Improving Intrusion Detection with Federated Learning for Enhanced Privacy Protection

SAHAAYA ARUL MARY S. A., SAMEER CHAUHAN, LUV SACHDEVA
School of Computer Science and Engineering
Vellore Institute of Technology, Vellore,
Vellore - 632014, Tamilnadu, INDIA

Abstract: There are legitimate privacy and security issues with the processing of massive amounts of sensitive data required to detect breaches, abnormalities, and security risks in network traffic (including IoT). Federated learning, a type of distributed machine learning, lets many people work together to train a singlemodel while keeping data privacy and independence. An alternative to training and assessing the modelon a central computer is a federated educational setting, whereby each client learn a local model havingthe same structure that is trained on its own dataset. After that, an aggregation server receives these local models and uses federated averaging to create an optimal global model. Designing efficient and effective solutions for intrusion detection systems (IDS) is greatly facilitated by this technique. We evaluated the efficacy of federated instruction for IDSs to that of conventional deep learning models in this study. Throughthe implementation of random client selection, our research shows that federated learning outperformed deep learning in terms of accuracy and loss, especially in data privacy and security-focused situations. Wedemonstrate via experiments how federated learning may build global models without exposing sensitive data, reducing the dangers of data leaks and breaches. The results show that federated average in federatedlearning could change the way IDS solutions are made, making them safer, more efficient, and more useful.

Key-words: security of communication networks, federated learning, anomaly detection, intrusion detection systems, and data privacy

Received: March 11, 2024. Revised: August 12, 2024. Accepted: September 11, 2024. Published: October 31, 2024.

1. Introduction

AS a potent instrument for identifying intrusions in computer networks, machine learning (ML) has gained prominence in recent times. Nevertheless, the efficacy of conventional machine learning methodologies is significantly impacted by the accessibility of extensive and varied datasets, which may prove to be arduous to procure in practical applications [1,2]. The aforementioned difficulty is compounded by the dynamic and dispersed characteristics of contemporary computer networks, which produce enormous volumes of data in an inconsistent and real-time fashion [3]. Furthermore, the centralized structure of classic machine learning algorithms poses substantial issues about the privacy and security of data, particularly in sensitive settings such as the healthcare industry, the financial sector, and the national security sector. The potential consequences of data breaches, leakage, and illegal utilization of sensitive information are such that they can erode confidence and hinder the implementation of machine learning strategies in these fields, consequently restricting their overall effectiveness and potential [4]. Federated learning has come up as a promising solution to tackle these challenges by facilitating distributed machine learning while ensuring the preservation of data privacy and security[24].

Federated learning is a viable method for creating effective intrusion detection systems. Federated learning makes it possible for several parties or customers to participate in the training of a shared model in a collaborative manner while maintaining the confidentiality and decentralization of their data [5,24]. In order to train the model, every client educates its model individually on its own information, and only changes to the model are shared with a central aggregators

or server. This is done rather than transmitting data to a centralized server for training purposes. The server collects the model changes from a number of different clients and then updates the global model. The updated global model is then given back to the clients for more iterations for further development. To improve the global model, this procedure is repeated in an iterative manner, but the raw data is not shared with any of the customers [6]. In the context of distributed computing or Internet of Things network computing, such a paradigm is thought to be advantageous. Traditional machine learning algorithms for intrusion detection systems (IDSs) have a number of benefits that federated learning does not. In the first place, it makes it possible to create models that are more precise and reliable by using the variety of data that is collected from a number of different customers. The second benefit is that it makes it possible to develop global models without jeopardizing the confidentiality and safetyof critical data information. Thirdly, it has the potential to save the energy and money needed to train massive machinelearning models by reducing the transmission costs. Fourth, it opens up IDS solutions to various customers with different data heterogeneity, improving their scalability and efficiency[6].

The purpose of our research was to assess the efficacy of federated learning techniques in creating IDSs using the widely used NSL-KDD dataset [7,8], which is a dataset for network hacking detection. We used the horizontal federated learning framework that aggregates client data and uses random sampling to accomplish this goal in each training cycle. Our federated learning model's performance was contrasted with that of a conventional deep learning model trainedon a central dataset. Based on our experimental findings,the federated learning method achieved better accuracy and loss than the conventional deep learning method. Because federated learning enabled us to create a strong IDS solution while protecting the confidentiality of the specific client data, this enhancement was especially apparent in situations when data security and privacy were paramount.

Our research shows that federated learning is a greatway to build intrusion detection systems (IDS) for situa- tions where data security and privacy are paramount. For both accuracy and loss, our trials showed that the federate learning model was superior than the classic deep learning model. When contrasted with the conventional deep learningmodel, the federate learning model outperformed it with a 98.067% accuracy rate and a reduced loss rate. Since the federate learning model outperforms the conventional deep learning model in terms of accuracy and loss rate, it may be concluded that it is superior at identifying network intrusions.As a result, intrusion detection systems may become more precise and trustworthy, which is great news for industries like healthcare, banking, and national security that deal with sensitive data and want to keep it safe from prying eyes. One way to make federated learning even more efficient and successful for intrusion detection systems is to employ a horizontal design with average aggregation and client selection at random. To avoid overfitting and make the model more generalizable, this design makes sure that all clients have access to different types of training data. The average aggregation approach is a great tool for dealing with local dataset variability and making sure that the global model accurately represents the client data. Furthermore, by selecting customers at random, we can guarantee that each client has an equal chance to take part in the training and provide feedback on the final model.

What follows is an outline of the rest of the paper. Following this, we will give you the rundown on federated learning and how it helps IDSs. As a follow-up, we will demonstrate relevant research. Next, we will go over the steps that were taken to gather data for this study. Last but not least, we outline the results and importance of this study.

2. Background

2.1 System for Federated Learning in Anomaly Intrusion Detection

Federated learning presents a multitude of benefits in the context of devising IDS (intrusion detection system) solutions that are both effective and efficient [9]. The subsequent advantages emphasize the potential of federated learning as a viable strategy for intrusion detection system (IDS) solutions:

Federated learning guarantees privacy by enabling the building of a global intrusion detection system (IDS) model without requiring the exchange of sensitive security records from particular branches or organizations. This is accomplished via the process of data sharing. The privacy and security of sensitive information is safeguarded via the use of federated learning, which involves training on dispersed data without transferring the actual data itself. In an intrusion detection system (IDS), where data on network traffic & security events might be very sensitive, this is of the utmost importance.

