Grenze International Journal of Engineering and Technology, January Issue

Distributed Attack Detection Framework in Cloud Environment using Hybrid Activation-based BiLSTM with Virus Colony Optimization

Balakrishna Kancherla[1], Immadisetti Yamuna[2] and Shaik Zahira[3]
[1]Assistant Professor, Dept. of Computer Science and Engineering. RVR & JC College of Engineering, Guntur, Andhra Pradesh-522019
Email: Kancherla.Balakrishna@Gmail.Com
[2]Dept. of Artificial Intelligence and Machine Learning, Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh 522508
Email: Immadisetti1234@Gmail.Com
[3]Dept. of CSE(AI&ML), Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh 522508
Email: Zahirashaik4545@Gmail.Com

***Abstract*— One of the key advantages of cloud computing is its ability to swiftly roll out services and applications. Cyberattacks within cloud storage have dramatically increased since smart devices are becoming more and more popular and the many security flaws in networks. Modern society is becoming increasingly dependent on information and communication technology, which is increased its susceptibility to many types of cyberattacks. Distributed denial-of-service (DDoS) assaults are difficult to halt by blocking a single source since they overwhelm victims with traffic from several sources. Web servers, network equipment, and application servers are the targets of these attacks.**

In this research, a Hybrid Activation-based Bidirectional Long Short-Term Memory (HA-BiLSTM) model improved by Virus Colony Optimization (VCO) is employed to create a framework for distributed attack detection in cloud environments. This system improves the efficacy and precision of cyber threat detection, hence mitigating the security threats associated with cloud computing. It uses K-Nearest Neighbors (KNN) to handle missing values and Generative Adversarial Networks (GANs) to balance data, increasing the accuracy of threat identification and reducing cloud security issues. The results indicate that the HA-BiLSTM-VCO model performs better than current approaches across a range of criteria. The HA-BiLSTM-VCO model effectively uses the BoT-IoT and UNSW datasets to show enhanced performance metrics. Regarding the BoT-IoT data, it attains 97.35% accuracy, 97.52% specificity, and 97.19% sensitivity. On the UNSW dataset, it also demonstrates 97.08% sensitivity, 97.80% specificity, and 97.44% accuracy. Overall, by offering a strong defense against dispersed attacks and advancing the continuous development of cloud security frameworks, this proposed technique greatly improves security measures in cloud environments.

***Index Terms*— Cloud Computing; Distributed Attack Detection; Bidirectional Long Short-Term Memory; Activation functions; Virus Colony Optimization.**

Grenze ID: 01.GIJET.12.1.48

I. INTRODUCTION

Cloud Computing (CC) enables on-demand delivery of hardware and software resources through data centers, offering scalability, flexibility, and cost-efficiency via virtualization and shared environments. However, device heterogeneity, multi-vendor infrastructures, and growing ICT-IoT ecosystems introduce severe security challenges. Enterprises and research institutions are prime targets of distributed network attacks, particularly reconnaissance and DDoS, which cause financial, reputational, and operational damage. Traditional machine learning-based intrusion detection systems (IDS) have attempted to counter these threats but suffer from low detection rates, poor performance on unseen attack variations, and limited feature engineering.

To overcome these challenges, we propose the Hybrid Activation-based BiLSTM with Virus Colony Optimization (HA-BiLSTM-VCO) model. This approach integrates KNN imputation for preprocessing with a BiLSTM framework enhanced by hybrid activation and VCO optimization. The proposed model effectively addresses data imbalance while improving accuracy, sensitivity, and specificity in detecting distributed attacks within cloud environments. Experimental validation shows its superiority over conventional IDS, establishing a new benchmark for cloud cybersecurity.

II. LITERATURE SURVEY

A review of the literature on the instruments available for identifying distributed attacks in the cloud is given in this section. It clarifies low rates as well. Jullian, O., et al. [9] created a Distributed Deep Learning (DL) Framework to identify several cybersecurity threats within a single defense system. They tested two models: LSTM and a feedforward neural network. With an accuracy of up to 99.95%, the results demonstrated remarkable effectiveness. Scalability issues in various IoT scenarios are revealed by the framework's difficulties with the NSL-KDD dataset. Pasha, M.J., et al. [5] presented the Hybrid Approach for Low-Rate DDoS Detection (HA-LRDD), an AI-powered system that combines a deep autoencoder with a Deep Convolutional Neural Network (CNN). The Dynamic Low-Rate DDoS Mitigation (DLDM) technique was also created by them to lessen the impact of attacks after they are detected. The suggested method is efficient, but it leaves low-rate DDoS attacks unabated, underscoring the need for additional optimization. Talpur, F., et al. [10] offer a novel method that combines machine learning methods with evolutionary optimization algorithms. It introduces the optimization methods XGB-GA, RF-GA, and SVM-GA and employs EAs with TPOT-Genetic Programming to identify DDoS attacks. High accuracy was attained by further optimizing the models with EAs. Still, the approach mainly concentrates on predicting certain resources, necessitating additional verification on disk consumption, cost-effectiveness, network performance, and energy efficiency through a variety of multivariate datasets. Ouhssini, M., et al. [1] present the DeepDefend framework as a novel approach to cloud-based DDoS attack detection and mitigation. The architecture can significantly improve cloud security, which is essential in the connected digital world of today. But to increase threat mitigation effectiveness and guarantee more reliable defense against changing cyber threats, it needs faster DDoS detection and more robust preventive systems.

