SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I10P118

Volume 12 Issue 10, 232-251, October 2025

Original Article

An Optimized Federated Learning Algorithm for Decentralized Intrusion Detection Systems Using Artificial Bee Colony Optimization

Shourya Shukla[1], Ajay Singh Raghuvanshi[1], Saikat Majumder[1]

[1]*Department of Electronics and Communication Engineering, National Institute of Technology, Raipur, Chhattisgarh, India.*

[1]*Corresponding Author : sshukla.phd2019.ece@nitrr.ac.in*

Received: 18 August 2025 Revised: 20 September 2025 Accepted: 19 October 2025 Published: 31 October 2025

Abstract - *With the advancement in IoT technologies, Wireless Sensor Networks have found many applications in the modern era. Due to this, the malicious activities in the networks have seen a major surge. Data theft and manipulation have been a serious concern among researchers. In real-time scenarios, the data communication from nodes to the servers increases the communication overhead and makes the network vulnerable to attacks. A decentralized detection strategy has become a necessity to detect these intrusions efficiently. Federated Learning algorithms have been a major choice for decentralized learning frameworks. The federated models learn the data patterns based on the trained local models. In this paper, a novel model aggregation strategy has been proposed. The weightage or local share of each client is optimized using the Artificial Bee Colony Optimization algorithm, called the optimized local share. The optimized local share has been utilized for three neural network architectures with five-layer deep structures. A Fully Connected Network, a Long Short-Term Memory network, and a hybrid network were employed to detect intrusions in the network. The NSL-KDD and UNSW-NB15 datasets have been distributed into 5, 10, and 20 clients for local training and aggregated using optimized local shares. Binary and multi-class classification achieved high accuracies, comparable to State-of-the-Art frameworks and centralized learning models, while ensuring data privacy and integrity of each client.*

Keywords - *Federated Learning, Intrusion Detection Systems, Artificial Bee Colony Optimization, Deep learning, Model Aggregation.*

1. Introduction

With the advancement in Internet of Things (IoT) technologies, automated Machine-to-Machine (M2M) communication has become a reality. IoT uses resource-constrained, low-weight wireless communication technologies to communicate between small embedded systems. IoT devices are small computing devices that communicate via networking protocols such as Bluetooth, Wireless Fidelity (WIFI), zigbee, etc., with each other without human intervention [1].

Smart healthcare equipment, automated drones, networked sensors, and smart wearable technologies are a few examples of prominent IoT-based smart technologies. These devices have the ability to analyze data and communicate with other IoT devices through ubiquitous interactions such as alerts, alarms, and other warning systems [2]. IoT devices working in small clusters form Wireless Sensor Networks (WSN), where each cluster has sensor devices, communication devices, and a processing unit. In the WSN, each sensor has the ability to sense different data.

The increase in IoT-based Machine-to-Machine communication has led to many security concerns. Data privacy and Data integrity are often compromised in such scenarios. An act in which data theft, bandwidth disruption, or resource manipulation is deliberately performed is known as an Intrusion.

The Intrusions are malicious activities that may be active or passive in nature. Active attacks include spontaneous data or resource depletion. Denial of Service attacks, such as ping of death, in which large ping data is sent to the victim machine, depleting its bandwidth. Other commonly employed active attacks include backdoor, fuzzer, exploits, remote to local, etc.

On the other hand, probe attacks are passive attacks that seek vulnerabilities in the network. These attacks remain dormant for most of the time, searching for vulnerabilities in the victim node. Once such a vulnerability is found, a passive attack sends information about the same to the intruder. The attacker then initiates an attack on the vulnerability and, in most cases, executes a successful intrusion.

Early detection of these Intrusions leads to safeguarding the network and the network resources. Intrusion Detection Systems (IDS) have been able to detect any malicious activity in the IoT or WSN. IDS shows zero trust toward the packet received by any node in the network. IDS verifies each packet based on the ETS signature or learns a pattern to detect unwanted activity [3]. These checks are not only performed for external communication, intra-network communications, such as sensed data packets in a WSN, but inter-cluster packet transfers within a network are also examined. These steps are performed not only to protect the network from external threats but also to detect corrupt internal nodes within the network. Signature-based IDS (SIDS) maintains a record of known attacks and has to be frequently updated to detect malicious activities. The signature-based IDS has a major drawback when it comes to zero-day attacks. Any novel attack generally remains undetected by the signature-based IDS. To overcome this vulnerability of the SIDS, anomaly-based IDS has gained popularity among many researchers in recent years. The anomaly-based IDS has the ability to learn the pattern of normal packets. Any severe variation from the normal behavior of the packet is marked as an anomaly. This feature of anomaly-based IDS enables it to detect a zero-day attack.

In the anomaly based IDS, as the number of features extracted increases, the complexity of the pattern exceeds human cognitive capability in detecting the intrusions. Artificial Intelligence, like Machine Learning (ML) and Deep Learning (DL) algorithms, learns to differentiate the normal and attack patterns successfully. The ML and DL algorithms require a large amount of data to train the classifiers efficiently.

However, recently, many countries and organizations have restricted the use of users' data to train and test for classification purposes due to the risk of data theft, data leakage, and, in some cases, even data duplication. In 2016, the General Data Protection Regulation (GDPR) was imposed by the European Union to safeguard the personal data of citizens all over the world. In this regulation, user consent was required to access their data [4]. This regulation considered IP addresses, unique identifiers, and access points as users' personal data. Similar regulations were passed by the California Consumer Privacy Act (CCPA) of the USA and the Personal Data Protection Act (PDPA) in Singapore. The Information Technology (IT) Act & Rules in India in the year 2023 published a Digital Personal Data Protection Act, which gave the Indian citizen the right to access, modify, or delete personal data from a database.

Moreover, these regulations have deprived researchers from collecting and training their intelligent systems for the early detection of intrusions. In the case of WSN, the problem of data islands has become a severe issue. Each node or access point has its own data, and data sharing among nodes, or clusters, or even with the base station has been restricted in many networks due to these regulations. One of the effective solutions to this problem is deploying and routing nodes as a distributed infrastructure. The decentralization property of the distributed infrastructure enables a network to maintain data privacy and integrity. As a result, intercommunication within a network creates vulnerabilities in the system. The centralized data processing can lead to major data breaches, and several other attempts can be made to disrupt the normal working of the IoT devices [5].

Federated Learning (FL) has emerged as a novel approach for training an expert system on distributed data. Federated Learning utilizes the local data for training, and the global model is trained based on the aggregation of local models. FedAvg [6] and FedProx [7] have been the most used FL algorithms. The FedAvg algorithm uses the average of the trained weights as an aggregation function for the global weight. Meanwhile, FedProx introduced a proximal term to the local loss function for optimal weight assignment. The FL algorithms have advanced over conventional centralized learning models when dealing with non-Independent and Identically Distributed (non-IID) data. FL models share the locally learnt model to form the global model, whereas in a centralized training model, the data acquired by each node is shared with the server for training. Hence, with FL, user data is preserved in real-time scenarios while the global FL model achieves comparable accuracies to the centralized deep learning architectures.

In this paper, an Optimized Federated learning model has been proposed. The proposed model is an extension of the averaging function used to train the global model through locally trained models. The weightage of each locally trained is optimized using a metaheuristic optimization algorithm. Artificial Bee Colony Optimization has been employed to tune the weightage or local share of the locally trained model on non-IID data. The network intrusion datasets NSL-KDD and UNSW-NB15 have been employed to train the local IDS on different numbers of clients. This paper compares local and global accuracies based on a fully connected network, long short-term memory, and a hybrid model employing fully connected and LSTM layers.

2. Literature Survey

A comprehensive survey of state-of-the-art research has been presented in the field of federated learning and optimized FL.

Idrissi et al. [8] proposed an FL-based NIDS. The authors proposed an anomaly-based detection scheme for various datasets. USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018 datasets were employed. Autoencoders, along with their variants, were employed along with two variations of federated learning for the detection of anomalous packets. FedAvg and FedProx were used to emulate the distributed

architecture of the network. However, different autoencoders were able to detect the anomalies better for different datasets. Hence, it was concluded that federated learning with different deep learning architectures can provide suitable results in various scenarios.

Friha et al. [9] suggested an FL-based IDS model for agricultural IoT. Locally trained FCN, CNN, and RNN were utilized for a modified Federated learning algorithm. The authors demonstrated an Agriculture 4.0-based IDS with data privacy and integrity. A mini-batch gradient was employed for the weight update mechanism in the distributed scenario, and the authors achieved 93.29% accuracy.

Oliviera et al. [10] proposed a distributed IDS while maintaining the balance between accuracy and robustness. For rule-based attacks, a membership function-based decision was employed. In addition to the membership, varying levels of Gaussian Noise were added to the attack instances. An inversion attack was performed to train a detection device. A batch size of 1000 with different noise level was trained for 10 rounds, and 10 rounds each round consisted of 10 epochs, achieving 96.2% accuracy.

Jin et al. [11] proposed a solution for catastrophic forgetting for older classes. The authors proposed a class balance gradient loss function, which was employed to update the learning rate of the deep learning architecture. UNSW-NB15 and CICIDS2018 datasets were used to train the CNN-GRU-based model for 10 client systems. 68.764% accuracy was achieved for UNSW-NB15, and 99.62% accuracy for the CICIDS2018 dataset was achieved.

Li et al proposed an IDS with a distributed configuration. Dynamic weighted aggregation was used for learning. CSE-CIC-IDS2018 was employed to detect some of the latest attacks. To discard some of the inaccurate models, a local model has to perform better than a threshold value, which was set to 0.75. The CNN architecture was used with a 512 batch size for 10 rounds. The dynamic weighted algorithm gave high accuracy with less communication overhead [12].

Thein et al. [13] proposed an IDS model against poisoning attacks. The authors also focused on the heterogeneity of the data under study. They proposed a logit adjustment loss function based on mini-batches to train the local models. Each local model was trained on a 3-layer CNN architecture consisting of 256, 128, and 64 nodes, respectively. A dropout layer was added to preserve the most relevant 80% of the features extracted by CNN layers. A temperature scaling function was introduced in the cross-entropy function to adjust the logit adjustment loss.

Attota et al in [14] proposed a multi-view based federated learning algorithm for the detection of malicious packets in the network. The extracted features were selected using the Grey Wolf Optimization (GWO) algorithm. Biflow View, Packet View, and Uniflow View were used to train the local models. The authors achieved 94.175% accuracy for the Random Forest Classifier.

Lazzarini et al. [15] proposed a shallow artificial neural network architecture for clients, and the FedAvg algorithm was employed for aggregation of the local model to form a global model. ToN_IoT and CICIDS2017 datasets were used to create the decentralized scenario. The shallow network architecture consisted of a 3-layer fully connected network, with a 0.01 learning rate and 5 rounds of training with 5 to 8 epochs per round. The proposed model achieved 97.59% accuracy.

Zhao et al. [16] proposed a semi-supervised FL model for IDS. The authors focus on three issues for federated learning based IDS: data reproduction, non-IID data, and communication overhead. The distillation method and the CNN architecture were introduced for classifier and discriminator networks. An eight-layer CNN was employed for feature extraction for five communication rounds with the Adam optimizer. The authors were able to achieve 87.40% accuracy on the semi-supervised IoT data.

Verma et al. [17] proposed an FCN, CNN, and LSTM-based hybrid neural network for training non-IID data. The authors employed encryption techniques to secure the model gradient over the communication channel. With the log loss function, the hybrid model was able to achieve 99.44% accuracy over the IIoT dataset with 15 client nodes used to train the hybrid 11-layer model on distributed data.