• Protection of Personal Rights: Federated learning guarantees privacy by enabling the building of a global intrusion detection system (IDS) model without requiring the exchange of sensitive security records from particular branches or organizations. This is accomplished via the process of data sharing. The privacy and security of sensitive information is safeguarded via the use of federated learning, which involves training on dispersed data without transferring the actual data itself. In an intrusion detection system (IDS), where data on network traffic & security events might be very sensitive, this is of the utmost importance.

• Increased Safety & Protection: Federated learning offers advantages in terms of data security since it stores the information on local devices & reduces the amount of data that is sent to a centralized server. In IDS applications, one of the most major concerns is the possibility of illegal access, interception, or theft of data while it is being sent. This strategy helps to limit the likelihood of these happening.

• Enhanced efficiency: Traditional server-based IDS programs may involve transmitting significant volumes of data to a centralized server for processing, which increases com-

munication expenses. Federated learning share just model updates, reducing data transfers and communication costs. With limited network capacity and large data quantities, this efficiency advantage is crucial.

• Ability to grow: Federated learning is very flexible, which means it can be used for IDS apps where data is spread out across many devices or places. In smart city wireless networks with many sensors that track traffic, for example, federated learning can create a global IDS model that looks at data from every sensor while keeping the data local. It makes it possible to analyze and find incidents of security across the whole network.

• Approach Based on Collaboration: Federated learning makes it possible for several companies to work together without jeopardizing the confidentiality and safety of their confidential information. With federated learning, several entities in an IDS may train their local models on their own data while still contributing to a common global model. This is especially useful when multiple departments or organizations need to collaborate. By embracing a wide variety of data sources and points of view, this partnership improves the overall effectiveness of the intrusion detection system (IDS).

• Updating in Real-Time: Federated learning allows for the simultaneous training of local models on all devices and their instantaneous transmission of training results to a central server. The intrusion detection system can thus quickly adjust to the evolving network environment and identify emerging threats as they happen.

• Better accuracy: Federated learning makes use of a bigger and more varied collection, which improves performance and accuracy. By giving the model different types of data from various sources to learn from, the IDS can find more types of threats, which makes it more accurate overall.

• Shorter Training Time: Federated learning decreases training time and resources compared to centralized methods. IDS improves efficiency and speeds model training by training on local devices.

• Cost reduction: By using networked devices that are already in place for model training, federated learning reduces costs. Organizations may use their existing infrastructure without requiring extra expenditures in hardware or software, leading to a cost-effective implementation and upkeep of intrusion detection systems (IDS).

Federated learning is an attractive strategy for creating cutting-edge, efficient, collaborative IDS solutions because it takes use of these benefits.

2.2 The Architecture of Federated Learning

The federated learning architecture comprises two discrete methodologies: vertical federated learning and horizontal federated learning. These approaches both facilitate the collaborative training of models while safeguarding data privacy.

Horizontal federated learning comprises numerous clients, such as various enterprises or network segments, who share the same set of features but use diverse examples of those characteristics. For instance, it's very uncommon for different companies to keep identical network traffic records, but from their own networks. Clients in this architecture work together to build an intrusion detection model, but they don't exchange raw traffic data directly. Customers usually do this by updating their models on a central server after training them locally on their own traffic data. The server combines these modifications and returns the modified model to the clients. When applied to several networks, horizontal federated learning improves intrusion detection accuracy without compromising data privacy.

However, in vertical federated learning, a shared collection of examples is used by several clients, each of which may have a unique set of characteristics. These clients may be network monitoring tools or sensors. For example, these clients may access many data sources, including system logs, sensor readings, and network traffic logs, but they all relate to the same system or network. Without sharing any raw data, the clients work together to build an intrusion detection model that uses a combination of characteristics. Clients train local models using their own data sources and then share changes with each other. By exploiting these enhancements, the clients work together to create an integrated model that combines various sources of information while maintaining each source's anonymity. By adding a wider variety of data while yet protecting the confidentiality of the data source, vertical federated learning improves the accuracy of intrusion detection.

2.3 Techniques for Federated Learning

Federated learning methods have become strong ways to train machine learning models together while keeping data private and preventing unauthorized access[24]. By using these methods, businesses can use the knowledge of many autonomous devices without putting private data at risk. Table 1 compares the four main federated learning methods: averaging, differential privacy, safe aggregation, and transfer learning.

Table 2 provides a summary and comparison of the various aggregation techniques that are used in federated learning.

2.4 Federated Learning Aggregation

Usually, the aggregation process takes place at a centralized server or aggregator, which is in charge of gathering client-provided local model changes and integrating them to create a global model. In FL, there are a several popular techniques for aggregation:

1. FedAvg, or Federated Averaging: FedAvg is a technique of aggregation that is used often in FL. The global model update in FedAvg is the average of client local model updates, usually gradients or model weights. In order to update the global model, the central server averages the local changes. FedAvg is easy to implement and has the potential to achieve high performance in a wide variety of real-world circumstances.

TABLE 1. Techniques for federated learning compared[24].

Technique	Description	Advantages	Disadvantages
Federated Averaging	Central server aggregates model updates from multiple clients and sends updated model back to clients.	Efficient, scales well to a large number of clients, and preserves the privacy of client data.	Slow convergence due to communication bottleneck; potential bias towards more frequently updated clients.
Federated Learning with Differential Privacy	Adds noise to model updates to protect client privacy.	Provides strong privacy guarantees; allows for more diverse client participation.	Introduces noise to model updates, which may reduce model accuracy.
Federated Learning with Secure Aggregation	Utilizes secure multiparty computation to aggregate model updates without revealing client data.	Provides strong privacy guarantees; preserves client data privacy, even in case of compromised server.	Computationally expensive; may require specialized hardware.
Federated Transfer Learning	Clients transfer knowledge learned from their localdata to a shared model.	Enables learning across domains and improves model generalization.	Requires similar data distributions across clients, which may lead to bias if clients have vastly different data.