A. Challenges

The difficulties encountered by current attack detection research are described below:

- The Distributed DL framework [9] performance could deteriorate if fog nodes function in different settings with different data types, which could result in inconsistent attack detection.
- The DL methods, which can be computationally costly and may need a significant quantity of training data, are a major component of the HA-LRDD [5] model.
- Because of its emphasis on predicting certain resources, the ML-Based Evolutionary Algorithms [10] approach necessitates additional assessment of disk usage, cost-effectiveness, network performance, and energy efficiency utilizing a variety of multivariate datasets for validation.
- The DeepDefend [1] system is limited to its dependence on faster DDoS detection and more robust prevention techniques, which necessitates appropriate cloud-based time series datasets.

B. Problem statement

Although there are many benefits to the move to cloud computing, there are also serious security risks. Organizations are more susceptible to sophisticated cyber threats such as data breaches, illegal access, and different types of cyberattacks as they depend more and more on cloud services. The security environment is made more difficult by the dispersed nature of cloud infrastructures, the growth of IoT devices, and remote workers. Sensitive data is being insufficiently protected by traditional perimeter-based security methods, which depend on a well-defined border. Therefore, there is an urgent need for innovative solutions to effectively

identify and handle threats in real-time. One such approach is the proposed HA-BiLSTM model, which attempts to address these problems in cloud systems and assess how well they detect and react to cyber threats.

III. Hybrid Activation-based Bidirectional Long Short-term Memory with Virus Colony Optimization

The proposed investigation objectives to detect threats in cloud contexts uses the BOT-IoT and UNSW datasets as their primary data sources. To handle missing values, K-Nearest Neighbors (KNN) imputation is utilized. It estimates them from related data points to ensure data quality. A Generative Adversarial Network (GAN) is used to produce synthetic samples to balance the dataset and prevent the model from favoring more prevalent attack types. Following preprocessing, the information is entered into a BiLSTM model, which is then improved by hybrid activation functions, including SoftMax, Tanh, Sigmoid, and ReLU. These features increase the model's learning effectiveness and aid in capturing a variety of attack patterns. BiLSTM parameters are adjusted using the metaheuristic algorithm Virus Colony Optimization (VCO) to further refine the model and increase its accuracy in identifying security risks. Additional information is used to assess the trained model's capacity to distinguish between harmless and malicious activity during the prediction phase. To improve cybersecurity threat identification and response in IoT and cloud contexts, the model learns from input and predicts whether it reflects an attack or normal activity. The system model is displayed in Fig. 1.

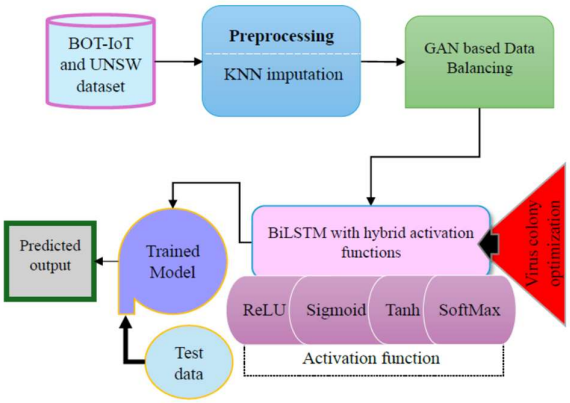


Fig. 1. Distributed Attack Detection System Model

A. Input Data for Attack Detection

The BOT-IoT [11] and UNSW [12] datasets are used for attack detection with the dimension of [N×42]. These datasets are mathematically represented as follows:

$$D = \{Z_{j,k}; 1 \le j \le x, 1 \le k \le y \qquad (1)$$

Here D refers the datasets, j represents the j^{th} individual observation data points, $k$refers the k^{th}attributes of the data, $Z_{j,k}$ denotes the k^{th} attribute in the j^{th} individual data point of the database $x \times y$. After that, the dataset is sent to the preprocessing module, where any missing values are filled using KNN imputation.

B. K-Nearest Neighbors for data pre-processing

KNN [13] is a popular instance-based, slow-learning method in data mining. To estimate missing data, it chooses the k closest neighbors according to a distance metric and uses the mean for numerical characteristics or mode for categorical features to anticipate missing values. KNN imputation works well with datasets that have a lot of missing values.

$$kn = \sqrt{wt \times \left(mi_{j,k} - D_{j,k}\right)^2}, j, k \in fe \qquad (2)$$

Here kn denotes the K-Nearest Neighbors, $D_{j,k}$ represents the datasets, mi refers to the missing values in the datasets, wt stands as the weight, and fe indicates the features of the datasets. The KNN is preprocessed by imputing missing data, resulting in a data dimension of$[N \times 42]$, which is then fed into the GAN.

C. Generative Adversarial Network for Data Balancing

A deep learning technique for creating synthetic data is a Generative Adversarial Network (GAN) [14], which is made up of two network models, namely Generator Ge and the Discriminator Di. The Generator creates a generated sample using random noise as input, and the Discriminator receives both a generated sample and a sample of real training data. The two are trained in a continuous cycle where the Generator improves at creating more realistic samples and the Discriminator improves at differentiating between real and fake data. Eventually, this competition results in the creation of extremely realistic samples.

$$H = E[log\ D\ E(kn)] + E\left[log\left(1 - DE\left(Ge(\mathbb{Z})\right)\right)\right] \quad (3)$$

Here H evaluates how well the discriminator can differentiate between produced and actual samples, $log\ D\ E$shows the log-likelihood that actual data will be accurately classified by the discriminator, $log\left(1 - DE\left(Ge(\mathbb{Z})\right)\right)$ indicates the log-likelihood that the discriminator will properly categorize generated samples as false, whereas kn indicates the result of the KNK imputation.