Authors in [18] proposed an automatic weight-optimized federated learning algorithm. In this model, the loss function was optimized by taking heterogeneity into consideration. The MNIST dataset was used for training. The dynamic model with a novel loss function provided high accuracy on MNIST data for a two-layer CNN structure.

Li et al in [19] proposed a weight-optimized federated learning algorithm for the MNIST dataset. The authors employed several bio-inspired optimization algorithms to optimize the weights of the local clients. A genetic algorithm was applied to randomly generated weights, and new offspring were used to test the accuracy. If the new accuracy is found to be better than the older model, the model is replaced. The model provided 68% testing accuracy.

Park et al. [20] proposed a Particle Swarm Optimization (PSO) based FL algorithm. PSO was employed to reduce the model's communication cost. In each iteration, the best-performing client shares the model parameters instead of the weight update from each client. The weight update steps were performed on a hybrid deep learning structure with four CNN and four DNN layers on the CIFAR and MNIST datasets.

A similar approach was employed by the author in [21], where PSO was employed to reduce the communication overhead and hasten the decision for lung legion due to COVID-19 infection. Instead of the weight update by the back propagation step in the neural networks, PSO was employed to optimize the speed of change in the weight matrix.

Xu et al in [22] proposed a learning rate optimizer for federated learning algorithms. Dynamic learning rates were adapted by local clients to overcome the effects of a fading channel. The authors tested the model on CIFAR and MNIST datasets. However, 92.51% accuracy was achieved for CIFAR with a dynamic learning rate. The algorithm proved to perform better than the FedAvg algorithm.

Other distributed systems include the use of various distributed algorithms and distributed datasets. Blockchain technology and cloud-based IoT services have been one among the most widely used distributed technologies along with federated learning. Kumar et al. [23] proposed a blockchain-based IDS for a distributed system architecture. The authors used fog computing to detect a DDoS attack. RF and XGBoost algorithms were employed in a fog IoT environment to detect malicious packets.

Gad et al in [24] employed the ToN-IoT dataset to learn the model based on IoT devices. An XGBoost classifier was used to compare the accuracies on the full set of features and reduced sets of features. The feature selection was performed using chi-square analysis and a correlation matrix. The reduced set was given to SMOTE to mitigate the class imbalance problem. The XGBoost classifier was able to achieve 98.3% accuracy on the dataset.

Samunnisa et al. [25] proposed a distributed cloud computing algorithm for IDS. The authors employed clustering and classification algorithms for detection purposes. Clustering algorithms such as K-means and GMM were employed for feature transformation, and Machine learning algorithms were used for classification based on the transformed features. Different thresholds were defined for the RF classifier; the 0.5 threshold gave 99.85% accuracy.

Segura et al. [26] focused on the Software Defined Networks (SDN). The author used the IEEE 805.15.4 protocol for the nodes. Online change point detectors were installed on the nodes to identify any malicious activity in the network. Rule-based or signature-based thresholds were proposed for the centralized and distributed detection model.

Sokkalingam et al. [27] proposed a hybrid optimization algorithm with Support Vector Machine (SVM) for the detection of intrusive packets. 10-fold cross-validation on the NSL-KDD dataset. Particle Swarm Optimization and Harris Hawk Optimization were employed for feature selection, and SVM was used for classification purposes. Eight features were selected for training, and the model was able to achieve 97% accuracy.

Based on the rationale survey, the following gaps were found in the existing state-of-the-art methods:

- In real-world scenarios, the data collected by the sensor is often sent to the central node. This results in an increased communication overhead in the network.
- In [8–10], authors employed a decentralized framework for detection purposes. However, the authors did not take class imbalance into consideration.

Based on the mentioned research gaps, the proposed research has the following highlights:

- For a decentralized approach, a federated learning framework has been proposed in the proposed model for a different number of clients.
- To incorporate the class imbalance problem, the Artificial Bee Colony optimization algorithm has been employed to evaluate the optimized local share for each client.
- Three different deep neural network architectures have been trained for the detection of intrusions on two standard datasets, NSL-KDD and UNSW-NB15. Both binary and multi-class global classifiers were trained based on optimized local share.

The paper is organized as follows: The first section comprises the Introduction to FL and IDS. Section II consists of the literature survey based on optimization algorithms and federated learning models. The methodology of the framework is provided in Section III. Section IV contains details about the experimental results and a discussion. Section V has the conclusion of the research along with future work.

3. Methodology

In this paper, an optimized federated learning model has been proposed. The weighted federated learning algorithm has been optimized using Artificial Bee Colony Optimization. The framework has been depicted in Figure 1. Three different neural network architectures have been trained on the local data, and a Global model has been trained using model aggregation. The detailed description of the methodology employed is as follows:

3.1. Federated Learning

An Intrusion Detection System is deployed in the network with either a centralized or a distributed infrastructure. In the centralized deployment of the IDS, the data sensed by each cluster or client is communicated to the base station. At the base station, the data is analyzed for any malicious packets. If the packet contains a virus, the packet gets discarded. In the case of a distributed deployment strategy, the IDS is equipped with multiple nodes over the network [28]. For the distributed learning framework, Federated Learning provides a

hierarchical approach between the distributed nodes and the central station. The decentralized training is performed by the client, and the base station acts as a server for the deployed IDS. The training and testing of the framework are divided between the server and client sides, where the training of the data is performed by the clients, and on the other hand, the model aggregation is performed by the server, where the accuracy of the framework is tested.

The data can be divided horizontally or vertically. In the horizontal distribution of data, the data instances are randomly distributed among the clients, whereas in the case of vertical distribution, the features are divided among the clients. Each client contributes to the training of a different set of features. In the proposed algorithm, the network intrusion datasets are distributed horizontally and equally among each client.

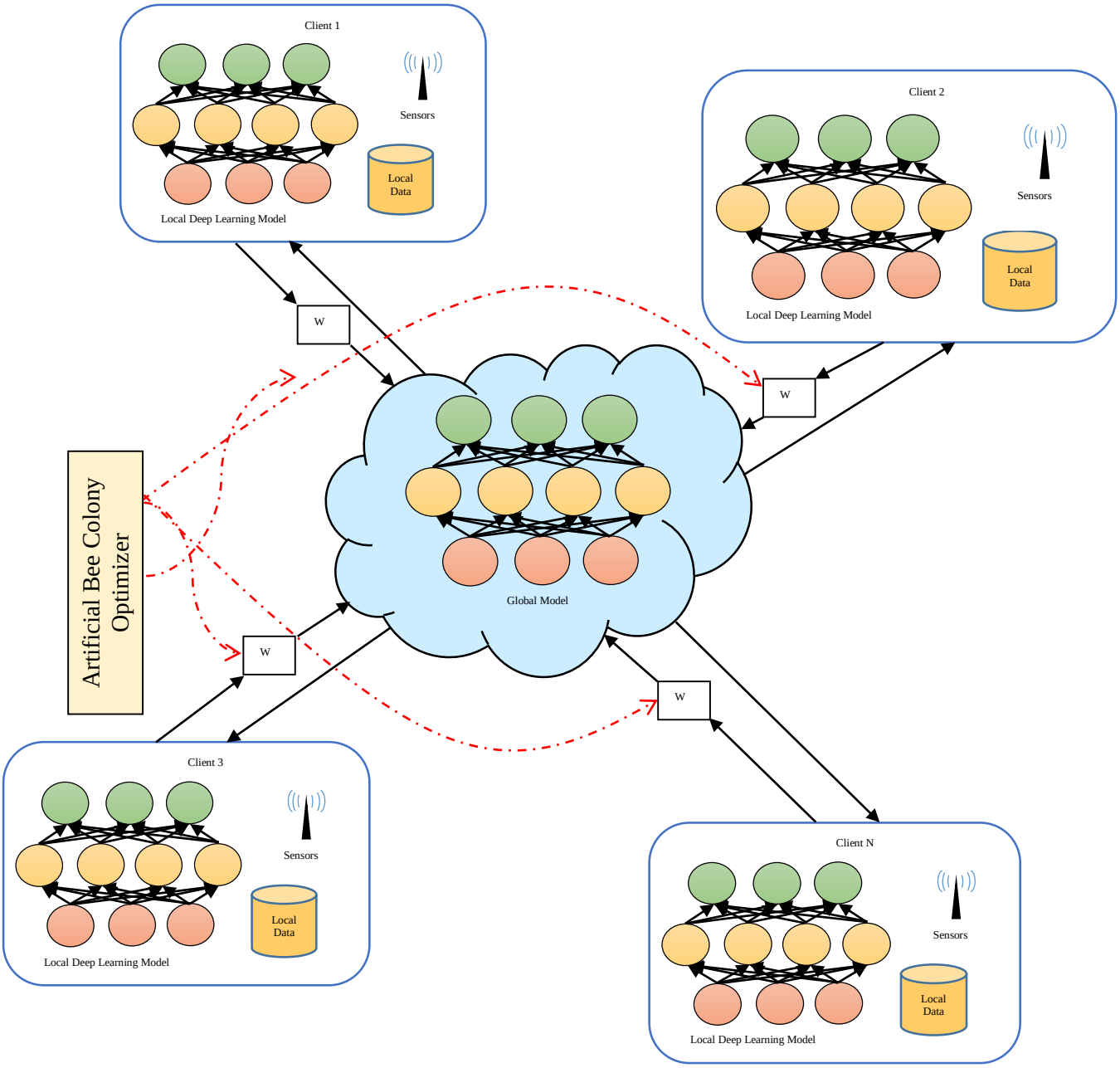

Fig. 1 Pictorial representation of the proposed framework

In Federated Learning, many variations have been invented in recent years. FedAvg is one of the most commonly used algorithms [6]. In this method, the weights of the neural network, trained by each client, are averaged, and hence, equal weightage is provided to each client. In the case of weighted FedAvg, the unequal data distribution is performed by random

allocation of the sensed data to a client. The percentage data acts as the weights of the client while model aggregation is performed. These models lack the ability to take data heterogeneity into consideration. FedProx and FedPSO [20] are also very readily used Federated Learning algorithms. The FedProx algorithm tweaks the loss function with a penalty term responsible for resisting any abrupt change in the client model from the global model.

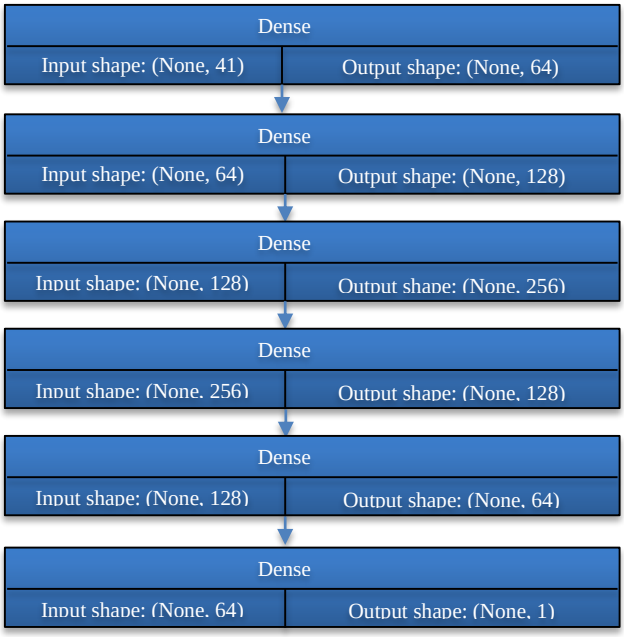

Fig. 2 Fully connected network layers

In the proposed framework, each client's weightage or local share is generated by an Artificial Bee Colony algorithm-based exploration strategy. The proposed framework consists of the following steps:

3.1.1. Global Model Initialization

In this step, the Neural Networks are generated by the server. Three different neural networks have been proposed for comparative training and testing. In this step, the weight initialization, layer activation, and learning rates are defined along with other hyperparameters. The three neural networks used for training and testing are:

For the first Neural Network, a five-layer hidden fully connected network is initialized with random weights and a learning rate. Binary classification is performed using binary cross-entropy, whereas multi-class classification is performed using sparse categorical cross-entropy.