TABLE 2. Shows aggregation approaches in federated learning.

Aggregation Method	Description	Use Cases
Federated Averaging	Average of the model parameters from all participating devices is taken as the updated model.	Image classification, speech recognition, natural language processing.
Federated Stochastic Gradient Descent (FSGD)	Aggregation of stochastic gradients from all participating devices to update the global model.	-
Federated Learning with Secure Aggregation (FSA)	Encrypted data and model parameters are transferred from participating devices to a central server, where the aggregation is performed with the help of secure multiparty computation (MPC).	Privacy-sensitive applications such as healthcare and finance.
Federated Distillation	Model compression technique where a smaller, more lightweight model is trained on the global model using knowledge distillation.	Healthcare, finance, edge computing.

2. Weighted Federated Averaging: An improvement on FedAvg, Weighted Federated Averaging gives various clients varying amounts of weight while aggregating their data. Weights may be determined by a variety of factors, including sample size or the efficiency of customers' offline models. When updating the local model, the central server takes the provided weights into consideration and computes the weighted average. Clients may be treated differently depending on their contributions or skills using weighted federated averaging [8].

FedAvgM; Federated Averaging using Momentum:An improvement on federated averaging, federated averaging with momentum (FedAvgM) updates the system by factoring in the current moment. By considering previous gradients, momentum speeds up the point of convergence of the optimisation process in classic gradient descent [10]. Similarly, FedAvgM uses the participating devices' prior gradients to speed up the federated optimization algorithm's convergence.

The updating equation for FedAvgM is as follows:

$$v_t = mu * v_{t-1} + lr * g_t, \quad (1)$$

$$w_t = w_{t-1} - v_t, \quad (2)$$

where v_t is the momentum vector at time t, mu is the momentum coefficient, lr is the learning rate, g_t is the average gradient of the participating devices at time t, and w_t is the updated model at time t.

FedAvgM has been shown to improve the convergence rate and final accuracy of the federated learning algorithm, especially in scenarios where the participating devices have a

heterogeneous data distribution. To provide a deeper understanding of FedAvgM, we defined the following variables:

- θ= Current global model parameters;
- θ_i = Current global model parameters;
- θ_{prev} = the previous global model parameters;
- g_i = local model update of device i;
- m = the momentum term;
- L_i = the local loss.

(a) Initialization:

- θ and m are initialized by the central server.

(b) Client Update:

- Each client device i updates its local model parameter θ_i by minimizing its local loss function $L_i(\theta_i)$.
- The local model update for the device i is given by

$$g_i = \theta_i - \theta. \quad (3)$$

(c) Server Aggregation:

- The central server aggregates the local model updates from all the devices to obtain the current aggregated local model update, which is denoted as G. The aggregation is performed by summing up all the local model updates across all client devices and dividing by the total number of client devices (N) to obtain the average:

$$G = (\sum g_i)/N. \quad (4)$$

(d) Momentum Update:

- The momentum term m is updated using the formula:

$$m \leftarrow \beta m + (1-\beta)(\theta - \theta_{\text{prev}}) + G \quad (5)$$

where, β is the momentum parameter. The formula combines the previous value of m, the difference between the current and previous global model parameters, and the aggregated local model update to calculate the new value of m.

1. Aggregation Security: When it comes to FL, one aggregation approach that prioritizes privacy and security is secure aggregation. Secure multiparty computations (SMPC) and homomorphic encryption are two examples of cryptographic methods that are often used to safeguard local model changes while they are being aggregated. While secure aggregation offers robust privacy protections, it may be more difficult and incur computational burden than alternative aggregation approaches.

2.5 Autoencoder

Often called "deep learning," deep neural networks (DNNs) represent a state-of-the-art branch of machine learning at the forefront of artificial intelligence (AI). They are built on multi-level representation learning techniques for modelling intricate interactions between data. Thus, traits and concepts at higher levels are defined in terms of those at lower levels. The back-propagation (BP) approach, so termed because it propagates the error in the neural network's estimate backward from the output layer towards the input layer, has historically been used to train neural networks. Along the process, we can modify the model parameters using BP. Regretfully, the BP algorithm has a number of flaws that prevented it from working effectively with DNNs. Autoencoders (AE) are type of artificial neural network that aims to copy their inputs to their outputs . They work by compressing the input into a latent-space representation also known as bottleneck, and then reconstructing the output from this representation.Autoencoder is an unsupervised machine learning algorithm. We can define autoencoder as feature extraction algorithm. The input data may be in the form of speech, text, image, or video. An Autoencoder finds a representation or code in order to perform useful transformations on the input data.The Decoder generates the output sequence by predicting the next output y_t given the hidden state h_t.

The hidden states h_i are computed using the formula:

$$h_t = f(W^{(hh)} h_{t-1}). \quad (6)$$

The output y_t at time step t is computed using the formula:

$$y_t = softmax(W^S h_t). \quad (7)$$

The output is calculated using the hidden state at the current time step together with the respective weight W(S). Softmax is used to create a probability vector that will help us determine the final output.

3. Related Literature

In the area of breach detection, federated learning has gotten a lot of interest as a way to protect privacy that lets various groups train models together without sharing private data. This part talks about the linked works on intruder detection using collaborative learning. This shows the efforts and progress scholars have made in this area.

FELIDS, an intrusion detection system based on federated learning and specifically developed to safeguard agricultural IoT infrastructures, was introduced by Friha et al. [18]. Aiming to protect data privacy, the system uses local learning, where devices share model changes with a central computer to make the recognition model better. Three types of deep learning models are used by the FELIDS system to make the farm IoT safer: neural networks that are deep, neural networks with convolution, and neural networks with recurrence. Three different sets of data were used to test the suggested intrusion detection system: CSE-CIC-IDS2018, MQTTset, as well as InSDN. The testing findings showed that when compared to standard centralized machine learning approaches, FELIDS was far better at identifying assaults and protecting the privacy of data from Internet of Things devices[24].