Next, the model receives the balanced data with the size of $[N \times 42]$, which has been trained to identify attacks. Every single data sample must have a label during the training phase. The following represents the class sample:

$$D_1 = g_j = \begin{cases} 0, Normal \\ 1, DDoS \\ 2, DoS \\ 3, Re\ c\ onnaissance \\ 4, Theft \end{cases} \quad (4)$$

$$D_2 = g_j = \begin{cases} 0, Normal \\ 1, Analysis \\ 2, Backdoors \\ 3, DoS \\ 4, Exploits \\ 5, Fuzzers \\ 6, Generic \\ 7, Re\ c\ onnaissance \\ 8, Shellcode \\ 9, Worms \end{cases} \quad (5)$$

Here D_1 refers to the BOT-IoT dataset, D_2 stands for the UNSW dataset, and $g_j$determines the label for the j^{th} sample in the dataset.

D. Hybrid Activation-based Bidirectional Long Short-Term Memory with Virus Colony Optimization

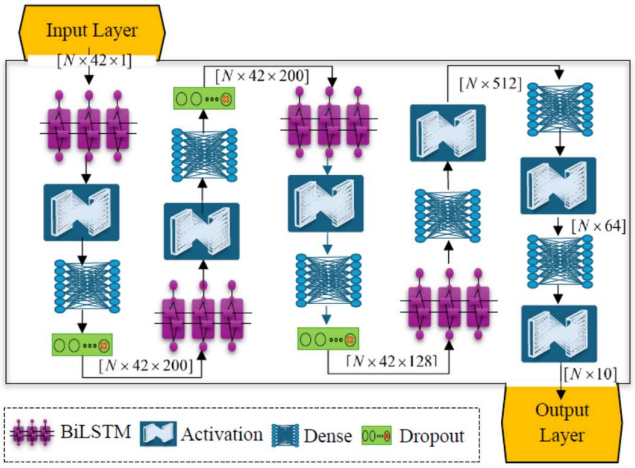


Fig. 2. The architecture of the HA-BiLSTM model

The model architecture in Fig. 2 is intended for cybersecurity distributed attack detection. It uses several BiLSTM layers to capture temporal dependencies from both past and future contexts, and it processes input data with a shape of $[N \times 42 \times 1]$. The model starts with a BiLSTM layer and then adds activation and dropout layers to improve generalization with the dimension $[N \times 42 \times 200]$. The LSTM units are progressively decreased from $[N \times 42 \times 200]$ to $[N \times 42 \times 128]$ by repeating this pattern. Following the sequential feature extraction process, the compressed features are processed by a Dense layer and converted into a $[N \times 512]$ representation. The dimensions are further reduced $[N \times 64]$ by further activation layers and another Dense layer. Finally, the output of the dense layer, which assigns probabilities across 10 attack classes using a SoftMax activation function, is $[N \times 10]$. The proposed HA-BiLSTM model is ideal for identifying dispersed cyberattacks through the examination of sequential patterns in log data or network traffic. It efficiently detects irregularities and groups them into various assault types. It is very good at identifying complex and dynamic dangers because of its deep hierarchical structure, which captures both short-term and long-term dependencies.

Activation Function

An activation function is a type of mathematical function, which controls the output value of each neuron in neural layers, is essential for controlling information flow and network gradients. Detailed information on popular activation functions like SoftMax, Tanh, Sigmoid, and ReLU is provided below [15].

Sigmoid: It is the most widely utilized activation function due to its non-linearity. The sigmoid function transforms the numbers from 0 to 1. It could be characterized as:

$$AT_1 = a(b) = \frac{1}{c^{-b}} \tag{6}$$

Here $a(b)$ represents the input to the function and c refers the base of the natural logarithm. The sigmoid function is a smooth S-shaped function that is continuously differentiable. The function's derivative is:

$$AT_1 = a'(b) = 1 - smd(b) \tag{7}$$

Where smddenotes the sigmoid function. Additionally, because the sigmoid function is not symmetric about zero, all of the neuronal output values will have the same sign. The sigmoid function can be scaled to improve this problem.

Tanh: The sigmoid function is comparable to the tanh function, also known as the Hyperbolic Tangent function. However, it is symmetrical around the origin. As a result, the outputs from previous layers that are sent as input to the next layer have different signs. It can be defined as follows:

$$AT_2 = a(b) = 2smd(2b) - 1 \tag{8}$$

ReLU: Rectified Liner Unit (ReLU) is a popular non-linear activation function in neural networks. The benefit of using the ReLU function is that not all neurons are active at the same time, meaning that a neuron will only be deactivated if the output of the linear transformation is zero. The ReLU function is defined mathematically as follows:

$$AT_3 = a(b) = max(0, b) \tag{9}$$

SoftMax: The SoftMax function combines many sigmoid functions. Given that a sigmoid function yields values between 0 and 1, these can be regarded as the probabilities of specific class data points. In contrast to sigmoid functions, which are employed in binary classification, the SoftMax function can be applied to multiclass classification issues. The function returns the probability for each class's data point. It can be stated as:

$$AT_4 = sf(d)_i = \frac{e}{\sum_{f=1}^{F} e^{df}} \tag{10}$$

Here sf, it stands for the SoftMax function, $(d)_i$denotes the input vector's i^{th} element, Frepresents the overall classes there are in the classification problem, and $\sum_{f=1}^{F} e^{df}$shows the sum of the exponentials of all elements in the input vector d where $i = 1, \ldots, F$. The number of classes in the target will correspond to the number of neurons in the network's output layer when we construct a network or model for multiple-class classification. The activation function $AT$is finalized as,

$$AT = \{\|AT_1, AT_2, AT_3, AT_4, AT_5\} \tag{11}$$

The above-mentioned equation shows the hybrid activation function which is integrated with the BiLSTM model are discussed in the below sections.