The FCN is a deep learning architecture in which the output of each filter of the preceding layer is connected to every filter of the succeeding layer. This characteristic of the dense network enables it to establish linear as well as non-linear relations between input and output. The features extracted by the Fully Connected Layers are given as:

$$y = \phi(\zeta * x + \beta) \quad (1)$$

Where y is the extracted feature for the x input, the weights and bias are given as ζ, β, respectively.

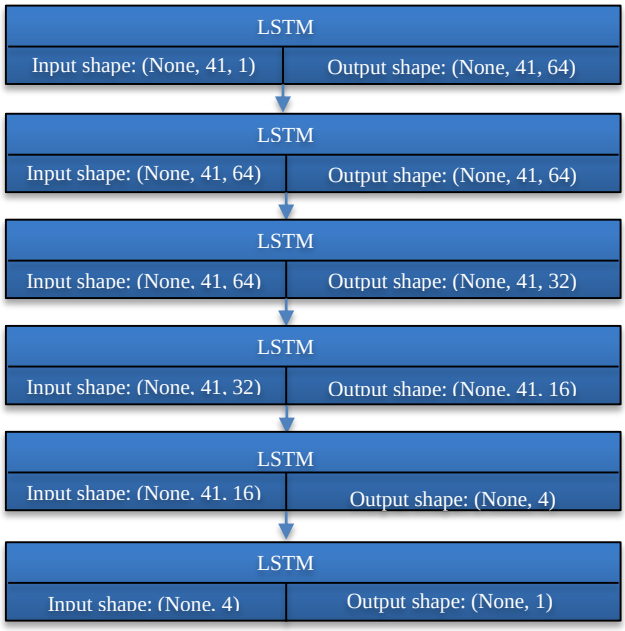

Fig. 3 LSTM network layers

This ϕ is the activation function used to introduce non-linearity in the feature extraction process. In the FCN, the ReLU activation is employed to eliminate negative features. The ReLU function is given as:

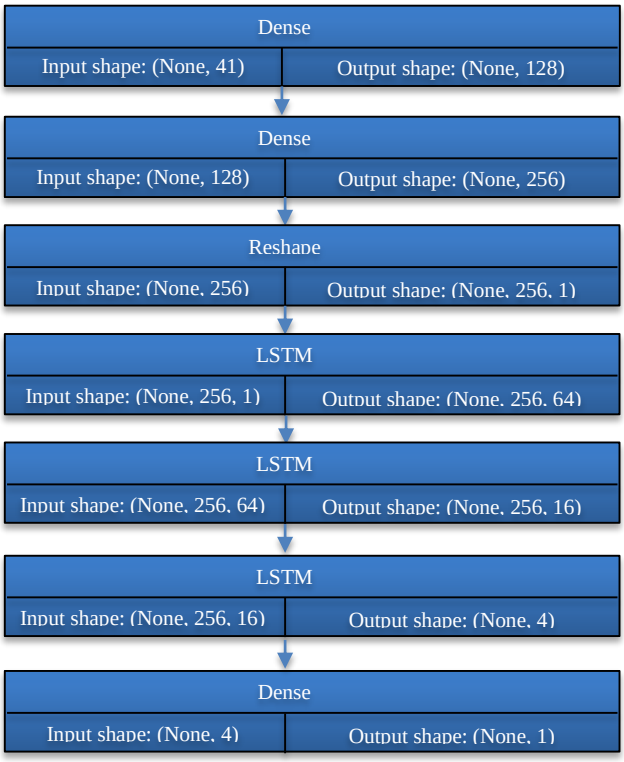

Fig. 4 Hybrid network layers

$$Re\,L\,U(\theta) = \phi(\theta) = max(\theta, 0) \quad (2)$$

A five-layer Fully Connected Network has been employed in the proposed algorithm for training each client, as shown in Figure 2. The input layer has 41 filters for the NSL-KDD dataset, whereas it has 43 filters for the UNSW-NB15 dataset. The input layer is followed by a 64 filter dense layer; more features were extracted with 128 and 256 filter layers. To mitigate overfitting, the important features were selected and passed to the output layer with 128 and 64 filter layers providing a bottleneck structure.

LSTM Network: The Long Short-Term Memory Network is a feedback-type neural network. The LSTM is a special case of a Recurrent Network that comprises the input gate, forget gate, and output gates. The feedback behaviour of the LSTM enables it to analyse sequential data or signal more efficiently. In the proposed model, the five-layer hidden-layer LSTM network assumes that the intrusion features are sequential in nature. The proposed architecture consists of an input layer with units equal to the number of features in the dataset. The input layer is followed by the hidden network. The hidden layer forms a converging bottleneck structure for preserving important features. The hidden layer consists of two 64-unit layers followed by a converging 32, 16, and 4-unit layers for feature extraction and selection purposes. The feature extraction of the LSTM layer relies on feedback for longer retention of the extracted information. The decision to store or eliminate an extracted feature is made by the forget gate. The feature extraction is given as:

$$\begin{aligned} fgt_\tau &= \sigma(\zeta_{fgt}.[h_{\tau-1}, x_\tau] + \beta_{fgt}) \\ inp_\tau &= \sigma(\zeta_{inp}.[h_{\tau-1}, x_\tau] + \beta_{inp}) \\ \hat{C}_\tau &= tanh(\zeta_C.[h_{\tau-1}, x_\tau] + \beta_C) \\ C_\tau &= fgt_t.C_{\tau-1} + inp_\tau.\hat{C}_\tau \\ out_\tau &= \sigma(\zeta_{out}.[h_{\tau-1}, x_\tau] + \beta_{out}) \\ &and \\ h_\tau &= out_\tau.tanh(C_\tau) \end{aligned} \quad (3)$$

Where fgt is the output of the forget gate, inp is the input gate, $\hat{C}_\tau$ is the candidate cell state forτtime, $C$is the cell state, and $out, h$are the output and hidden states, respectively. The activation functions are fixed for LSTM networks, with $\sigma$the sigmoid and hyperbolic tangent functions for candidate and hidden states given by:

$$tanh(\alpha) = \frac{e^{i\alpha} - e^{-i\alpha}}{e^{i\alpha} + e^{-i\alpha}} \quad (4)$$

Hybrid Neural Network (HNN): In the hybrid model, the HNN consists of 2-layer Fully Connected Layers, a reshape layer, and then followed by three LSTM layers, as shown in Figure 3. The reshape layer was added to make the features extracted by dense layers compatible with the sequential analysers of the LSTM layers. The structure consists of a dense layer of 128 filters, followed by another dense layer with 256 filters. The extracted features were made into a sequential tensor using a reshape layer. The LSTM part of the hybrid neural network consisted of 64, 16, and 4 units for the selection of relevant features. The dense layers were activated using ReLU activation, whereas the LSTM layers were activated by the hyperbolic tangent function.

3.1.2. Local Weight Updates

Each client model updates their model weights through back-propagation. The output of the neural network after each epoch calculates the loss present in the prediction of the malicious packet. Cross-entropy has been employed as the loss function for the proposed framework. The loss function is given as:

$$\ell(\gamma, \overline{\gamma}) = -[\gamma * log(\overline{\gamma}) + (1 - \gamma) * log(1 - \overline{\gamma})] \quad (5)$$

Where γ is the true class, and $\overline{\gamma}$is the predicted class by the local model? Then, the gradient of the loss function is evaluated, and the new weights are given as:

$$\zeta_{new} = \zeta - \eta * \frac{\partial \ell}{\partial \zeta} \quad (6)$$

Where η is the learning rate.

3.1.3. Global Aggregation

The global model is evaluated once all the client models are trained with their sensed data. This step is known as model aggregation. This step represents the most important feature of Federated Learning. Instead of communicating data, each client broadcasts their learnt model, and with the help of these models, a global model is constructed. In the proposed framework, instead of averaging or weightage being assigned according to the data share, the weightage allocation or the local share allocation is performed by exploring an optimal set of shares using a metaheuristic algorithm. In this paper, the Artificial Bee Colony algorithm has been employed for exploration purposes. The proposed global weight is formed by the sum of client models with their respective local shares. The global weight is given as:

$$\zeta_{Gbl} = \sum_\kappa w_\kappa.\zeta_\kappa = w_1\zeta_1 + w_2\zeta_2 + \ldots + w_N\zeta_N \quad (7)$$

Where $\zeta_{Gbl}$is the global model, $\zeta_\kappa$is the k[th] client's weights, and $w_\kappa$is the optimal local share of the k[th] client.

3.2. Artificial Bee Colony Optimization (ABCO)

In this paper, an FL-based neural network has been proposed for the detection of malicious packets in a decentralized network. The trained neural networks trained by each client are used to form the global model. The FedAvg [6], as the name suggests, forms the aggregate using the mean or weighted average of the model weights of each client. Weighted averaging of the models is performed on the percentage of data used to train the model, without taking Class imbalance and local model accuracies into account. Hence, these algorithms lack the qualitative aspect of the local models and focus on the quantitative aspect.

In this paper, the weightage of the local share of the clients is considered independent of the quantitative aspect of the data; the study focused on the optimal weights known as the local share of the client models. The paper proposes an ABCO-based optimum local share for each client.

ABCO was proposed by Karaboga in 2007 [29]. In the optimization algorithm, the optimal solution is inspired by a swarm of bees around the hive. The honeybees are a highly social species, so their swarm behaviour around the hive is studied in the ABCO. Swarm Intelligence is the study of the collective behaviour of a social colony or other animal societies to design an algorithm or problem-solving strategies. Self-organization and division of tasks among the bees inspired the ABCO algorithm. The solutions in the optimization problem are considered a food source or connected to the objective function of profitability.