Attota et al. [19] suggested MV-FLID as a way to find intrusions in IoT networks using multiview federated learning. The writers talked about the problems with current breach detection methods and emphasized the need for smarter, more private methods. MV-FLID uses multiview learning and distributed learning to better find and classify attacks while protecting the privacy of data. The test results showed that MV-FLID was more accurate than other ways. The

writers said that MV-FLID protected data privacy through shared learning, but this paper didn't go into detail about how privacy was protected or talk about possible flaws. To make the suggested method for protecting personal IoT datamore credible, it would be helpful to talk more about privacy-preserving techniques, such as methods of encryption or differential privacy.

For Internet of Things (IoT) networks, Rahman et al. [20] presented a federated learning (FL)–based intrusion detection system (IDS). Using the NSL-KDD dataset, they did three different tests: centralized learning, device-based learning, and shared learning. Compared to the centralized model, the FL-based IDS performed quite similarly in their experimental assessment, reaching an accuracy of around 83.09%.

Nguyen et al. [21] suggested an IDS (intrusion detection system) for the IoT that uses federated learning, or FL, and a computerized method that is tailored to different types of devices. When tried on real devices that had the Mirai malware on them, this work had an amazing average success rate of 95.6 percent for finding threats, and it did so with an average response time about 257 ms. Also, the system didn't give off many fake alarms. It's important to keep in mind, though, that the suggested model was only made to find attacks on IoT devices. It wasn't thought about any other possible threats that could affect different parts ofthe environment, like advanced networking technologies like SDN and services like FTP and SSH.

4. Protecting Privacy in Intrusion Detection with Federated Learning

Finding Anomalies with Autoencoder Model: For the purpose of constructing an intrusion detection system, an autoencoder was used. One kind of neural network that intrusion detection systems may use for anomaly detection is an autoencoder. In this method, autoencoders are trained only on data that is representative of typical network traffic activities. Using an encoder network, the autoencoder learnt to compress the regular traffic patterns during training. The representation was then decoded by a decoder network, which resulted in the data being returned to its original state. Reconstruction error, defined as the discrepancy between input and reconstructed data, was the primary metric for the training aim.

An autoencoder that had been trained was applied to fresh instances of network traffic that had not been observed before during the phase of anomaly detection. In order to determine the degree of similarity in the observed traffic with the usual patterns that were learnt, the system was able to calculate the reconstruction error that occurred from the input and the output that was rebuilt. The instances that had larger reconstruction errors were categorized as anomalies, which indicated that there was a possibility of intrusions or assaults taking place. To discriminate between typical and aberrant cases, a threshold was chosen. Anomalies were identified as instances that had reconstruction errors that were higher than the threshold, whereas instances that had errors that were lower than the threshold were judged to be normal.

The autoencoder consists of an encoder and decoder network, where the encoder maps the input data X to a lower-dimensional representation Z through the encoding function h such that $Z = h(X)$,, and the decoder maps the lower-dimensional representation Z to the output data Ythrough the decoding function g such that $Y = g(Z)$. The autoencoder's job is to learn how to reduce the input data so that it keeps only the most important parts while reducing the difference between it and the recovered data as much as possible. This is also known as lowering the reconstruction error.

$$L(X, Y) = \|X - Y\|^2. \quad (8)$$

where X represents the data that was supplied, and Y the data that was rebuilt.

During the training phase, it is necessary to minimize the error in reconstruction L(X, Y) among the input data & the reconstructed data. This may be accomplished by adjusting the parameters of the decoder and encoder networks via the use of descent gradients or other optimization algorithms. After the autoencoder has been trained, it may be used for a variety of activities, including the encoding of input data in order to produce representations with lower dimensions, the generation of new data samples based on the representations that have been learnt, and the utilization of the encoder as a module for extraction of features for jobs that are further down the line. An anomaly detection method employed the trained autoencoder to rebuild fresh network traffic dataand compare the reconstruction error to a threshold. It was determined that the network traffic was considered to be an anomaly if the amount of reconstruction error was more than the threshold. This method excels at uncovering previously unseen anomalies since the autoencoder can pick up on any changes from the typical patterns it learnt during training. Figure 1 is a graphical depiction of the method that is used for the purpose of identifying anomalies using an autoencoder device.

The visualization of anomaly identification using autoencoders is shown in Figure 1.

Federated-Learning-Based Intrusion Detection System, or FELIDS:

Figure 2 shows one approach to construct federated averaging using federated machine learning's horizontal design. Here, N devices get the input, and they individually train their own model using the data that is physically available to them. Each client may train their own compressed data representation using a deep autoencoder as a local model. Following client-device-specific model training, the federated server receives model parameters and uses a federated averaging procedure to aggregate them with all client-device parameters. Maintaining the confidentiality of individual client data is made possible by this method, enabling the server to incorporate all clients' collective knowledge into the global model.

For the federated-learning-based system for intrusion detection (FELIDS) to start its federated-learning (FL) pro-

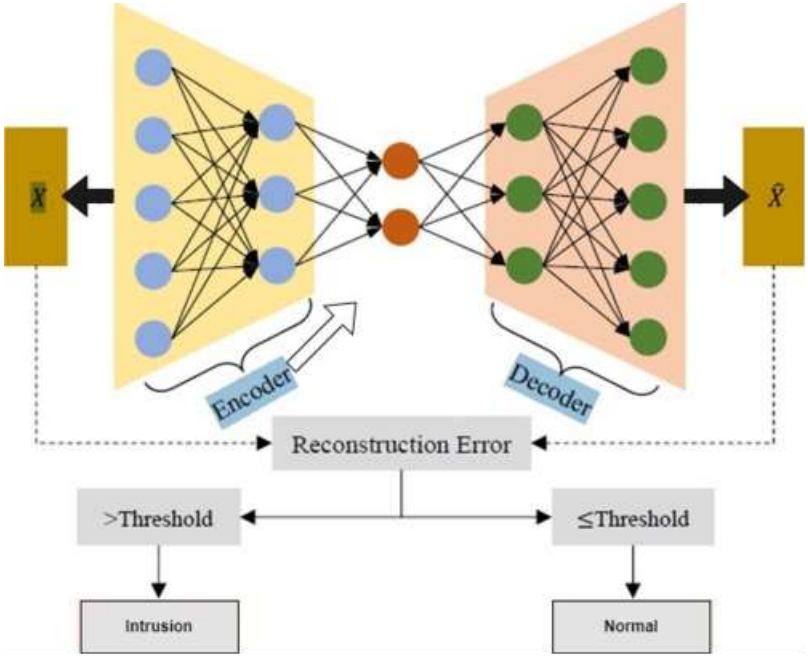

FIGURE 1. The visualization of anomaly identification using autoencoders is shown .