Bidirectional Long Short-Term Memory

BiLSTM considers the past and future network traffic characteristics [16]. The forward LSTM cell state and the backward LSTM cell state are the two parts of its hidden layer. The network traffic data first travels through the input layer before entering the hidden layer, where it participates in both forward and backward calculations. The output layer receives the results from the hidden layer and creates the final output by combining the forward and backward LSTM outputs based on predetermined weights. Figure 2 shows the structure of the BiLSTM network.

$$fr(ti) = \tau\left(T_{fr} hi_{ti-1} + P_{fr}\eta_{ti} + ba_{fr}\right) \quad (12)$$

$$in(ti) = \tau(T_{in} hi_{ti-1} + P_{in}\eta_{ti} + ba_{in}) \quad (13)$$

$$if(ti) = AT\left(T_{if} hi_{ti-1} + P_{if}\eta_{ti} + ba_{if}\right) \quad (14)$$

$$ou(ti) = \tau(T_{ou} hi_{ti-1} + P_{ou}\eta_{ti} + ba_{ou}) \quad (15)$$

Here $in(ti)$, $fr(ti)$, and $ou(ti)$ refers to the input gate, forget gate, and output gate at a time ti, $f(ti)$represents the initial feature of hi_{ti-1}a time ti, and AT indicates the hybrid activation which is evaluated in Equation (11).

$$hi(ti) = ou(ti)\Phi AT\left(ci(ti)\right) \quad (16)$$

Equation (16), which is obtained from the output gate $ou(ti)$ and the cell state $ci(ti)$, yields the final hidden state $hi(ti)$ at the moment ti. Where the Hadamard product is denoted by Φ. The Categorical Cross-Entropy (CCE) loss function is one of the hyperparameters of the proposed HA-BiLSTM-VCO model. By reducing the CCE loss, the deep learning model gives the right class a larger probability and the wrong classes a lower probability. The loss function can be represented mathematically as follows:

$$CCE = -\frac{1}{\ell}\sum_{sa=}^{\ell} \sum_{pp=1}^{\ell o} \Theta_{sa}\, log\left(\hat{\theta}_{sapp}\right) \quad (17)$$

Here CCE stands for the Categorical Cross Entropy, ℓo illustrates the number of classes, $\hat{O}_{sapp}$shows the predicted value of the sa^{th} sample and pp^{th}class, Θ_{sspp} denotes the expected output of the sa^{th}sample and pp^{th}class, Σrepresents the summation, and ℓ states the number of samples. To get the best results, the optimization procedure is used to train the model and is applied above the HA-BiLSTM model's dense layer. The next section explains the process of the VCO algorithm.

E. Virus Colony Optimization

A nature-inspired optimization algorithm called Virus Colony Optimization (VCO) [17] imitates the virus's infection and dispersion tactics. It uses a dual-population model with colonies of host cells and viruses. The technique uses the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for exploitation and Gaussian random walks for exploration. Furthermore, VCO has an immune response mechanism that promotes diversity by allowing fewer efficient viruses to emerge. These characteristics make VCO useful for both limited and unconstrained optimization problems by enhancing convergence rates and increasing the accuracy of identifying optimal solutions.

Initialization: To provide a diverse optimization starting point, the VCO method initializes a dual-population model of host cells and viruses, establishing solution counts and positions using random values.

$$Z_{m,n} = op_n + s(qp_n - op_n) \quad (18)$$

where, $m = 1,2,\ldots,X$, and $n = 1,2,\ldots,r$. Here X denotes the number of solutions, $r$indicates how many dimensions there are in the search space, Z_m stands for the initial position vector of m^{th}a solution, $s$and indicates the random numbers of intervals $(0,1)$.

$$Z = \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_m \\ \vdots \\ \rho_X \end{bmatrix} = \begin{bmatrix} \rho_{1,1} & \cdots & \rho_{1,n} & \cdots & \rho_{1,r} \\ \vdots & & & & \\ \rho_{m,1} & \cdots & \rho_{m,n} & \cdots & \rho_{m,r} \\ \vdots & & & & \\ \rho_{X,1} & \cdots & \rho_{X,n} & \cdots & \rho_{X,r} \end{bmatrix} \quad (19)$$

Fitness Function: In a VCO, the fitness weight value is typically used to manage the maximum accuracy. The algorithm determines the fitness weight based on the accuracy as evaluated in equation (20).

$$I = \left(\frac{tu_{ps}+tu_{ng}}{tu_{ps}+tu_{ng}+fa_{ng}+fa_{ps}}\right) \quad (20)$$

Here I denotes the fitness measures, tu_{ng} , shows true negative value, and fa_{ps} indicates false positive value.
Phase (i) Inquisition phase: Using local and global search probability, the inquisition phase iteratively modifies solution placements based on the most well-known solutions for the best possible exploration and exploitation.
Case (i) antecedent based updated phase: $[J \geq 0.5]$, Probabilities selected by both local and global search provide the basis for the iteration $J = \left(1 - \frac{u}{u_{mx}}()\right)$, here $u$indicates the current iteration and $u_{mx}$is the maximum iteration. The position updated equation as,

$$Z_m^{(u+1)} = Gau(Z_{bst}^u, \kappa) + \left(s_1 \cdot Z_{bst}^u - s_2 \cdot Z_m^{(u)}\right) \quad (21)$$