According to the division of labour in the beehive, the bee population can be divided into three groups. The first group represents the Employed Bees (EB), followed by Onlooker Bees (OB) and Scout Bees (SB), collectively known as Unemployed Bees (UB). The population of the bees is initialized as:

$$w = low\_bnd + r * (upp\_bnd - low\_bnd) \quad (8)$$

Where N_P is the number of bees. $N_C$represents the number of clients employed to train the local models, forming a list of arrays. The dimension of the lower bound given by low_bnd, upper bounds given by upp_bnd , and random number sequence (r) is ($N_P \times N_C$). As the local shares are the weightage of the local client, they must sum up to 1; hence, each bee in the population is transformed to make fit the requirement by using the equation given below. Once the population of the bees is compensated, the Employed Bee Phase is executed.

$$w_{comp} = \frac{|w_i|}{\sum_{i=1}^{N_C} |w_i|} \quad (9)$$

In the ABCO, the fitness function of the solution and the objective function are related as given in the equation:

$$ftn = \begin{cases} \frac{1}{1+F_{obj}}, if F_{obj} \geq 0 \\ 1 + |F_{obj}|, if F_{obj} < 0 \end{cases} \quad (10)$$

3.2.1. Employed Bee Phase (EBP)

The bee population is then divided into EB and UB. The EB are the bees that are currently exploiting a food source or contain information such as distance and direction from the optimal solution. The EB can perform one of the three tasks: either dance around the food source and recruit new onlooker bees with a certain probability, or abandon the food source or continue to forage for the optimal solution. The OB watches the EB dance and interacts with them in hopes of becoming EB, whereas the SB spontaneously start searching around the hive. In the EBP, the number of food sources, the number of EB, and OB are set to Np/2, that is, half of the population forages for an optimal solution. In this phase, all the solutions get a chance to generate a new solution. The new solution in the employed bee phase is given by:

$$w_{new}^i = w_{xst}^i + \varphi * (w_{xst}^i - w_{ptn}^i) \quad (11)$$

Where phi φ is the random number between -1 and +1, Wp is the randomly selected partner bee, $w_{new}^i$and is the local share of the ith local client randomly selected to generate new solutions. As the new solution changes the total share, equation 3 is applied to the new solutions, which maintains their value as 1. After the new solutions are generated, they are used as a local share of the clients to generate global weights. The new global weights are tested on the data, and objective functions are evaluated for each solution or local share matrix. A greedy selection algorithm is applied to the new set of solutions. The greedy search algorithm is given by:

$$\begin{cases} w \leftarrow w_{new} \\ F_{obj} \leftarrow F_{obj,new} \end{cases}, if ftn_{new} > ftn \quad (12)$$

A trial counter is incremented each time the new solution is inferior to the existing one. If the new replaces the existing solution, the trial counter is set to 0.

$$trial_{new} = \begin{cases} trial + 1, if ftn \geq ftn_{new} \\ 0, otherwise \end{cases} \quad (13)$$

In the next step, the probabilities of modification of each solution are generated. The probabilities are based on the fitness function derived from the objective function. Solutions with higher fitness value have higher chances of participating in the onlooker bee phase. The probability using the equation:

$$prob_i = 0.9 * \frac{ftn_i}{max(ftn)} + 0.1 \quad (14)$$

After calculating the probabilities of the solutions, the Onlooker Bee Phase is implemented.

3.2.2. Onlooker Bee Phase (OBP)

In the OBP, each of the solutions gets a chance to generate a new solution based on a random number. Onlooker bees in the hive interact with worker bees, and the information about the solution is passed to the working bees with a random probability higher than the evaluated probability.

$$if prob_i > rnd$$
$$w_{new}^i = w_{xst}^i + \varphi * (w_{xst}^i - w_{ptn}^i) \quad (15)$$

Where r is the random number between 0 and 1, phi (Φ) is the random number between -1 and +1, and Wp is the randomly selected partner bee. w_{new}^i is the local share of the ith local client randomly selected to generate new solutions.

The roulette wheel method is employed for onlooker bees to generate new solutions. In the next step, a similar approach to the Employed bee phase is used. The new solutions are modified and then used to train the neural networks on the network intrusion datasets. The new fitness values are evaluated, and based on the greedy selection algorithm, the solution with higher fitness values is accepted. If the existing solution is found to be better, the trial counter is incremented; otherwise, it is reset to 0.

3.2.3. Scout Bee Phase (SBP)

In the SBP, the solution that has exceeded the abandonment criteria of the trial limit is replaced by a new solution. Only one solution enters this phase at a given round. The new solution is altogether generated using the initialization equation. For the new solution, the trial counter is reset to 0. In the proposed method, the abandonment criteria are set to 50. A higher value of the abandonment criteria is set to reduce the chances of the mitigation of a potential global optimum. The optimal set of local shares is obtained at the end of all iterations.

3.3. Objective Function

This paper has replicated data acquisition and detection of malicious activities in a real-world scenario of a decentralized IoT or WSN. The dataset has been distributed among various clients, and the models are trained locally. Each client has been trained to use a fully connected network, an LSTM network, and a hybrid network for the detection of intrusions. A novel model aggregation strategy has been proposed.

The weights learnt by the neural network during training are used to build the global model. The weightage parameters of each client, known as local share, are optimized by the ABCO. The ABCO initiates the local share randomly, and with each iteration, the local share matrix tries to approach optimal values based on the global accuracy. The objective of the optimization is to maximize the accuracy of the global model aggregated in the server by assigning an optimal local share to each client. The sum of weightage or the local share has to be unity, representing the qualitative approach to the FL model aggregation. The proposed objective function is given as:

$$objective = \max_{[w_1, w_2, \ldots w_n]} \frac{\#Packets(Class_{pred} = Class_{act})}{Total \rightleftarrows Packets} \quad (16)$$

Where $w_1, w_2, \ldots w_n$ is the local share of $Client_1, Client_2, \ldots Client_n$ the ratio of packets with correctly predicted classes to the total number of packets, which represents the accuracy of the neural networks.

4. Experimental Results and Discussion

4.1. Experimental Setup

The experiment was performed on a Dell Precision 5820 workstation. The workstation has a 32 GB DDR4 RAM with a 2 TB HDD. The Workstation is equipped with an Intel Xeon W-2133 CPU and a 4 GB Nvidia Quadro P2000 GPU. For the software, the proposed model was trained and tested using Python 3.8.20 with Numpy version 1.23.5 and Pandas version 2.2.3. The preprocessing steps were performed using the sklearn 2.2.0 library. Deep Learning libraries were used instead of federated learning libraries as they provide more leverage to modify the FL weight matrices. Tensorflow base library with version 2.3.0 was employed to train the local model and aggregate the global model.

4.2. Datasets

NSL-KDD Dataset: The NSL-KDD dataset is the most popular dataset for network-based Intrusions. The origin of this data dates back to 1998. The network intrusion simulated by DARPANET in 1998 was published as the DARPA dataset, which contained the packets received while multiple attacks were simulated on the Defense ARPANET systems. The experimentations were performed by MIT Lincoln Lab. Network traffic features were extracted from the raw packets of the DARPA dataset, and a new dataset was formed, which has more than 500,000 instances. These extracted features were named the KDD CUP99 or simply the KDD99 dataset. The KDD99 dataset has data duplication and redundancy issues. Tavallaee mitigated these issues [30]. The redundant data were removed from the KDD99 dataset, and a more manageable NSL-KDD dataset was proposed.

The dataset contains 125,973 packet instances along with 41 network features for each. The dataset contains instances belonging to 24 attack types and normal packet data for the benchmark. The 24 types of attacks belong to 4 classes, namely, DoS, which is a denial of service attack strategy. During a DoS attack, the network resources of a victim machine are depleted to the extent that complete failure of the node takes place. The next is the R2L attack, in which the unauthorized attacker tries to penetrate the network. The U2R is the user-to-root attack in which a user tries to gain superuser access. The last class is the Probe class, in which the passive attacks are executed to find vulnerabilities in the system. The dataset contains basic features, content features, time-based features, and host-based features.

UNSW-NB15 dataset: The dataset was recorded by sampling the network packets by the University of New South Wales (UNSW) in 2015 using the IXIA Perfect Storm toolbox. The dataset contains 9 attack types, and normal class instances were also provided [31]. The dataset consists of 43 features, which are categorized as Time features, basic features, flow features, and content features. The dataset has a collection of active attacks, such as Fuzzer, DoS, Backdoor, Generic, Shellcode, and worm; On the other hand, passive attacks such as Analysis have been simulated using 3 network structures. The classes are highly imbalanced, leading to problems of over-fitting and under-fitting during the training of an anomaly-based IDS.