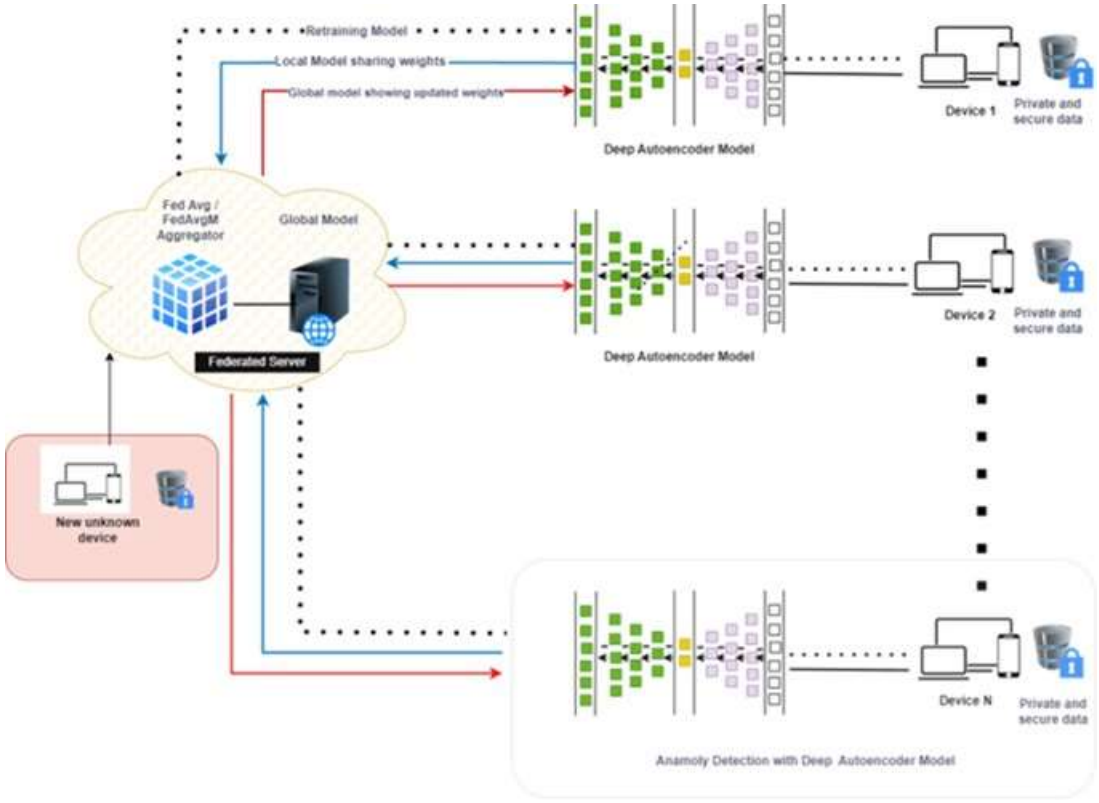

FIGURE 2. Intruder detection system with a horizontal design. .

cess, the FELIDS server chooses C out of K edge nodes (FELIDS clients) to take part in the calculation for R FL rounds. A strong intrusion detection model is to be trained cooperatively. Using a secure gRPC channel (Google remote

procedure call), clients and servers may communicate data securely. Both the client and the server can be authenticated and their communication can be encrypted since this channel has built-in support for SSL/TLS (transport layer security). During the FL process, all data transmitted is protected by this secure channel, ensuring its secrecy and integrity. Figure 2 shows the FL process following the steps and algorithms mentioned in Algorithm 1 and 2, both of which are based on the Fed Average algorithm [22]. This is done once all the chosen clients have connected over the secure gRPC channel.

Here is how the FL procedure operates:

1. The parameters of the global model are initialized by the server.

2. Using their own local data, each client trains a model individually, using the FL training approach.

3. Through the use of the secure gRPC channel, the clients submit their local model modifications to the server in a secure manner.

4. The federated averaging (FedAvg) technique is often used by the server in order to aggregate the local model updates that have been received.

5. The server implements the aggregate model updates in order to update the global model parameters.

6. Steps 2-5 are repeated for R FL rounds, enabling models to develop and converge into a superior intrusion detection model.

7. Following the FL procedure, the system-wide intrusion detection system uses the final global model.

After the clients have been linked, the algorithm willcarry out FedAvg, which is a process that includes pickinga subset of customers and delivering them the parameters ofthe current model. The model is then trained locally by each chosen client using the client's own private data, and after the model parameters have been updated, they are sent back to the server. Each client transmits their own set of parameters to the server, which then calculates a new global model and returns the results to the clients. In order to complete a certain number of communication cycles, this procedure is repeated.

When everything is said and done, the algorithm is terminated and the clients are released. By avoiding server-side data sharing and storing training data locally, this method aims to increase ML model accuracy without compromising user privacy.