Here $Z_m^{(u+1)}$ stands for the current position vector of the m^{th} solution, Z_{bst}^uindicates the best solution, s_1, s_2 are the random values between $(0,1)$, Gaurepresents the Gaussian distribution, κdenotes the Gaussian parameter, and u refers to the current iteration.

$$\kappa = \frac{log(u)}{u}\left(Z_m^{(u)} - Z_{bst}^{(u)}\right) \quad (22)$$

The $Gaussian(Z_{bst}^u, \kappa)$ returns a new position, computed by sampling from a Gaussian distribution centered around Z_{bst} with a standard deviation of κ. The optimal solution and the current position are the only factors that affect the position equation above. The best answer might not be close to the global optimal solution at first. The algorithm can so fail to arrive at the best answer. Because of this restriction, the algorithm is less able to investigate various areas of the search space, which makes it more difficult to discover the best answer and may even trap it in local optima. To address this, the equation is hybridized with fractional theory.

$$Z_m^{(u+1)} = Gau\left(Z_{bst}^{(u)}, \kappa\right) + s_1 \cdot Z_{bst}^{(u)} - s_2 \cdot Z_m^{(u)} \quad (23)$$

$$Z_m^{(u+1)} - Z_m^{(u)} = Gau\left(Z_{bst}^{(u)}, \kappa\right) + s_1 \cdot Z_{bst}^{(u)} - s_2 \cdot Z_m^{(u)} - Z_m^{(u)} (24)$$

The effectiveness of numerous algorithms is greatly enhanced in domains includes pattern matching, edge detection, filtering, curve fitting, and modeling by utilizing Fractional Calculus (FC) theory, a subfield of applied mathematics. When FC is applied to the equation's Left-Hand Side (L.H.S.), it can be assessed in the manner described below:

$$\Upsilon^{\alpha}\left(Z_m^{(u+1)}\right) = Gau\left(Z_{bst}^{(u)}, \kappa\right) + s_1 \cdot Z_{bst}^{(u)} - s_2 \cdot Z_m^{(u)} - Z_m^{(u)} (25)$$

$$Z_m^{(u+1)} - \alpha Z_m^{(u)} - \frac{1}{2}\alpha Z_m^{(u+1)} - \frac{1}{6}\alpha(1-\alpha)Z_m^{(u-2)} = Gau\left(Z_{bst}^{(u)}, \kappa\right) + s_1 \cdot Z_{bst}^{(u)} - (1+s)Z_m^{(u)} \quad (26)$$

$$Z_m^{(u+1)} = Gau(Z_{bst}^u, \kappa) + s_1 \cdot Z_{bst}^u - (1+s)Z_m^{(u)} + \alpha Z_m^{(u)} + \frac{1}{2}\alpha Z_m^{(u+1)} + \frac{1}{6}\alpha(1-\alpha)Z^{(u-2)} \quad (27)$$

$$Z_m^{(u+1)} = Gau(Z_{bst}^u, \kappa) + s_1 \cdot Z_{bst}^u - (1+s-\alpha)Z_m + \frac{1}{2}\alpha Z_m^{(u+1)} + \frac{1}{6}\alpha(1-\alpha)Z^{(u-2)} \quad (28)$$

Where $Z_m^{(u-2)}$ denotes the position vector for the m^{th}solution in the $(u-1)^{th}$. Using FC theory, which updates depending on historical, current, and best solution locations, the above equation is hybridized by integrating the position values of the solution from earlier iterations. By using this method, the algorithm can avoid local optima and more effectively explore other areas of the search space. It also speeds up the optimization algorithm's convergence and directs the search toward the best answers.
Case (ii) delving phase: $[J < 0.5]$, the update solution of this phase begins by satisfying the above condition.

$$Z_m^{(u+1)} = Z_{mn}^{(u+1)} + \sigma_m^{(u)} \times X_m\left(0, v^{(u)}\right) \quad (29)$$

Here $X_m\left(t, v^{(u)}\right)$ represents the normal distribution with mean z, $v^{(u)}$ denotes the covariance matrix, $\sigma_m^{(u)} > 0$ shows the step size parameter with u^{th}iteration, and mn refers to the mean value. Initially Z_{mn} is initiated by,

$$Z_{mn}^o = \frac{\sum_{m=1}^{X} Z_m}{X} \quad (30)$$

Here Z_{mn}^o refers to the mean value of the position vector for all solutions.

$$Z_{mn}^{o} = \frac{1}{\lambda}\sum_{m=1}^{\lambda} t_m \cdot Z_m^{\lambda bst} \quad (31)$$

$$t_m = \frac{ow(\lambda+1)}{\sum_{n=1}^{\lambda}(ow(\lambda+1)-ow(n))} \quad (32)$$

Here $\lambda = [\frac{X}{2}]$, where λ indicates the number of best solutions, $t_m$represents the combination weights, and $m$denotes the index of the best individual.

