*, vol. 15, no. 5, pp. 1593-1612, Sep. 2023, doi: 10.1007/s12559-023-10139-2.
- [19] G. Sreelatha, A.V. Babu, and D. Midhunchakkaravarthy, "Improved security in cloud using sandpiper and extended equilibrium deep transfer learning based intrusion detection," *Cluster Comput.*, vol. 25, no. 5, pp. 3129-3144, Oct. 2022, doi: 10.1007/s10586-021-03516-9.
- [20] D. Selvapandian and R. Santhosh, "Deep learning approach for intrusion detection in IoT-multi cloud environment," *Autom. Software Eng.*, vol. 28, no. 2, p. 19, Sep. 2021, doi: 10.1007/s10515-021-00298-7.
- [21] D.K. Jain, W. Ding, and K. Kotecha, "Training fuzzy deep neural network with honey badger algorithm for intrusion detection in cloud environment," *Int. J. Mach. Learn. Cybern.*, vol. 14, no. 6, pp. 2221-2237, Jun. 2023, doi: 10.1007/s13042-022-01758-6.
- [22] L. Karuppusamy, J. Ravi, M. Dabhu, and S. Lakshmanan, "Chronological salp swarm algorithm based deep belief network for intrusion detection in cloud using fuzzy entropy," *Int. J. Numer. Modell. Electron. Networks Devices Fields*, vol. 35, no. 1, p. e2948, Jan./Feb. 2022, doi: 10.1002/jnm.2948.
- [23] S.S. Sathiyadhas, and M.C.V.S. Antony, "A network intrusion detection system in cloud computing environment using dragonfly improved invasive weed optimization integrated Shepard convolutional neural network," *Int. J. Adapt. Control Signal Process.*, vol. 36, no. 5, pp. 1060-1076, May 2022, doi: 10.1002/acs.3386.
- [24] M.A. Alohal, M. Elsadig, F.N. Al-Wesabi, M. Al Duhayyim, A.M. Hilal, and A. Motwakel, "Enhanced chimp optimization-based feature selection with fuzzy logic-based intrusion detection system in cloud environment," *Appl. Sci.*, vol. 13, no. 4, p. 2580, Feb. 2023. doi: 10.3390/app13042580.
- [25] M. Zeeshan, Q. Riaz, M.A. Bilal, M.K. Shahzad, H. Jabeen, S.A. Haider, and A. Rahim, "Protocol-based deep intrusion detection for dos and ddos attacks using unsw-nb15 and bot-iot data-sets," *IEEE Access*, vol. 10, pp.2269-2283, 2021, doi: 10.1109/ACCESS.2021.3137201.
- [26] A. Alsirhani, M.M. Alshahrani, A.M. Hassan, A.I. Taloba, R.M. Abd El-Aziz, and A.H. Samak, "Implementation of African vulture optimization algorithm based on deep learning for cybersecurity intrusion detection," *Alexandria Engineering Journal*, vol. 79, pp.105-115, 2023, doi: 10.1016/j.aej.2023.07.077.
- [27] E. Erin, and B. Semiz, "Spectral Analysis of Cardiogenic Vibrations to Distinguish Between Valvular Heart Diseases," in *Proceedings of the 16th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2023)*, vol. 4, pp. 212-219, 2023, doi: 10.5220/0011663900003414.
- [28] G. Li, A. Zhang, Q. Zhang, D. Wu, and C. Zhan, "Pearson correlation coefficient-based performance enhancement of broad learning system for stock price prediction," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 5, pp.2413-2417, 2022. doi: 10.1109/TCSII.2022.3160266.
- [29] L. Li, Y.A. Algabri, and Z.P. Liu, "Identifying diagnostic biomarkers of breast cancer based on gene expression data and ensemble feature selection," *Current Bioinformatics*, vol. 18, no. 3, pp. 232-246, 2023. doi: 10.2174/157489361866623011153243.
- [30] T. Alladi, V. Kohli, V. Chamola, and F.R. Yu, "A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems," *Digital Communications And Networks*, vol. 9, no. 5, pp.1113-1122, 2023. doi: 10.1016/j.dcan.2022.06.018.
- [31] P. Singh, G.S. Gaba, A. Kaur, M. Hedabou, and A. Gurtov, "Dew-cloud-based hierarchical federated learning for intrusion detection in IoMT," *IEEE journal of biomedical and health informatics*, vol. 27, no. 2, pp. 722-731, 2022. doi: 10.1109/JBHI.2022.3186250.
- [32] G. Manoharam, M.S.M. Kasihmuddin, S.N.F.M.A. Antony, N.A. Romli, N.A. Rusdi, S. Abdeen, and M.A. Mansor, "Log-Linear-Based Logic Mining with Multi-Discrete Hopfield Neural Network," *Mathematics*, vol. 11, no. 9, p.2121, 2023. doi: 10.3390/math11092121.
- [33] M.S.M. Kasihmuddin, N.A. Romli, G. Manoharam, M.A. and Mansor, "Multi-unit discrete hopfield neural network for higher order supervised learning through logic mining: optimal performance design and attribute selection," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 5, p.101554, 2023, doi: 10.1016/j.jksuci.2023.101554.
- [34] S.Z.M. Jamaludin, N.A. Romli, M.S.M. Kasihmuddin, A. Baharum, M.A. Mansor, M.F. and Marsani, "Novel logic mining incorporating log linear approach," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10, pp. 9011-9027, 2022, doi: 10.1016/j.jksuci.2022.08.026.
- [35] N.E. Zamri, M.A. Mansor, M.S.M. Kasihmuddin, S.S. Sidik, A. Alway, N.A. Romli, Y. Guo, and S.Z.M. Jamaludin, "A modified reverse-based analysis logic mining model with weighted random 2 satisfiability logic in discrete hopfield neural network and multi-objective training of modified niched genetic algorithm," *Expert Systems with Applications*, vol. 240, p.122307, 2024, doi: 10.1016/j.eswa.2023.122307.
- [36] N.E. Zamri, S.A. Azhar, M.A. Mansor, A. Alway, and M.S.M. Kasihmuddin, "Weighted random k satisfiability for k= 1, 2 (r2SAT) in discrete Hopfield neural network," *Applied Soft Computing*, vol. 126, p.109312, 2022. doi: 10.1016/j.asoc.2022.109312.