Algorithm 1: StartServer Algorithm

```
Procedure STARTSERVER(K, C, R)
    while K ≠ length(ConnectedClients()) do
        // Loop until all clients areconnectedloop;
    end
    FedAvg();
    ReleaseClients();
Procedure FedAvg ()
    w_1 ← GenericModel();
    for t = 1, . . . , R do
        S_t ← Subset(max(C · K, 1), random);
        for k ∈ S_t do
            w_{t+1}^k ← Client_k · FedAvg(w_t)
        end
        w_{t+1} ← Σ_{k=1}^{n_t} (1/n_t) w_k;
    end
```

The "StartServer" is the component that is used to initiate the federated learning process, and Algorithm 1 provides a high-level definition of the concept. K, which represents the overall amount of FELIDS clients, C, which represents the proportion of clients selected to take part in each round, and R, which represents the number of rounds that the feder- ated learning process is required to complete, are the three parameters that are used by the algorithm. The algorithm's main loop continues until K clients connect to the server. The "FedAvg" algorithm, which executes R cycles of federated learning, is implemented within this iteration. During every round, a randomly chosen portion of C *K clients is selected to take part in the process. These clients then educate their local model by using the global model that is currently functioning. A fresh global model is then created by combining the models that have been changed at the local level. The customers are liberated from the procedure after all of the rounds have been finished. An explanation of the algorithm is provided in the next section:

•D: The entire dataset used for training the machine learning model.

•B: The number of clients or subsets into which the dataset is divided during training.

•E: The number of local training epochs for each client during each round of FedAvg.

•P: The preprocessed dataset obtained after applying the preprocess function to D.

• w: The model parameters (weights and biases) shared between clients and the server.

• B: The local dataset batch on each client obtained by splitting P into B subsets.

• η: The learning rate, which controls the step size of the model parameter updates.

• ∇ f (w, b): The gradient of the loss function f (w, b) with respect to w

Federated-learning-based intrusion detection systems employ FIELDS client algorithm 2 for federated averaging (FedAvg). When training the global model with their private

data, this procedure is done by every FELIDS client k. Initially, the client communicates with the FELIDS server in order to get the generic model. Local weights are generated by the client after parallelly training the generic model with their private data. Every user has a preprocessed database that is divided into localized mini batches of size B. The client then calculates the fresh weights by upgrading the previous ones utilizing the rate of learning, averaged gradient, and minibatch. This procedure is repeated until the new weights are computed. In the event that the settings have been modified, after being calculated, the client then transmits them to the FIELDS server. FIELDS clients, in contrast to centralized learning, only exchange the updated model parameters, which were trained on the local data. Thisis in contrast to centralized learning. A new updated global model is generated by the FIELDS server by applying the average update to the aggregated updated parameters that arereceived from the various FELIDS clients. Last but not least, the FELIDS server sends the revised global model parameters to all of the FELIDS clients so that they may get further enhancements based on their newly acquired local data.

Algorithm 2: Client Algorithm

Input: Parameters *D, B, E*
Output: None
Algorithm *StartClient(D, B, E)*:
  $P \leftarrow$ PreProcess(D);
  while *ServerConnect()* **do**
    FedAvg($w \leftarrow$ FetchParams());
  end
  SaveParams(w);
Function *FedAvg(w)*:
  $B \leftarrow$ Split(P, B);
  for $i = 1, \ldots, E$ **do**
    for $b \in B$ **do**
      $w \leftarrow w - \eta \nabla f(w, b)$;
    end
  end
  return *w to Server*;

5. Findings and Experiments

To train deep learning & federated learning models, the NSL-KDD dataset had been preprocessed and divided into test and training sets. The federated learning configuration ensures data confidentiality and safety by establishing an encrypted communication protocol among the central server & clients. The test dataset was used to assess the global model that was created by aggregating local model changes. With 125,973 records for training and 22,544 records for testing, we acquired and prepared the NSL-KDD dataset. For the objectives of training and validating the model, the dataset for training was divided into two sets: one for validation and the other for training [7].

The following was involved in the federated learning setup: To protect the confidentiality of the information, we developed a secure communication protocol that would allow the central server to communicate with the clients while encrypting and authenticating their data. We specified the hyperparameters and global model architecture for the federated learning system.

Here are the steps involved in the local model update: The local datasets were used to train the models by each client. After deciding on an appropriate machine learning technique, we used the datasets from each client to train a local model. For every customer, the particular model parameters, denoted as θi, were revised.

The NSL-KDD dataset was employed to train our model. The training set comprised 125,973 records, while the test set comprised 22,544 records. To train and verify the model, we divided the training dataset into two sets, one for training and one for validation, using a 4:1 ratio. After the training was finished, we used the test dataset to see how well the model performed. The global model M was obtained by aggregating the local model updates θ1, θ2,..., θN through the utilization of federated averaging as well as a customized aggregation function.

To examine how well the global model M performed, we used the NSL-KDD dataset's test dataset to quantify the model's accuracy and several intrusion detection measures.

In terms of accuracy and loss values, Table 3 compares federated learning with deep learning. In general, and particularly as training continued, the findings showed that federated learning might outperform deep learning in terms of accuracy. Keep in mind that larger loss values could accompany this enhanced precision.

In terms of accuracy and loss values, Table 3 compares federated learning with deep learning. In general, and particularly as training continued, the findings showed that federated learning might outperform deep learning in terms of accuracy. Taking into consideration the possibility that this enhanced accuracy would be accompanied by greater loss values is an essential consideration.

It must be noted, nevertheless, that the dataset and model architectural choices might impact these results. This means that these variables may affect the outcomes. There is a deep learning column and a federated learning column in this table. In this training table, each row stands for a distinct iteration. Deep learning had an initial accuracy of 78.33% and federated learning of 94.40%, while the loss values were 1.9984% and 0.5441%, respectively, as shown in the first row of the table.

During the course of the training, both models significantly improved in terms of their accuracy and loss. The results of the fifth phase or epoch demonstrate that federated learning did better than deep learning, with an accuracy rate of 96.54% as well as a loss of 18.1923%. This is in contrast to the outcomes of deep learning, which had an effectiveness of 85.37% as well as a loss of 0.6459%.

In the tenth round, or epoch, federated learning maintained its superiority over deep learning, which had a loss of 38.5893% and an accuracy of 97.15%. In contrast, the deep learning results had a loss of 0.4505. In conclusion, the

results from the twentieth round, also known as the epoch, showed that deep learning had a loss of 0.2824% and an accuracy of 94.53%, while federated learning had the best overall performance at 97.77%.

Table 3 displays the results of the testing conducted on federated learning and deep learning.