Phase (ii) Rank-based updation phase:

Calculate performance ranks based on the fitness values of the solutions to determine their effectiveness.

$$K_{rnk(m)} = I(Z_m) \quad (33)$$

$$K_{rnk(m)} = \frac{[I(Z_1)+I(Z_2)+\ldots+I(Z_X)]}{X} \quad (34)$$

Here K_{rnk} stands for the performance rank, and $I(Z_m)$ denotes the fitness value of m^{th} the solution. The updated position is

$$Z_m^{(u+1)} = \begin{cases} Z_{\lambda}^{(u)} - rnd \cdot \left(Z_{\hbar}^{(u)} - Z_m^{(u)}\right), if K_{rnk(ag)} > K_{rnk(m)} \\ Z_m^{(u)}, otherwise \end{cases} \quad (35)$$

Here $\lambda, \hbar$ represents the random selected index from $[1,2,\ldots,X], \lambda \neq \hbar$, and rnd indicates the random value within the range of $(0,1)$, and agrefers to the average value. The average fitness solution and the worst fitness solution are contrasted in the equation above. A random answer updates the worst solution's position if it is below average, increasing the current solution's fitness score. By strengthening weaker solutions, this method helps the algorithm avoid achieving premature convergence. Consequently, it raises the likelihood of locating the global optimum, even for intricate issues.

Termination: After evaluating the best solution, the iteration gets terminated.

IV. Results and Discussion

This section discusses the performance and comparative analysis of the data obtained from the HA-BiLSTM-VCO model, which is used to detect distributed attacks.

A. Experimental Setup

The proposed configuration is tested using the Python software on Windows 10 with an Intel Core CPU and 16 GB of RAM memory.

B. Dataset Description

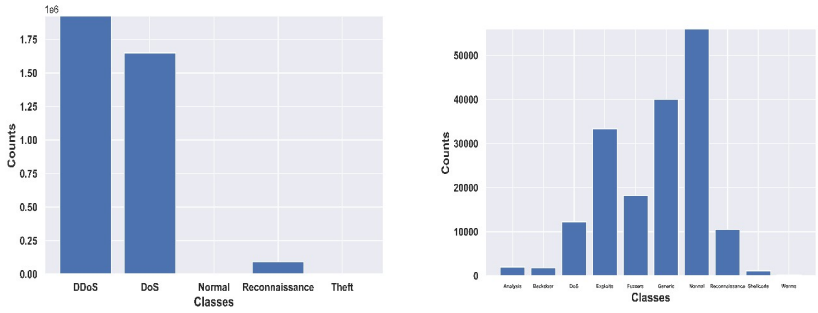
The BoT-IoT dataset [11] was created at UNSW Canberra's Cyber Range Lab, which replicates a realistic network environment with both typical and botnet traffic. Numerous attack types are covered, including OS and Service Scans, Data Exfiltration, DDoS, DoS, and Keylogging. By protocol, DDoS and DoS assaults are further classified.

The dataset can be found in CSV, pcap, and argus files, among other forms. With almost 72 million records, the original pcap files are 69.3 GB in size. However, 16.7 GB of flow traffic was recovered and saved in CSV format. Organizing the files by assault category facilitates tagging. Raw network packets for the UNSW-NB15 dataset were generated using the UNSW Canberra Cyber Range Lab's IXIA PerfectStorm program [12]. The dataset was designed to incorporate both simulated modern attack behaviors and actual modern network operations.

The dataset includes nine attack categories, including fuzzers, analysis, backdoors, DoS, exploits, generic, reconnaissance, shellcode, and worms. The UNSW-NB15_features.csv file contains a detailed list of the twelve algorithms and 49 features that were generated using tools such as Argus and Bro-IDS. In total, 100 GB of raw traffic was captured using the tcpdump program.

C. Dataset Visualization

Different class numbers are displayed graphically in Fig. 4 for the BOT-IoT and UNSW datasets. The horizontal axis displays the class conditions, while the vertical axis displays the counts. By measuring the count rate across dataset classes, information about their frequency and distribution may be obtained. This representation facilitates efficient analysis of the features and variances of the dataset.



BOT-IoT dataset UNSW dataset

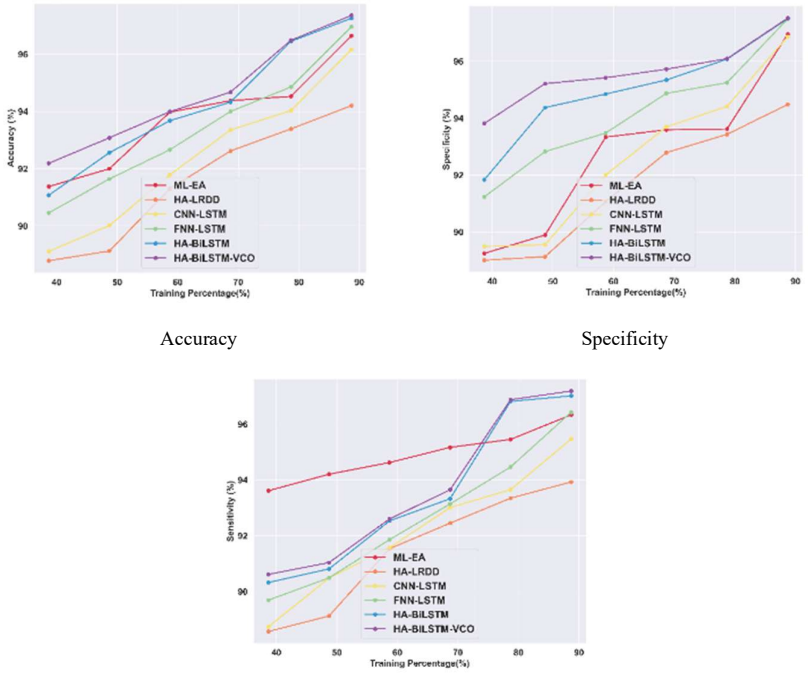
Fig. 4. Data visualization of BOT-IoT and UNSW dataset

D. Comparative Analysis

This section presents a comparison of the HA-BiLSTM-VCO model with the current models: FNN-LSTM [9], HA-LRDD [5], ML-EA [10], CNN-LSTM [1], and with the HA-BiLSTM model. The analysis is based on the UNSW Dataset [12] and the BOT-IoT Dataset [11].