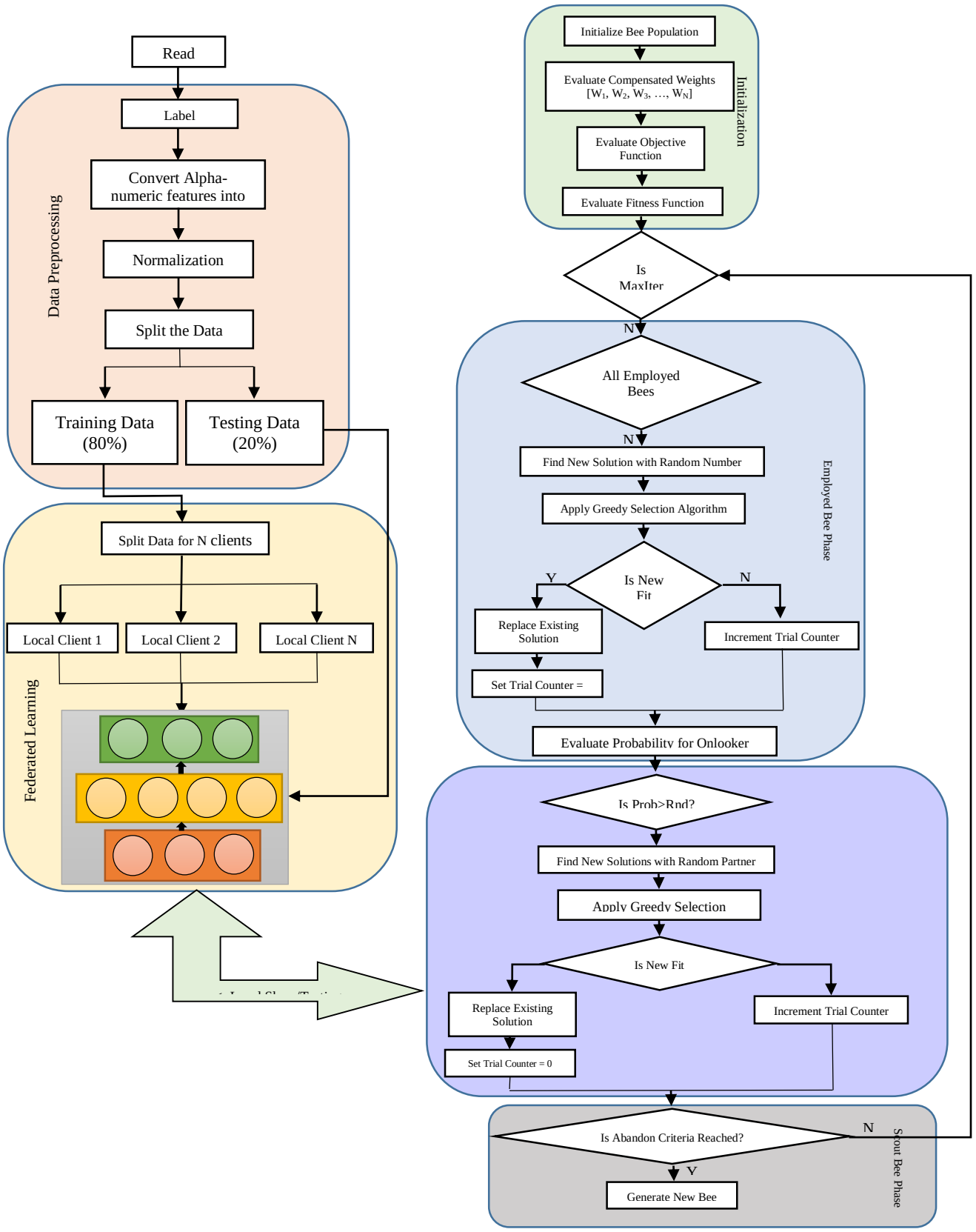

Fig. 5 Flowchart of the proposed algorithm

4.3. Experimental Steps

Figure 5 shows the flowchart of the proposed method. The network intrusion data is read and stored as dataframes using the Pandas library. The data undergoes preprocessing before classification and detection of malicious activities in the network. The preprocessing starts with the class label encoding. For binary classification, the NSL-KDD and UNSW-NB15 datasets, which have malicious packets, have been assigned +1 classes, and those instances that are normal classes are assigned 0 classes. However, in the case of multi-class classifications, the instance classes were assigned based on the type of attack executed by the packet. Since there are 24 attack categories in the NSL-KDD dataset, the 4 major attacks are assigned 4 labels: DoS, probe, R2L, and U2R, respectively. Moreover, it was observed that the dataset suffers from a severe class imbalance problem. Hence, these four attack classes, along with the normal class, create a multi-class dataset for classification. On the other hand, there are 10 classes in the UNSW-NB15 dataset. The 9 attack classes include analysis, backdoor, DoS, exploits, fuzzer, generic, reconnaissance, shellcode, and worm attacks, along with the normal class. High class imbalance was observed in the dataset. The next step in the preprocessing stage includes label encoding the data. The alphanumeric features are assigned numeric values before being fed to the tensorflow environment. The NSL-KDD dataset has three alphanumeric features, namely, protocol_type, service, and flag. The protocol_type contains information about the communication protocol used by the nodes in the network. TCP, UDP, and ICMP protocols were assigned labels using the label encoder. Service includes network services employed at the destination. These are http, ftp, telnet, or smtp. The flag feature contains the flag given to each packet, including 11 flags that were used to define the status of a packet. Similarly, in the UNSW-NB15 dataset, there are three features with alphanumeric values. The protocol used by the network is mentioned in the proto feature, and the service used in the destination is mentioned in the service feature. The connection status was also coded using alphanumeric keywords. These features are then normalized to confine their values between 0 and 1. This step makes sure that the features follow a similar random distribution of their values. The normalized values of the features are given by:

Table 1. Parameters and values used in the experiments

S. No.	Parameters	Values
1	Number of Clients	[5, 10, 20]
2	Types of Neural Networks	FCN, LSTM, Hybrid Neural Network
3	Batch Size	32
4	Epochs per Round	10
5	Federated Rounds	10
6	Learning Rate	0.01
7	Layer Activation	FCN: ReLU
		LSTM: Tanh
8	Optimizer	Adam
9	Loss	Binary Classifier: Binary Cross-Entropy
		Multiclass Classifier: Sparse Categorical Cross-entropy
10	Metric	Accuracy

$$\rho_N = \frac{\rho_i - \rho_{Min}}{\rho_{Max} - \rho_{Min}} \quad (17)$$

Where ρ_i is the i^{th}instance of the feature, with the extrema values of the feature, ρ_{Max}, ρ_{Min}respectively. Training and testing were split in an 80% to 20% ratio with a random shuffle.

The data and classes are divided equally among N clients in the federated learning phase. The proposed work used three neural network architectures to train the data locally. A five-layer Fully Connected Network, a five-layer LSTM network, and a hybrid network with two Fully Connected Layers followed by three LSTM layers were concatenated. The Fully Connected Layers were activated using the ReLU activation function, and the LSTM layers used the hyperbolic tangent function as their activation function. For the input layers, the number of filters or units in the neural network was set to the number of features in the dataset. In the UNSW-NB15 dataset, the ID feature was removed as it contains the packet ID and provides no information about the malicious activities. The output layers were initialized with a sigmoid activation function and 1 unit for the detection of the intrusive packet. In the case of multi-class classification, the output units were made equal to the number of classes, and a softmax activation function was used for probabilistic assignment of classes.

Parallel to the federated learning phase, the Artificial Bee Colony Optimization was initialized with a random population of bees. The bee determines the local share of each local client. The initial population is used to find the preliminary global accuracies of all the local clients. The local weights and their respective local share are used to form the global model as given in equation 7. These global models then make up the

employed and unemployed bee populations. The employed bees are then partnered with a random bee and form a new solution. Half of the total swarm size participates in this phase. The greedy algorithm is used to replace the solutions with better ones. This forms the EBP.

In the next step, the OBP is executed. The roulette interaction takes place based on their probabilities of interaction. New local shares are explored in this phase, and global accuracies are evaluated. It is not guaranteed that every employed bee participates in the exploration of new solutions. The new solutions are then compared to these existing solutions using the greedy algorithm. If a solution exceeds the abandonment criteria, the solution enters the scout phase. In this phase, new solutions are explored, and the solution with abandoning criteria is exploited. The global accuracies are evaluated using the testing dataset, and after the iterations are over, the best solution is achieved. The parameters used in the experiments are provided in Table 1. The optimal weights or optimal local share are used to aggregate the global model, and the results are presented in the next section.

4.4. Results

4.4.1. Binary Classification on NSL-KDD Dataset

The average results of binary classification on the NSL-KDD dataset have been illustrated in Table 2. In the experiment, the dataset has been horizontally divided into 5, 10, and 20 clients, respectively. The non-IID is divided randomly, and each client was provided with different numbers of normal and attack data. For the 5 client scenarios, the first client received 10,678 instances belonging to the normal class and 9,477 instances of the attack class.

On the other hand, the rest of the clients received 10,776, 10,944, 10,732, and 10,791 normal packets along with 9,379, 9,211, 9,423, and 9,364 packets of intrusion classes, respectively. The data was applied to a Fully Connected Network having a 5-layer architecture. The local network weights were initialized with a random normal initializer. A 0.01 learning rate with the Adam optimizer was employed for weight optimization. Due to the limited data received by each client, the initial training suffered from low accuracy.

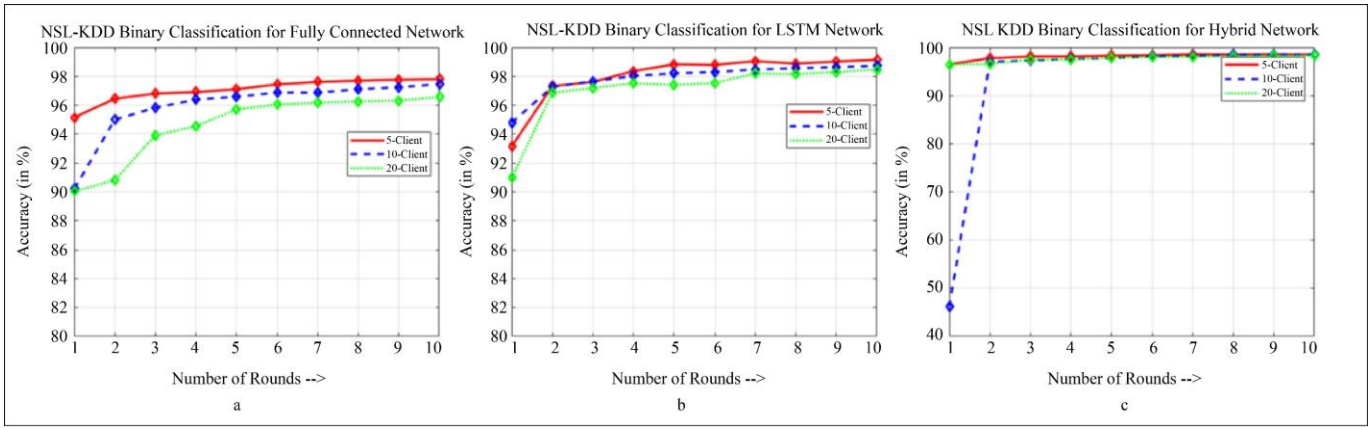

Fig. 6 Results for binary classification of NSL-KDD for (a) Fully connected network, (b) LSTM, and (c) Hybrid networks for 5, 10, and clients per round.

Table 1. Results for NSL-KDD binary classifications

Neural Network	Number of Clients	Average Normal Instances	Average Attack Instances	Average Initial Accuracy (Round 1)	Average Final Accuracy (Round 10)	Global Testing Accuracy
Fully Connected Network	5	10784.2	9370.8	88.02	97.49	97.81
	10	5394.7	4682.3	90.21	98.08	97.46
	20	2696.9	2341.1	90.82	96.77	96.57
LSTM Network	5	10755	9400	93.986	98.81	99.14
	10	5398.2	4678.8	92.677	98.36	98.73
	20	2698.75	2339.25	90.56	97.89	98.48
Hybrid Network	5	10758.4	9396.6	94.61	98.58	98.54
	10	5378.9	4698.1	84.48	97.44	98.56
	20	2692.58	2345.41	96.48	98.10	98.38

In the first round, the accuracy was 88.66% for the first client, 87.26% for the second client, 87.62% for the third client, 87.72% for the fourth client, and 88.86% for the fifth client. The global testing accuracy is 95.61% for the optimal local share. The global testing accuracy against the number of rounds has been depicted in Figure 6(a) for a 5-client scenario. With optimal local share provided to the training node, the accuracies are increased from 95.16% to 97.81% for the last round. This is the final result of training the global model through the aggregation of the local model with an optimal local share.

For 10 client systems, the data is divided into 10 equal sections, and each client received different ratios of normal and intrusion instances. The first client instances consist of 5384 and 4693 instances of normal and intrusion classes, respectively. The second client received 5412 normal and 4665 malicious instances, whereas the third client had 5451 and 4626 instances. Other clients had 5455, 5372, 5372, 5360, 5413, 5445, 5319, and 5371 normal instances and 4622, 4705, 4717, 4664, 4632, 4758, and 4706 intrusion instances, respectively. The average initial local accuracy of 90.21% was achieved. Furthermore, the accuracy after every round has been depicted in Figure 6(a). From Figure 6(a), it can be concluded that, as the rounds increase, the global model trains with the optimal local models learns efficiently without sharing the local data. This feature of the proposed algorithm enhances global accuracy without compromising the privacy of the local nodes. The final accuracy of 97.46% was achieved by the global model for 10 client scenarios on the FCN. The global accuracy of the 10-client scenario is comparable to the 5-client scenario, which shows that the data insufficiency issue is resolved in the proposed method.

Figure 6(a) shows the accuracy of 20 clients on a Fully Connected Network. The average normal and intrusive instances distributed among 20 clients were found to be 2696.9 and 2341.1, respectively. The initial round accuracies of all the local nodes averaged at 90.82% whereas with optimal federated learning, the final round training reached 96.77%. At the end of each round, the global accuracy went from 90.03% to 96.57%.

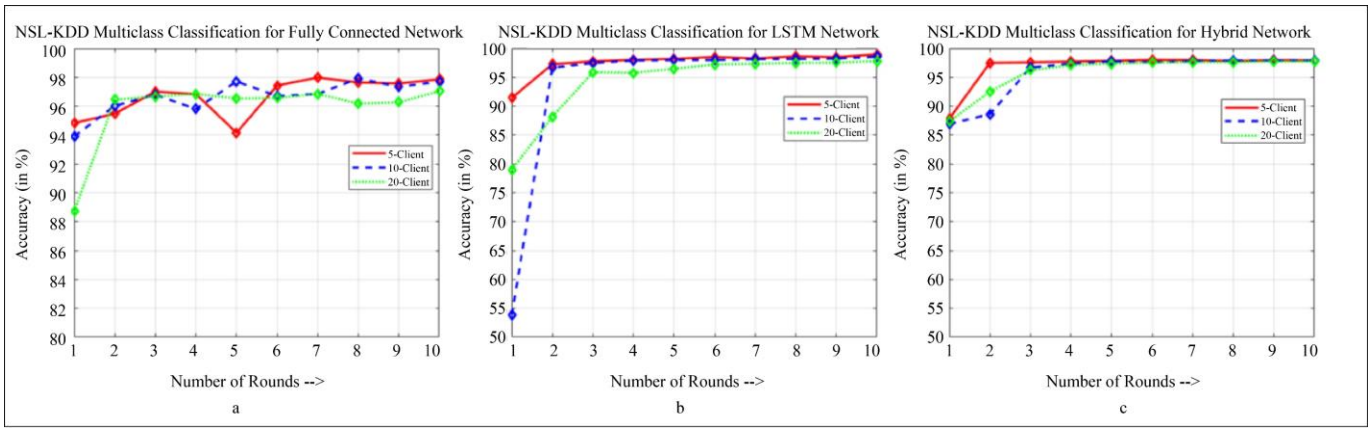

Fig. 7 Results for multiclass classification of NSL-KDD for (a) Fully connected network, (b) LSTM, and (c) Hybrid networks for 5, 10, and clients per round.