TABLE 3. Accuracy and Loss for Deep Learning and Federated Learning

Learning Types	Deep Learning		Federated Learning	
	Accuracy	Loss	Accuracy	Loss
1st (round/epoch)	78.33%	1.9984%	94.40%	0.5441%
5th (round/epoch)	85.37%	0.6459%	96.54%	18.1923%
10th (round/epoch)	91.19%	0.4505%	97.15%	38.5893%
20th (round/epoch)	94.53%	0.2824%	97.77%	58.3108%

In general, the table illustrates that federated learninghas the potential to attain a better level of accuracy than deep learning, especially as the training process advances. However, this may come at the expense of a larger loss rate. On the other hand, it is important to keep in mind that the outcomes may differ based on the particular dataset and model architecture that was used.

In contrast to the deep learning technique, which might take up to fifteen epochs to get equivalent performance, the federated learning strategy achieved acceptable loss and accuracy already in the first round (Figures 3 and 4). Furthermore, the findings show that both methods used the same amount of data every round or epoch, but federated learning achieved better outcomes than deep learning. On the other hand, federated learning's validation loss rose in tandem with the number of rounds, suggesting that overfitting happened sooner with this method.

As shown in Figure 5, the confusion matrix for federated learning is shown. Accuracy, precision, recollection, and the F1-score are some of the significant metrics that can be calculated with the use of this information since it offers a clear split of the true positive results, the real negatives, fake positives, and false negatives. Having a higher F1-score implies that there is a greater balance between the accuracy and recall, which suggests that the classifier is more trustworthy for tasks that include binary classification [23]. This indicates that the model was successful in making accurate predictions for about 98.067% of the total cases, as it attained an accuracy of 98.067%. There was a 97.4% percent success rate, or 0.974 percent accuracy, in the model's positive instance identification. Taking into account both recall and accuracy, the F1 score came out to 98.210%, demonstrating excellent overall performance. Furthermore, the rate of true positives (TPR), which is sometimes referred to as sensitivities or recollection, was 0.99058, which indicates that the model accurately detected 99.058 % percentage among the positive tests. Furthermore, the model has a low rate of false positives (FPR) of 0.03075%, which indicates that it had a very small amount of mistakes that were considered to be false positives.

The confusion matrix for federated learning is shown in Figure 6. Additionally, a confusion matrix graphic is shown in the supplied text. Viewed here are the true positive (TPR) and false positive (FPR) rates for a federated learning model over a range of threshold settings.Based on the data shown in the figure, the accuracy is 98.067%, the precision is 0.974%, the F1-score is 98.210, the TPR is 0.99058%, and the FPR is 0.03075% at the threshold that was previously selected.

Based on these measures, it seems that the model performs very well in terms of precision, accuracy, and F1 score, and it also has a low percentage of false positives.

There are a number of limitations and challenges associated with federated learning in systems for intrusion detection. These include the following: data variation, overhead for communication, data imbalance, concerns regarding security and privacy, model aggregation weaknesses, client availability issues, a lack of a global point of view, and difficulties in managing model drifting. For FL in IDS to become more efficient and safe, it is vital that these difficulties be addressed.

6. Conclusions

In conclusion, the primary objective of our study was to assess the efficacy of federated learning in terms of improving systems for intrusion detection (IDS) that are designed to protect confidential information. The distributed machine learning technique known as federated learning makes it possible to train a shared model in a collaborative manner while yet preserving the decentralization of data and the confidentiality of the information.

Our studies and comparisons with classic deep learning models have shown that federated learning, when using a randomly chosen group of clients, surpasses deep learning both in terms of accuracy and loss in an Intrusion Detection System (IDS). This benefit is especially noteworthy in situations where the protection and confidentiality of data are of utmost importance.

Through the use of federated learning, we successfully created worldwide models minus the need of exchanging sensitive data, thus minimizing the potential dangers linked to data breaches or unauthorized disclosure. Our research findings suggest that federated learning holds the potential to transform the creation of IDS solutions, making them more powerful, streamlined, and secure.

The research study used the NSL-KDD dataset, a well utilized database for network intrusion detection. It built a horizontal learning federation architecture with average aggregation & random client selection. The experimental results repeatedly showed that the federated learning methodology outperformed traditional deep learning approaches in terms of accuracy and loss.

This finding is very significant, since it enables the development of more accurate and reliable intrusion detection systems that can effectively prevent security breaches and protect sensitive data in several sectors like healthcare, fi- nance, and national security.

FIGURE 3. shows the outcomes of deep learning.

FIGURE 4. Results of federated learning are shown.

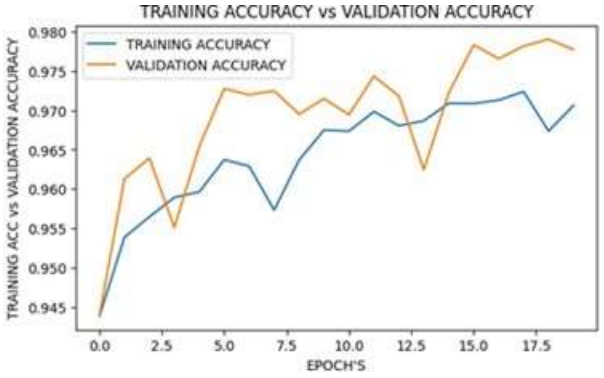

FIGURE 5. Results of federated learning are shown.

Subsequent investigation in this field might go deeper into improving the federated learning technique, examining its implementation on other datasets, and tackling the obsta- cles associated with scalability and diverse client settings.In summary, federated learning has significant potential in enhancing the privacy of systems that detect intrusions and bolstering network security.

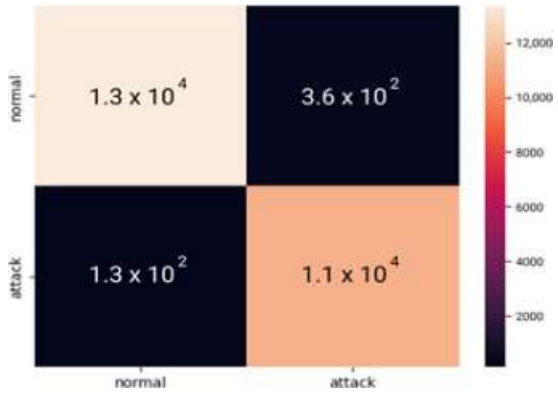

FIGURE 6. The confusion matrix for federated learning is shown .