Comparative Analysis with BOT-IoT Dataset

Using the BoT-IoT dataset, the suggested HA-BiLSTM-VCO model is contrasted with other approaches, such as FNN-LSTM, HA-LRDD, ML-EA, and CNN-LSTM, as depicted in Fig. 5. With an accuracy of 97.35%, the HA-BiLSTM-VCO model outperformed the FNN-LSTM is 96.96%, HA-LRDD is 94.21%, ML-EA is 96.64%, CNN-LSTM is 96.16%, and HA-BiLSTM is 97.26% models for a 90% training percentage. With 90% training data, the HA-BiLSTM-VCO model outperforms the ML-EA by 0.9% with a sensitivity of 97.19%, 3.4% better than the HA-LRDD, 1.8% better than the CNN-LSTM, 0.8% better than the FNN-LSTM, and 0.2% better than the HA-BiLSTM models. Additionally, using 90% training data, the HA-BiLSTM-VCO model achieves a specificity of 97.52%, outperforming the current models by 0.58% over ML-EA, 3.11% over HA-LRDD, 0.67% over CNN-LSTM, 0.03% over FNN-LSTM, and 0.02% over HA-BiLSTM.



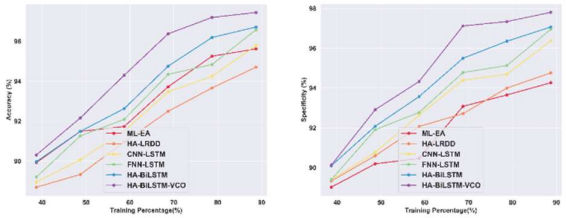
Accuracy Specificity

Sensitivity

Fig.5. Comparative Analysis with BOT-IoT Dataset

Comparative Analysis with UNSW Dataset

Fig. 6 compares the proposed HA-BiLSTM-VCO model with other approaches, such as FNN-LSTM, HA-LRDD, ML-EA, and CNN-LSTM, using the UNSW dataset. With 90% training data, the HA-BiLSTM-VCO model's sensitivity is 97.08%, which includes improvements of 0.1% over the ML-EA, 2.5% over the HA-LRDD, 1.9% over the CNN-LSTM, 0.9% over the FNN-LSTM, and 0.7% over the HA-BiLSTM models. Moreover, using 90% training data, the HA-BiLSTM-VCO model achieves a specificity of 97.80%, outperforming the current models by 94.27% at ML-EA, 94.76% at HA-LRDD, 96.37% at CNN-LSTM, 96.95% at FNN-LSTM, and 97.07% at HA-BiLSTM. At a 90% training percentage, the accuracy of the HA-BiLSTM-VCO model was 97.44%, greater than that of the FNN-LSTM (96.58%), HA-LRDD (94.70%), ML-EA (95.62%), CNN-LSTM (95.78%), and HA-BiLSTM (96.71%) models.



Accuracy Specificity

Fig. 6. Comparative Analysis with UNSW Dataset

E. Comparative Discussion

For distributed attack detection, the proposed HA-BiLSTM-VCO model is compared to alternative techniques, including CNN-LSTM [1], ML-EA [10], HA-LRDD [5], and FNN-LSTM [9]. Scalability problems, poor performance in completely preventing low-rate DDoS attacks, excessive disk utilization, cost inefficiency, network performance challenges, and energy inefficiency are only a few of the drawbacks of these current techniques. Furthermore, they mostly rely on quicker DDoS detection, which reduces their efficacy when tested on various multivariate datasets. These restrictions are addressed by the HA-BiLSTM-VCO model, which correctly detects network intrusions while resolving problems with complexity and data imbalance. Computational difficulties are resolved and convergence is improved by integrating the VCO method. Table 1 illustrates how well the HA-BiLSTM-VCO model performs overall when compared to current methods.

TABLE I. COMPARATIVE DISCUSSION FOR THE PROPOSED HA-BILSTM-VCO

Analysis/Methods		ML-EA	HA-LRDD	CNN-LSTM	FNN-LSTM	HA-BiLSTM	HA-BiLSTM-VCO
BOT-IoT dataset	*Accuracy (%)*	96.64%	94.21%	96.16%	96.96%	97.26%	97.35%
	Specificity (%)	96.96%	94.49%	96.87%	97.50%	97.50%	97.52%
	Sensitivity (%)	96.33%	93.93%	95.47%	96.43%	97.01%	97.19%
UNSW dataset	*Accuracy (%)*	95.62%	94.70%	95.78%	96.58%	96.71%	97.44%
	Specificity (%)	94.27%	94.76%	96.37%	96.95%	97.07%	97.80%
	Sensitivity (%)	97.01%	94.64%	95.20%	96.20%	96.36%	97.08%

V. CONCLUSION

The emerging field of cloud cybersecurity is greatly advanced by the proposed Distributed Attack Detection Framework that uses HA-BiLSTM linked with VCO. This approach tackles important issues including class imbalance, high dimensionality, and the complexity of the IoT as the cyber threat landscape changes. This novel

technique offers a strong basis in cloud security methodologies in addition to improving the accuracy of distributed threat detection. All things considered, the present investigation significantly advances the subject of cloud computing security by providing a potential path for future improvements in detecting frameworks and, eventually, protecting sensitive data from malevolent attacks. The HA-BiLSTM-VCO model demonstrates improved performance metrics by efficiently utilizing the BOT-IoT and UNSW datasets. The BOT-IoT dataset yields 97.35% accuracy, 97.52% specificity, and 97.19% sensitivity. Additionally, it displayed an accuracy of 97.44%, specificity of 97.80%, and sensitivity of 97.08% on the UNSW dataset. For better detection efficacy, future studies can evaluate additional datasets, enhance model performance, and investigate hybrid optimization techniques.