Table 3. Results for NSL-KDD multiclass classifications

Neural Network	Number of Clients	Avg. Normal Inst.	Avg. DoS Inst.	Avg. Probe Inst.	Avg. R2L Inst.	Avg. U2R Inst.	Avg. Initial Acc. (Round 1)	Avg. Final Acc. (Round 10)	Global Test Acc.
Fully Connected Network	5	9391.8	6597	1625	7.4	14.8	95.57	97.63	97.86
	10	4695.9	3298.5	812.5	3.7	7.4	94.54	97.53	97.70
	20	2347.95	1649.25	406.25	1.85	3.7	92.74	97.01	97.10
LSTM Network	5	10770.6	7506	1853.2	7.8	17.4	90.60	98.54	98.97
	10	5385	3752.8	926.6	3.9	8.7	67.02	97.98	98.56
	20	2692.05	1876.35	463.3	1.95	4.35	60.95	97.00	97.79
Hybrid Network	5	10792.8	7476.8	1860.2	8.4	16.8	86.02	97.77	97.92
	10	5396	3738.4	930	4.2	8.4	75.60	97.56	97.89
	20	2697.7	1869	465	2.1	4.2	76.82	97.23	97.74

Despite the fact that deep learning algorithms require a huge amount of training and testing data, an optimal federated learning model achieves a comparable accuracy with respect to a centralized learning algorithm with big data. For the LSTM network, the features of NSL-KDD data are considered as time series features. The data is divided into several clients. For a 5-client system, the average number of normal data and intrusion data was 10755 and 9400. The initial accuracy was found to be 93.986% and after 10 rounds of optimal federated learning aggregation of the global model, the training accuracy was increased to 98.81%. After the last round of aggregation, the final testing accuracy of 99.14% was achieved with the optimal local share of 0.183 for the first client, 0.209 for the second client, 0.356, 0.135, and 0.115 for the rest of the clients, respectively. Table 2 illustrates the class distribution and training and testing accuracies. The testing with respect to the rounds of training is depicted in Figure 6(b). For the 10 client scenario, the average normal and malicious instances were distributed as 5398.2 and 4678.8. Despite the limited data, the LSTM network's accuracy showed an increasing accuracy from 92.677% to 98.36% for the 10 rounds. The testing accuracy from 20% of the whole data came to 98.73% which is close to 5 client scenarios with half the data available for each node. In the case of the 20-client environment, the data available for each node is 4% of the total available data. Despite training on such limited data, the global model achieved 98.48% accuracy. The local models have an average initial accuracy of 90.56% and a final training accuracy of 97.89% in the last round. The LSTM showed better accuracy compared to the Fully Connected Network and the Hybrid Network for the binary classification of the KDD dataset.

The hybrid network consisted of 3 Fully Connected layers followed by 2 LSTM layers. The model was used to classify the malicious packets in the network. For 5 clients, each client had the average normal data with 10758.4 and malicious data with 9396.6 instances. The initial accuracy for all local clients averaged at 94.61% in the first round of training and improved to 98.58% in the last round. The testing accuracy also went up from 96.50% to 98.54%. Meanwhile, the federated accuracy for 10 clients was found to be 98.52%, which started with 46.23% only.

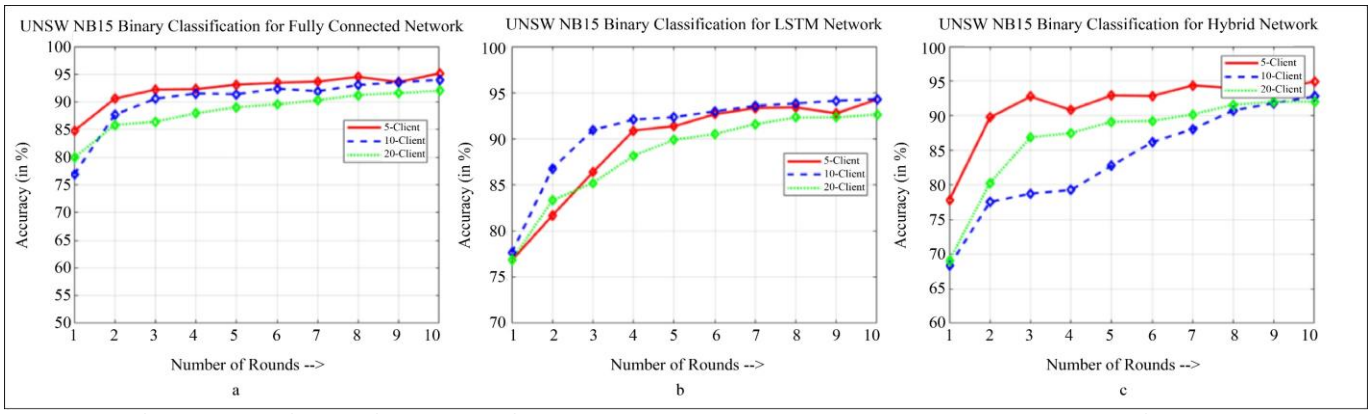

Fig. 8 Results for binary classification of UNSW NB15 for (a) Fully connected network, (b) LSTM, and (c) Hybrid networks for 5, 10, and clients per round.

Table 4. Results for UNSW NB15 binary classifications

Neural Network	Number of Clients	Average Normal Instances	Average Attack Instances	Average Initial Accuracy (Round 1)	Average Final Accuracy (Round 10)	Global Testing Accuracy
Fully Connected Network	5	5911.4	7261.6	84.66	93.984	95.61
	10	2955.4	3630.6	82.061	92.16	94.00
	20	1477.7	1815.3	79.0835	90.277	92.03
LSTM Network	5	5928.4	7244.6	70.672	92.894	94.26
	10	2959.7	3626.3	74.931	92.475	94.30
	20	1478.8	1814.2	72.0805	90.527	93.63
Hybrid Network	5	5919.6	7253.4	77.334	93.926	94.95
	10	2961.7	3624.3	69.809	90.596	92.67
	20	1479.75	1813.25	69.445	90.7035	92.04

The drastic change in accuracy is the result of the optimized federated approach employed for model aggregation. These accuracies were achieved on an average of 5378.9 normal and 4681.1 attack instances. Local models gave an average accuracy of 84.8% for the first round and 97.44% training accuracy in the last round. In the 20 client real-world problems, the average data distribution was 2692.58 for the normal class and 2345.41 for the attack class. The training accuracies in the first round averaged at 96.48% and improved to 98.10% by the last round. The accuracy has been shown in Table 2. Similarly, the testing accuracies were improved from 96.48% to 98.38%. Figure 6(c) depicts the increment in accuracy per round.

4.4.2. Multi-Class Classification for the NSL-KDD Dataset

In the multi-class problem, the task is not only to detect any intrusive packet, but also to classify the type of intrusion performed. The NSL-KDD dataset consists of 24 attack classes combined into 4 attack categories. The normal class is labelled as 0, whereas the attack classes were marked 1 to 4. The label encoder marked these classes in alphabetical order. DoS class was assigned as class 1, probe was given 2, R2L was provided with label 3, and U2R was given 4 as a label.

For the Fully Connected Network, the 5 client systems had an average of 9391.8 normal instances, 6597 DoS instances, 1625 probe instances, whereas only 7.4 and 14.8 average instances were obtained for R2L and U2R classes, respectively, depicting high class imbalance in the dataset. The initial average accuracy of the five local models was found to be 95.57% whereas the final accuracy was obtained at 97.36%. Table 3 shows the accuracies of the trained local models, the distribution of the classes, and the final testing accuracy. The testing accuracy for the first round was obtained to be 95.48% and it improved to 97.86% in the last round. The improvement in accuracy shows that the weight distribution performed by the ABCO algorithm has proven to be efficient in practical use. In the 10-client system, the class instances were halved for each client compared to the 5-client scenario. The normal class had an average of 4695.9 instances, DoS class had 3298.5 instances, probe class had 812.5, whereas R2L and U2R had 3.7 and 7.4 instances respectively. In the first round, the average training accuracy was 94.54%, whereas the testing accuracy of the aggregated model was 96.03%. After the optimized federated learning was executed for 10 rounds, the training accuracy averaged at 97.53% and the final testing accuracy was obtained as 97.70%. In the case of 20 clients, the average normal instances were 2347.95, DoS were 1649.25, probes were 406.25, and R2L and U2R were obtained at 1.85 and 3.7 only. The initial accuracy for training was found to be 92.74% whereas the initial testing accuracy was evaluated as 96.45%. After 10 rounds of optimal federated learning with ABCO determining the local share, the training accuracy was evaluated as 97.01% and the final testing accuracy was 97.10%.

For the LSTM network, the 5 client scenarios have 10770.6 average normal instances, 7506 DoS instances, 1853.2 probe instances, along with 7.8 and 17.4 instances of R2L and U2R, respectively. The initial average training accuracy was 90.60%, and after the global aggregation of the model with optimal local share, the testing accuracy in the first round was 97.26%. High testing accuracy determines that the model aggregation compensated for the losses in the local client model. At the end of the last round, the local models were able to achieve an average accuracy of 98.54% with a global testing accuracy of 98.97%. This accuracy shows that the optimal federated learning algorithm outperformed many state-of-the-art algorithms with even massive models. In the case of the 10 client systems, the results were not affected due to the lack of data individual local sites possessed. The normal class had an average of 5385 instances for all clients, DoS averaged at 3752.8 instances per client, probe averaged at 926.6 instances per client, R2L at 3.9, and U2R at 8.7 per client.