References

[1] Khraisat, A.; Alazab, A. , "A critical review of intrusion detection systems in the internet of things: Techniques, deployment strategy, validation strategy, attacks, public datasets and challenges." Cybersecurity 2021, 4, 18.

[2] Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, JSurvey of intrusion detection systems: Techniques, datasets and challenges. Cybersecurity 2019, 2, 20.

[3] Alazab, A.; Khraisat, A.; Singh, S."A Review on the Internet of Things (IoT) Forensics: Challenges, Techniques, and Evaluation of Digital Forensic Tools. In Digital Forensics-Challenges and New Frontiers; Reilly, ' Reilly,D.D., Ed.; IntechOpen: Rijeka, Croatia, 2023; Chapter 10'

[4] Alazab, A.; Khraisat, A.; Alazab, M.; Singh, S. , "Detection of obfuscated malicious JavaScript code. Future Internet " 2022, 14, 217.

[5] Agrawal, S.; Sarkar, S.; Aouedi, O.; Yenduri, G.; Piamrat, K.; Alazab, M.; Bhattacharya, S.; Maddikunta, P.K.R.; Gadekallu, T.R. "Federated learning for intrusion detection system: Concepts, challenges and future directions." Comput. Commun. 2022, 195, 346–361.

[6] Victor, N.; Alazab, M.; Bhattacharya, S.; Magnusson, S.; Mad- dikunta, P.K.R.; Ramana, K.; Gadekallu, T.R."Federated learning for iout: Concepts, applications, challenges and opportunities" arXiv 2022, arXiv:2207.13976.

[7] Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A "A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 "IEEE Symposium on Computational Intelligence for Security and Defense Applications ,Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.

[8] Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, AHybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Deci-

sion Tree Classifier and One Class Support Vector Machine, Electronics 2020, 9, 173.

[9] Ghimire, B.; Rawat, D.B. Recent advances on federated learning for cy- ber security and cybersecurity for federated learning for internet of things. IEEE Internet Things J,2022, 9, 8229–8249.

[10] Sun, T.; Li, D.; Wang, B. , "Decentralized federated averaging. "IEEE Trans. Pattern Anal. Mach. Intell 2022, 45, 4289–4301.

[11] Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.; Poor, H.V; "Federated learning with differential privacy: Algorithms and performance analysis."IEEE Trans. Inf. Forensics Secur.2020, 15, 3454–3469.

[12] Fereidooni, H.; Marchal, S.; Miettinen, M.; Mirhoseini, A.; Möllering, H.; Nguyen, T.D.; Rieger, P.; Sadeghi, A.R.; Schneider, T.; Yalame, H.et al. "SAFELearn: Secure aggregation for private federated learning. In Proceedings of the 2021" emphIEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 27 May 2021; IEEE: Piscataway, NJ, USA; pp. 56–62.

[13] Liu, Y.; Kang, Y.; Xing, C.; Chen, T.; Yang, Q.A secure federated transfer learning framework IEEE Intell. Syst. 2020, 35, 70–82.

[14] Hu, L.; Yan, H.; Li, L.; Pan, Z.; Liu, X.; Zhang, Z. "MHAT: An efficient model-heterogenous aggregation training scheme for federated learning," Inf. Sci. 2021, 560, 493–503.

[15] Elahi, F.; Fazlali, M.; Malazi, H.T.; Elahi, M. "Parallel fractional stochastic gradient descent with adaptive learning for recommender systems." IEEE Trans. Parallel Distrib. Syst,. 2022, 1–14.

[16] So, J.; He, C.; Yang, C.S.; Li, S.; Yu, Q.; Ali, R.E.; Guler, B.; Avestimehr, S. Lightsecagg: A lightweight and versatile design for secure aggregation in federated learning. Proc. Mach. Learn. Syst2022, 4, 694–720.

[17] Xing, H.; Xiao, Z.; Qu, R.; Zhu, Z.; Zhao, B., "An efficient federated distillation learning system for multitask time series classification.emphIEEE Trans. Instrum. Meas. "IEEE Trans. Instrum. Meas. 2022, 71, 1–12.

[18] Friha, O.; Ferrag, M.A.; Shu, L.; Maglaras, L.; Choo, K.K.R.; Nafaa, M."FELIDS: Federated learning-based intrusion detection system for agricultural Internet of Things. "J. Parallel Distrib. Comput. 2022, 165, 17–31.

[19] 1Attota, D.C.; Mothukuri, V.; Parizi, R.M.; Pouriyeh, S. "An ensemble multi-view federated learning intrusion detection for IoT. "IEEE Access. 2021, 9, 117734–117745.

[20] Rahman, S.A.; Tout, H.; Talhi, C.; Mourad, A."Internet of things intru- sion detection: Centralized, on-device, or federated learning?"IEEE Netw. .2020, 34, 310–317.

[21] Nguyen, T.D.; Marchal, S.; Miettinen, M.; Fereidooni, H.; Asokan, N.; Sadeghi, A.R. DÏoT: A federated self-learning anomaly detection system for IoT. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 756–767.

[22] McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.Y "Communication-efficient learning of deep networks from decentralized data." In Proceedings of the Artificial Intelligence and Statistics, PMLR, Ft. Lauderdale, FL, USA, 20–22 April 2017

[23] Alazab, A.; Khraisat, A.; Singh, S.; Bevinakoppa, S.; Mahdi, O.A. "Routing Attacks Detection in 6LoWPAN-Based Internet of Things." Electronics 2023, 12, 1320.

[24] Ammar Alazab ,Ansam Khraisat ,Sarabjot Singh and Tony Jan "Enhancing Privacy-Preserving Intrusion Detection through Federated Learning" Electronics 8 August 2023