**BIOGRAPHIES OF AUTHORS**

**Valavan Woothukadu Thirumaran**     is doing PhD (CSE) in Bharath Institute of Higher Education and Research (BIHER), Chennai. He has completed BES in Electronics Science in Alpha College, University of Madras in the year 2002. He has completed MSc in Information Technology in Shree Chandrababhu Jain College, University of Madras in the year 2006 with University 17<sup>th</sup> Rank. He has completed MPhil in Computer Science in Periyar University in the year 2007. He has completed MCA in Periyar University in the year 2009. He has completed ME degree in Computer science and Engineering in Sriram Engineering College, Anna University in the year 2012. He has 15 years of Industry experience. He can be contacted at email: wtvalavan@gmail.com.



**Nalini Joseph**     is working as a professor in Dept of CSE at Bharath Institute of science and technology, Bharath University, Chennai. She has completed PhD degree in Computer science and Engineering in Anna University in the domain of Wireless Sensor Networks. She completed ME degree in Computer science and Engineering in SRM Engg College, Madras University in the year 2000 and guiding scholars who are pursuing their PhD degree. She has 33 years of Experience (22.5yrs -Teaching and 10.5-Industry experience. Her areas of interest are WSN, Machine learning, Computer Networks, Deep learning and Internet of things. She has published many papers in international and national journals. She has presented papers in national and international conferences as well. She acted as guest speaker in FDPs and workshops organized by various reputed Institutions. She has organized workshops, seminars, Faculty Development Programmes, International conferences. She has worked in administrative cadre like HOD, DEAN(Academics,) Vice Principal, Principal In Engg college. Her strength is slow learners Training and Motivating them and Training Young faculty. She can be contacted at email: nalinijoseph.cse.cbcs@bharathuniv.ac.in.



**Umarani Srikanth**     is working as a professor in Dept of CSE at Panimalar Engineering College, Chennai. She has completed PhD degree in Computer science and Engineering in Anna University in the domain of soft computing. She completed ME degree in Computer science and Engineering in National Institute of Technology, Trichy in the year 1996 and guiding scholars who are pursuing their PhD degree. She has 26 years of teaching experience. Her areas of interest are Machine learning, Data Analytics, soft computing, and Internet of things. She has published many papers in international and national journals. She has presented papers in national and international conferences as well. She acted as guest speaker in FDPs and workshops organized by various reputed Institutions. She has organized plenty of workshops, seminars, Faculty Development Programmes, National and international conferences. She can be contacted at email: umaranisrikanth@gmail.com.