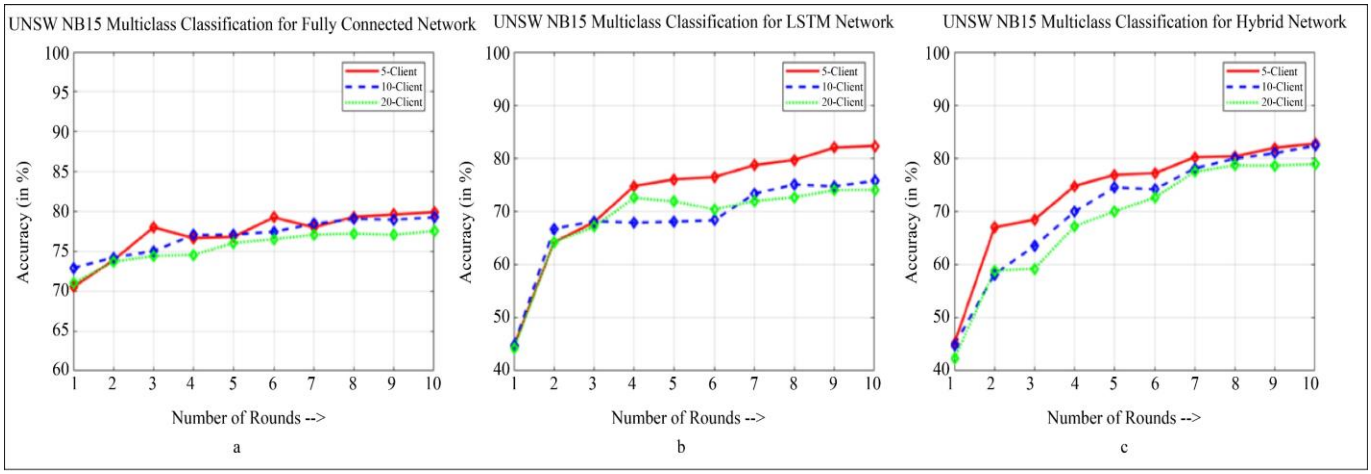

Fig. 9 Results for multiclass classification of UNSW NB15 for (a) Fully connected Network, (b) LSTM, and (c) Hybrid networks for 5, 10, and clients per round.

Table 5. Results for UNSW NB15 multiclass classifications

NN	No. of Cli	Avg. C1 Inst.	Avg. C2 Inst.	Avg. C3 Inst.	Avg. C4 Inst.	Avg. C5 Inst.	Avg. C6 Inst.	Avg. C7 Inst.	Avg. C8 Inst.	Avg. C9 Inst	Avg. C10 Inst	Avg. Initial Acc.	Avg. Final Acc.	Global Test Acc.
FCN	5	109.8	93.6	653	1772.4	964.6	3042	5909	559.4	61.6	7.6	71.73	78.71	79.91
	10	53.3	46.9	328.8	895.4	486.3	1504.4	2955	281.9	30.3	3.7	68.36	78.10	79.26
	20	26.65	23.45	164.4	447.7	243.15	752.2	1477.5	140.95	15.15	1.85	65.72	76.36	77.69
LSTM	5	109.2	99.6	658.6	1775.4	967.4	3024	5928.2	542.6	61.2	6.8	44.90	80.43	82.31
	10	53.9	46.2	328	884.6	487.1	1513.8	2960.7	277.6	30.9	3.2	46.48	74.36	75.51
	20	26.95	23.1	164	442.3	243.55	756.9	1480.3	138.8	15.45	1.6	44.23	74.1	75.05
Hybrid	5	109.8	92	657.4	1771	977.4	3028.6	5911.2	558.2	60.4	7	43.93	80.43	82.78
	10	53.7	47.2	327.2	885.1	481.5	1510.1	2967.1	280.6	30	3.5	44.35	81.44	82.35
	20	26.85	23.6	163.6	442.55	240.75	755.05	1483.55	140.3	15	1.75	42.38	77.57	78.89

Table 6. Comparison of the proposed framework with existing researches

S. NO.	Author	Technique	Accuracy
1.	VinayKumar [32]	Deep Learning	75.20
2.	Almeseidin [33]	J48	93.20
3.	Ingre [34]	DT	90.30
4.	Jin [35]	Rule Based	98
5.	Friha [9]	FL+CNN	93.29
6.	Jin [11]	FL+CNN-GRU	68.78
7.	**Our Method**	**FL+LSTM (KDD Binary)**	**99.14**
8.	**Our Method**	**FL+LSTM (KDD Multi)**	**98.97**
9.	**Our Method**	**FL+ FCN (NB15 Bin)**	**95.61**
10.	**Our Method**	**FL+ Hybrid (NB15 Multi)**	**82.78**

The round 1 average training accuracy was 87.02% and the aggregated global accuracy was 96.67%. In the final round, the global aggregate was able to achieve an accuracy of 98.56%, with an average training accuracy of 97.98%. In the 20-client scenario, the data were further distributed among 20 clients, with 2692.05, 1876.35, 463.3, 1.95, and 4.35 average instances for normal, DoS, probe, R2L, and U2R classes, respectively. In the initial round, an average of 80.95% training accuracy was observed, with a testing accuracy of 88.17%. As the local models were suffering from data insufficiency, the global aggregate showed slower convergence to the final value. In the last round of federated training, the average training accuracy was 97% with a global testing accuracy of 97.79%. The testing accuracies per round for different numbers of clients have been depicted in Figure 7(b).

For the Hybrid model, the 5 client scenario had an average instance of normal class as 10792.8, DoS class as 7476.8, probe class as 1860.2, R2L class as 8.4, and U2R class as 16.8. The first-round accuracies averaged at 76.82%, with local clients having no interactions among them. In the first interaction, as optimal models were shared, the testing accuracy was 87.84%. As the interaction took place, the accuracy went up to 97.23% for training and 97.92% for testing. The 10-client model showed similar results due to an optimized model aggregation strategy. The classes had an average of 5396, 3738.4, 930, 4.2, and 8.4 per client. The starting average training accuracy was 75.60% with an aggregated testing accuracy of 86.9%. With the training and model aggregation with optimal local share of each client, the final training accuracies were averaged at 97.56% and the global accuracy was found to be 97.89%. The 20-client scenario has an even lower number for each client. The normal class instances per client were evaluated as 2697.7, whereas DoS was 1869, probe instances were 465, and only 2.1 and 4.2 instances per client were R2L and U2R, respectively. The

initial accuracy of the local clients was averaged at 76.82% and the final accuracies after 10 rounds of optimized federated learning were improved to 97.23%. However, the testing accuracies, which were evaluated after each round of aggregation, were found to be improving from 87.37% to 97.74%. This is depicted in Figure 7(c), concluding that the data requirement can be minimized by sharing the model instead of the data.

4.4.3. UNSW-NB15 Binary Classification

Similar to the NSL-KDD dataset, for the UNSW-NB15 dataset, three neural networks with a fully connected network, a long short-term memory network, and a hybrid network with three fully connected layers and 2 LSTM layers were employed to train the local models. An aggregate using local share was formed, and the global model was tested with 20% of the data as testing data. Artificial Bee Colony optimization was used to calibrate the local share. The number of clients varied from 5 to 20.

For the Fully Connected Network, the 5 client scenario was trained with an average of 5911.4 normal instances per client, whereas an average of 7244.6 instances of attack per client were distributed randomly. The initial average training accuracy of 84.66% for 5 clients was evaluated. The global model was aggregated with an optimal local share, giving an accuracy of 90.59% in the first round. The accuracy improved to 93.98% for training in the last round, and the testing accuracy improved to 95.61% after 10 rounds of aggregation. Table 4 illustrates the distribution of data over clients and accuracy in the training phase. The testing accuracy after every round has been depicted in Figure 8. For 10 client systems, initial accuracy averaged at 82.06% with 2955.4 normal and 3630.6 attack instances per client. After all rounds of training, model aggregation, and global weight communication were performed, the average training accuracy was found to be 92.16%. The testing accuracies were initially evaluated as 87.69% and improved to 94% with the proposed framework. In the 20-client scenario, the average and attack instances per client were 1477.7 and 1815.3 for the binary classifications. In the first round, the average training accuracy was observed to be 79.08% and the aggregated testing accuracy was evaluated as 85.84%. As the proposed framework was executed, the average training accuracy improved to 90.27%, and the global model achieved an accuracy of 92.03%.

For the LSTM network, the five client models have a class distribution of 5928.4 to 7244.6 instances for normal to attack per client. The initial average accuracy was 70.67% which improved to 92.89% at the end of the last round of training, whereas the testing accuracy of the global model, without being explicitly trained on any data, came out to be 86.38% for the first round and was improved to 94.26%. Figure 8 shows the changes in accuracy with respect to the number of rounds in the training phase. In the 10 client system, the normal and attack instances per client averaged at 2959.7 and 3626.3 for normal and attack classes, respectively. The initial average accuracy in the first round, independent of other models, was found to be 69.80%. After the first aggregation, the global accuracy was evaluated as 90.95%. At the end of the last round, the average accuracy of the client was evaluated at approximately 90.6%, whereas the testing accuracy was evaluated at 94.30% for the global model. In the 20-client scenario, the average client accuracy was obtained as 69.44% with an average of 1479.75 normal and 1813.25 attack instances, respectively. The testing accuracy obtained for 20 client systems was 85.21% and improved to 92.63% by the tenth round. At the same time, the average training was 90.70% in the last round. LSTM with less data has proven to be more accurate compared to the Fully Connected Network, as the feedback networks create more attributes than the feed-forward network.

For the hybrid network, the 5 client scenario has an average of 5919.6 normal instances per client, whereas 7253.4 attack instances per client. The non-IID data was divided randomly among the clients, and federated learning with optimal local share obtained 77.33% accuracy in the initial phase, whereas a combined global system gave an accuracy of 77.82%. The model is run for 10 rounds, and the final average accuracy was obtained at 93.93% for training and 94.95% for testing. In the 10 client system, the average normal and attack instances were found to be 2961.7 and 3624.3 per client, respectively. The initial average training accuracy was evaluated as 69.8% and the first aggregate accuracy was obtained as 68.29%. After the maximum of 10 rounds of federated learning were completed, the average training accuracy improved to 90.6% and thus the tenth aggregate accuracy came out to be 92.78%.

For the 20-client scenario, the normal and attack classes are distributed at an average of 1479.75 and 1813.25 instances per client. The pre-federated learning accuracies averaged at 69.45% and after the first round of model aggregation, the accuracy was obtained as 69.08%. As the rounds of training were executed, the average training accuracy improved to 90.7%. The final testing accuracy was found to be 92.04%, comparable to the centralized accuracies achieved by state-of-the-art algorithms. The Figure depicts the comparison of the 5-client, 10-client, and 20-client systems for the hybrid neural network.

4.4.4. UNSW-NB15 Multi-Class Classification

The UNSW-NB15 dataset is a network intrusion dataset with 10 classes. The table illustrates the class distribution, average accuracies for training, and the final testing accuracy of the proposed model. The classes are labeled and encoded by the sklearn library; hence, the classes are assigned a numeric value in alphabetical order. The Analysis class was assigned zero and has been denoted as the C1 class. Similarly, Backdoor, DoS, Exploit, Fuzzer, and Generic classes were

designated as C2, C3, C4, C5, and C6, respectively. Meanwhile, the normal class has been denoted as C7. Reconnaissance, Shellcode, and Worm attacks were given C8, C9, and C10 classes in the table.

For the Fully Connected Network, the 5 client systems had only 109.2 average analysis instances, 93.6 backdoor instances, 61.6 shellcode instances, and 7.6 worm instances. The classes with a larger share of packets were DoS with 653 instances, exploits with 1772.4 instances, fuzzer with 964.3, and 3042 instances of the generic class per client were distributed. The initial average accuracy was obtained at 71.73% and the global model accuracy of 70.54% was achieved by the optimal framework. The final accuracies are obtained at 78.71%, which is the average training accuracy, and the global accuracy of 79.91% was obtained after 10 rounds of model aggregation. For the 10 client scenario, the analysis class had a share of 53.3 average instances per client, the backdoor class had 46.9 average instances, and 328.8 for DoS, 895.4 instances were present for training the 10 client system. 486.3, 1504.4, 281.9, 30.3, and 3.7 for other attack classes, whereas 2955 instances per client were distributed for the normal class. The training accuracy before the model aggregation was averaged at 68.36% and after the first global aggregate, the testing accuracy was found to be 72.93%. At the end of the tenth round, the average training accuracy reached 78.1% whereas the final global testing accuracy was improved to 79.26%. For the 20 client scenario, the normal class instances per client were averaged at 1477.5, and the attack classes were averaged at 26.65, 23.45, 164.4, 447.7, 243,15, 752.2, 140.95, 15.15, and 1.85 for analysis, backdoor, DoS, exploits, fuzzer, generic, reconnaissance, shellcode, and worm attacks, respectively. The initial training accuracy was found to be 65.72% and a testing accuracy of 70.99% was achieved after the first aggregation. The accuracy improved to 76.36% for average training accuracy and to 77.68% for testing accuracy. Table 5 shows the accuracy and the distribution of instances. Figure 9 depicts the testing accuracies per round for the Fully Connected Network with the UNSW-NB15 dataset divided into different numbers of clients.