REFERENCES

[1] M. Ouhssini, K. Afdel, E. Agherrabi, M. Akouhar, and A. Abarda, "DeepDefend: A comprehensive framework for DDoS attack detection and prevention in cloud computing." Journal of King Saud University-Computer and Information Sciences, 36(2), p.101938, 2024.

[2] M.K. Karthick, G. Kiruthiga, P.M. Saraswathi, B. Dhiyanesh, and R. Radha, "A subset scaling recursive feature collection based DDoS detection using behavioural based ideal neural network for security in a cloud environment." Procedia Computer Science, 215, pp.509-518, 2022.

[3] M. Ouhssini, K. Afdel, E Agherrabi, M. Akouhar, and A. Abarda, DeepDefend: A comprehensive framework for DDoS attack detection and prevention in cloud computing. Journal of King Saud University-Computer and Information Sciences, 36(2), p.101938, 2024.

[4] I.F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: Datasets and comparative study." Computer Networks, 188, p.107840, 2021.

[5] M.J. Pasha, K.P. Rao, A. Malla Reddy, and V. Bande, "LRDADF: An AI enabled framework for detecting low-rate DDoS attacks in cloud computing environments. Measurement:" Sensors, 28, p.100828, 2023.

[6] A.B.Z. Mahmoodi, S. Sheikhi, E. Peltonen, and P. Kostakos, "Autonomous federated learning for distributed intrusion detection systems in public networks." IEEE Access, 11, pp.121325-121339, 2023.

[7] M. Lyu, H.H. Gharakheili, and V. Sivaraman, "A survey on enterprise network security: Asset behavioral monitoring and distributed attack detection." IEEE Access. 2024.

[8] Z.M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. IEEE Communications Surveys & Tutorials, 19(4), pp.2432-2455, 2017.

[9] O. Jullian, B. Otero, E. Rodriguez, N. Gutierrez, H. Antona, and R. Canal, "Deep-learning based detection for cyber-attacks in IoT networks: A distributed attack detection framework." Journal of Network and Systems Management, 31(2), p.33, 2023.

[10] F. Talpur, I.A. Korejo, A.A. Chandio, A. Ghulam, and M.S.H. Talpur, "ML-Based Detection of DDoS Attacks Using Evolutionary Algorithms Optimization." Sensors, 24(5), p.1672, 2024.

[11] The BOT-IoT dataset, "https://research.unsw.edu.au/projects/bot-iot-dataset", accessed on February 2025.

[12] The UNSW dataset, "https://research.unsw.edu.au/projects/unsw-nb15-dataset", accessed on February 2025.

[13] I.D. Oktaviani, and P A.G. utrada, "Knn imputation to missing values of regression-based rain duration prediction on bmkg data." Jurnal Infotel, 14(4), pp.249-254, 2022.

[14] L. Vu, and Q.U. Nguyen, "Handling imbalanced data in intrusion detection systems using generative adversarial networks." Journal of Research and Development on Information and Communication Technology, 2020(1), pp.1-13, 2020.

[15] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks." Towards Data Sci, 6(12), pp.310-316, 2017.

[16] J. Zhang, X. Zhang, Z. Liu, F. Fu, Y. Jiao, and F. Xu, "A Network Intrusion Detection Model Based on BiLSTM with Multi-Head Attention Mechanism." Electronics, 12(19), p.4170, 2023.

[17] M.D. Li, H. Zhao, X.W. Weng, and T. Han, "A novel nature-inspired algorithm for optimization: Virus colony search." Advances in engineering software, 92, pp.65-88, 2016.

[18] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system." IEEE Access, 7, pp.41525-41550, 2019.

[19] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, and C. Wang, "Machine learning and deep learning methods for cybersecurity." IEEE Access, 6, pp.35365-35381, 2018.

[20] S. Sreekutty, A. Santhosh, and R. Achuthan, "A hybrid deep learning-based model for anomaly detection in cloud datacenters." Future Generation Computer Systems, 115, pp.350-360, 2021.

[21] Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud." Journal of Network and Computer Applications, 36(1), pp.42-57, 2013.

[22] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets." Computers & Security, 86, pp.147-167, 2019.

[23] Wei, Y. Li, Q. He, and M. Xu, “A DDoS attack detection method based on hybrid deep neural networks.” Security and Communication Networks, 2019, Article ID 7130862.
[24] Y. Wang, J. Zhang, D. Xu, J. Chen, and C. Xu, “An intrusion detection method based on BiLSTM-CNN for industrial control networks.” Future Internet, 14(5), p.145, 2022.
[25] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection.” IEEE Communications Surveys & Tutorials, 18(2), pp.1153-1176, 2016.
[26] Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning.” IEEE Communications Surveys & Tutorials, 10(4), pp.56-76, 2008.
[27] K. Scarfone and P. Mell, “Guide to intrusion detection and prevention systems (IDPS).” NIST Special Publication 800-94, pp.1-127, 2007.
[28] Vijay Kumar Gandam, E. Aravind, "Enhancing Cloud Security: A Novel Intrusion Detection System Using Deep Learning Algorithms." International Journal of Computer Applications, vol. 186, no. 44, pp. 36-42, Oct 2024. DOI:10.5120/ijca2024924070.