For the LSTM neural network, the 5 client simulations had a distribution of 5928.2 average normal instances, whereas the attack class had 658.6 for DoS, 1775.4 for exploits, 967.4 for fuzzer, and 3024 for generic classes. The classes with high imbalance suffered with even fewer instances, such as analysis, which had 109.2 instances, backdoor, which had 99.6, 61.2 for shellcode, and only 6.8 for the worm class. The initial average accuracy was found to be 44.90% and was improved to 80.43% in the last round. The testing accuracy of the global aggregate model was obtained at 44.69% in the first round and improved to 82.33% by the last round. For the 10 client systems, the normal class instances were 2960.7 per client, and the attacks per client are illustrated in Table 5. In the first round of training, the average of the training accuracies was 46.48% and at the end of all training rounds, the average accuracy increased to 74.36%. The aggregate model gave an accuracy of 44.89% and was improved to 75.51% after the tenth round. For the 20-client simulation, the initial accuracy for training was 44.23% and the testing accuracy was 44.23% as well. The federated learning algorithm with the optimal local share provided by the ABCO, the average training accuracy reached 74.1% and the testing accuracy reached 75.05% as depicted in Figure 9.

For the Hybrid Neural Network, the normal class instances were found to be 5911.2, 2967.1, and 1483.55 for 5, 10, and 20 clients, respectively. The generic class had 3028.6, 1510.1, and 755.05 for different client systems. The shellcode and worm class has the least number of data in the UNSW-NB15 dataset, with only 60.4, 30, and 15 instances per client for shellcode and 7, 3.5, and 1.75 for worm class. For the five client systems, the initial average accuracy was evaluated as 43.93% whereas the testing accuracy was found to be 45.21%. These accuracies were improved to 80.43% for training and 82.78% for testing the global model. In the case of 10 clients, the average training accuracy was observed to be comparable with the 5-client system, with 44.35% as training and 44.89% as testing accuracy initially, and was improved to 81.44% and 82.35% for training and testing, respectively. In the 20-client scenario, the training accuracy improved from 42.38% to 77.57% and the testing accuracy increased from 42.38% to 78.89%. For the multi-class classification of the UNSW-NB15 dataset, the hybrid neural network outperformed the Fully Connected and LSTM networks.

Table 6 compares state-of-the-art models published in different research studies. The proposed model has outperformed many algorithms with centralized and distributed training algorithms.

5. Conclusion

In this paper, a real-time scenario of the Wireless Sensor Network with a distributed architecture has been simulated. The simulated architecture was employed to train an Intrusion Detection System for decentralized data that was divided among various clients. The client-server relation has been optimized by exploring different local shares for each client in the network. Three different neural network architectures have been trained on NSL-KDD and UNSW-NB15 datasets for binary and multi-class classification. The training data was horizontally distributed among 5, 10, and 20 clients. Each client trained a Fully Connected Network, an LSTM network, and a hybrid Neural Network with 2 Fully Connected layers followed by 3 LSTM layers. The local shares were optimized using the Artificial Bee Colony Optimization algorithm with 10 population size. The Fully Connected Layer architecture consisted of 41 feature nodes followed by a 64-node hidden layer. The feature exploration was enhanced by 128 and 256-layer nodes. The important features were propagated through 128 and 64 nodes in the hidden layers. The client models

trained with limited data showed high aggregated accuracies. The LSTM architecture consisted of 64 units followed by another 64-unit layer. After these layers, the layers converged to the output with 32, 16, and 4 unit layers.

The LSTM model showed high accuracy and was able to achieve aggregate accuracy comparable to the state-of-the-art methodologies. On the other hand, the hybrid model outperformed the other models in multi-class classification with 128 and 256 nodes, a Fully Connected Layer followed by 3-layer LSTM networks with 64, 16, and 4 units. The proposed optimized algorithm outperformed many centralized trained and tested algorithms. The model sharing instead of data sharing does not adversely affect the system accuracy. In the future, many other metaheuristic optimization algorithms can be utilized to explore the optimum values of local share. Class imbalance issues can be addressed by adding synthetic data with generative algorithms such as SMOTE and GANs.

References

[1] Leonrdo Babun et al., "A Survey on IoT Platforms: Communication, Security, and Privacy Perspectives," *Computer Networks*, vol. 192, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Aejaz Nazir Lone, Suhel Mustajab, and Mahfooz Alam, "A Comprehensive Study on Cybersecurity Challenges and Opportunities in the IoT World," *Security and Privacy*, vol. 6, no. 6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Danish Javeed et al., "A Federated Learning-Based Zero Trust Intrusion Detection System for Internet of Things," *Ad Hoc Networks*, vol. 162, pp. 1-13, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] Evgenia Novikova, Elena Doynikova, and Sergey Golubev, "Federated Learning for Intrusion Detection in the Critical Infrastructures: Vertically Partitioned Data Use Case," *Algorithms*, vol. 15, no. 4, pp. 1-14, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Chen Zhang et al., "A Survey on Federated Learning," *Knowledge-Based Systems*, vol. 216, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Brendan McMahan et al., "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, PMLR, vol. 54, pp. 1273-1282, 2017. [Google Scholar] [Publisher Link]

[7] Tian Li et al., "Federated Optimization in Heterogeneous Networks," *Proceedings of Machine Learning and Systems 2*, 2018. [Google Scholar] [Publisher Link]

[8] Meryem Janati Idrissi et al., "Fed-ANIDS: Federated Learning for Anomaly-Based Network Intrusion Detection Systems," *Expert Systems with Applications*, vol. 234, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Othmane Friha et al., "FELIDS: Federated Learning-Based Intrusion Detection System for Agricultural Internet of Things," *Journal of Parallel and Distributed Computing*, vol. 165, pp. 17-31, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Jonathas A. de Oliveira et al., "F-NIDS — A Network Intrusion Detection System based on Federated Learning," *Computers Networks*, vol. 236, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Zhigang Jin et al., "FL-IIDS: A Novel Federated Learning-Based Incremental Intrusion Detection System," *Future Generation Computer Systems*, vol. 151, pp. 57-70, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Jianbin Li et al., "An Efficient Federated Learning System for Network Intrusion Detection," *IEEE Systems Journal*, vol. 17, no. 2, pp. 2455-2464, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Thin Tharaphe Thein, Yoshiaki Shiraishi, and Masakatu Morii, "Personalized Federated Learning-Based Intrusion Detection System: Poisoning Attack and Defense," *Future Generation Computer Systems*, vol. 153, pp. 182-192, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Dinesh Chowdary Attota et al., "An Ensemble Multi-View Federated Learning Intrusion Detection for IoT," *IEEE Access*, vol. 9, pp. 117734-117745, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Riccardo Lazzarini, Huaglory Tianfield, and Vassilis Charissis, "Federated Learning for IoT Intrusion Detection," *AI*, vol. 4, no. 3, pp. 509-530, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] Ruijie Zhao et al., "Semisupervised Federated-Learning-Based Intrusion Detection Method for Internet of Things," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8645-8657, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Priyanka Verma, John G. Breslin, and Donna O'Shea, "FLDID: Federated Learning Enabled Deep Intrusion Detection in Smart Manufacturing Industries," *Sensors*, vol. 22, no. 22, pp. 1-18, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] Xi Yu et al., "Federated Learning Optimization Algorithm for Automatic Weight Optimal," *Computational Intelligence and Neuroscience*, vol. 2022, no. 1, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Xinyan Li, Huimin Zhao, and Wu Deng, "IOFL: Intelligent-Optimization-Based Federated Learning for Non-IID Data," *IEEE Internet Things Journal*, vol. 11, no. 9, pp. 16693-16699, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Sunghwan Park, Yeryoung Suh, and Jaewoo Lee, "FedPSO: Federated Learning using Particle Swarm Optimization to Reduce Communication Costs," *Sensors*, vol. 21, no. 2, pp. 1-13, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Dasaradharami Reddy Kandati, and Thippa Reddy Gadekallu, "Federated Learning Approach for Early Detection of Chest Lesion Caused by COVID-19 Infection Using Particle Swarm Optimization," *Electronics*, vol. 12, no. 3, pp. 1-19, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[22] Chunmai Xu et al., "Learning Rate Optimization for Federated Learning Exploiting Over-the-Air Computation," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3742-3756, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[23] Randhir Kumar et al., "A Distributed Intrusion Detection System to Detect DDoS Attacks In Blockchain-Enabled IoT Network," *Journal of Parallel and Distributed Computing*, vol. 164, pp. 55-68, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Abdallah R. Gad et al., "A Distributed Intrusion Detection System using Machine Learning for IoT based on ToN-IoT Dataset," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 6, pp. 548-563, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] K. Samunnisa, G. Sunil Vijaya Kumar, and K. Madhavi, "Intrusion Detection System in Distributed Cloud Computing: Hybrid Clustering and Classification Methods," *Measurement: Sensors*, vol. 25, pp. 1-12, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[26] Gustavo A. Nunez Segura, Arsenia Chorti, and Cintia Borges Margi, "Centralized and Distributed Intrusion Detection for Resource-Constrained Wireless SDN Networks," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7746-7758, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[27] Sumathi Sokkalingam, and Rajesh Ramakrishnan, "An Intelligent Intrusion Detection System for Distributed Denial of Service Attacks: A Support Vector Machine with Hybrid Optimization Algorithm Based Approach," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 27, pp. 1-18, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[28] Shaashwat Agrawal et al., "Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions," *Computer Communications*, vol. 195, pp. 346-361, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[29] Dervis Karaboga, and Bahriye Basturk, "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm," *Journal of Global Optimization*, vol. 39, pp. 459-471, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[30] Mahbod Tavallaee et al., "A Detailed Analysis of the KDD CUP 99 Data Set," *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, pp. 1-6, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[31] Nour Moustafa, and Jill Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)," *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, ACT, Australia, pp. 1-6, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[32] R. Vinayakumar et al., "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525-41550, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[33] Mohammad Almseidin et al., "Evaluation of Machine Learning Algorithms for Intrusion Detection System," *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, Subotica, Serbia, pp. 277-282, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[34] Bhupendra Ingre, Anamika Yadav, and Atul Kumar Soni, "Decision Tree based Intrusion Detection System for NSL-KDD Dataset," *Information and Communication Technology for Intelligent Systems*, pp. 207-218, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[35] Dongzi Jin et al., "SwiftIDS: Real-Time Intrusion Detection System based on LightGBM and Parallel Intrusion Detection Mechanism," *Computers & Security*, vol. 97, 2020. [CrossRef] [Google Scholar] [Publisher Link]