*Journal of Computer Science*

Original Research Paper

# Fed-Deep CTRL (Light GBM)-Scalable, Interpretable, and Privacy-Preserving Federated Learning Framework for Wireless Intrusion Detection System

**Sudaroli Vijayakumar and V. Muthumanikandan**

*Department of Computer Science, Vellore Institute of Technology, Chennai, India*

*Article history*
Received: 18-07-2023
Revised: 15-12-2023
Accepted: 20-12-2023

Corresponding Author:
Sudaroli Vijayakumar
Department of Computer Science, Vellore Institute of Technology, Chennai, India
Email: oli.sudar@gmail.com

**Abstract:** A learning-based intrusion detection system automates the understanding and reporting of network traffic information. The systems using neural networks and machine learning have demonstrated good detection accuracy, but the accuracy is entirely reliant on the type and volume of data. Additionally, there are issues with its scalability, privacy, efficiency, and interpretability. With the innovative fed-deep CTRL (light GBM) method, this study focuses on creating a local-global federated architecture. On the federated learning framework, a novel method was developed using a mix of deep CTRL and Light GBM. Multiple clients in fed-deep CTRL handle the extraction of local attack data features that the server uses to train the detection to become more effective. The fed-deep CTRL was tested using the AWID dataset and its results were validated individually for each attack type. The overall accuracy for Amok was 97.21%, ARP was 96.42%, the beacon was 98.31%, caffe latte was 97.81%, CTS was 100%, death was 98.92%, disasso was 97.99% and evil twin was 98.46%. The overall accuracy of the model was observed to be 99.60%, the training overhead with 4 clients was 3.28 sec and the approach was shown to be highly interpretable, scalable, and secure.

**Keywords:** Wireless Intrusion Detection System, Federated Learning, Wireless Local Area Network WLAN 802.11, Federated Deep Neural Networks with Controllable Rule Representations (Fed-Deep CTRL)

## Introduction

Human progress in life is undergoing a full paradigm shift as a result of advancements in artificial intelligence, machine learning, and deep learning. Innovation is steadily moving in the direction of complete customization. Absolute privacy is required for complete personalization and corporations surely use these technological advancements to protect their network infrastructure. Because it directly affects strategic choices, the sort of information that flows into and out of an organization's network is crucial. As a result, each organization's directive is to monitor for data entering and departing the network. The intrusion detection system typically does this preparation for data verification.

The way different devices communicate with one another has reportedly grown significantly. Despite the existence of IoT and other types of connection networks, the fundamental concept behind making the devices communicate with one another is through a wired or wireless network. Owing to the infrastructure's complexity, wired media is not always preferable and Wireless Fidelity (Wi-Fi) has taken over as the industry standard for communication. For enabling high-quality communication, Wi-Fi is essential. As a result of the open nature of the wireless communication medium and the fact that anybody may tune into the frequency, security breaches are, unfortunately, extremely prevalent in this type of network. A simple brute force attack can be used to acquire full access to the data being communicated.

Traditional encryption techniques like Challenge Handshake Authentication Protocol (CHAP) and others are insufficient to protect user privacy in wireless networks. So, it is essential to have an intrusion detection system that can monitor, identify, and forecast the type of attack. Any intrusion detection system is built on this

principle and it has been asserted that utilizing machine learning or deep learning techniques, such intrusion detection offers great prediction accuracy (Cetin *et al*., 2019). False alarms produced by wireless intrusion detection systems are one issue that needs to be addressed by researchers. A few models based on deep learning and machine learning are presented to reduce the false alarm rate. Support Vector Machine (SVM), Markov models, and deep neural networks, among other conventional machine learning techniques, could not manage false alarms and detect attacks like a human (Khan and Gumaei, 2019). Recently, a lot of deep learning studies have been put forth to create an effective intrusion detection system (Merity *et al*., 2017; Vinayakumar *et al*., 2019). Although these intrusion detection systems are capable of classifying the attacks, they do have some drawbacks. The following are a few of the constraints that keep the study of intrusion detection systems lively:

- The accuracy of learning-based models is strongly influenced by the kind and amount of training data, however, this does not guarantee data privacy
- Most attack detection by learning-based models is merely a form of notification that is frequently handled by human intervention
- The machine learning and deep learning models' durability and scalability are inexplicable (Aldweesh *et al.*, 2020)
- The training procedure for machine and deep learning algorithms must be carried out centrally, which not only consumes a lot of computational resources but also affects classification accuracy and the number of false alarms. Classification accuracy declines and false alarm rates rise as network size climbs (Bhattacharya *et al*., 2020)

The main goal of this study is to provide a method that can be scaled, is interpretable, and produces few false positives. To overcome the issues with intrusion detection, we are proposing a method called fed-deep CTRL that uses federated learning (Tang *et al*., 2018; Agrawal *et al*., 2022). In this manner, the raw data from different devices stays on the client device; instead, the models are given, ensuring WLAN 802.11 customer privacy, and the attack outcomes are managed by the deep CTRL algorithm, which makes sure the neural network is governed by the rules.

Thus, the major contributions of this study are as follows:

- The first is to suggest a client-server architecture with federated learning for wireless intrusion detection, in which only network-wide sharing of privacy-preserving data occurs for training purposes
- Second, a deep CTRL method is suggested that can effectively handle both trained samples and adversarial samples, ensuring that it can manage both
- Another significant contribution of this study is its scalability, which allows for the client-side addition of fresh attack categories and low-cost classifier updates to reflect these new categories of data
- Deep CTRL, the fundamental classification algorithm, is capable of performing simultaneous rule and data learning. As the rules are under control, a higher degree of learning controllability is attained, preventing the need for retraining

As a result, the suggested framework includes a novel learning methodology that combines the benefits of federated learning with deep CTRL. This methodology not only aids in the development of a WIDS with a greater level of accuracy, but it also resolves concerns with interpretability and scalability. Using samples from the Aegean Wireless Intrusion Dataset (AWID), the categorization's accuracy and the quantity of false alarms are evaluated.

Wireless networks play a major part for everybody to remain associated with technological advancements. This innovation has developed conjointly picked up adequate notoriety. In any case, when it's talking about the security of wireless networks indeed a basic sniffing instrument can help somebody gain access to the entire network. There's the presence of progressed encryption plans and wireless intrusion detection systems. However cyber dangers are expanding colossally. This segment clarifies the essentials of wireless networks and wireless intrusion detection systems and the convenience of federated learning in anomaly locations.

### *Wireless Networks WLAN-802.11*

Wi-Fi 802.11, often known as wireless LAN, uses the wireless medium as a communication medium. The majority of the communication takes place through the access points. Wi-Fi uses the Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) standard as its fundamental convention, however, there are other ways to use it. Base stations are allowed to speak with one another with or without shaking hands if the communication channel is clear. Continuous acknowledgment is given for each transmission made between stations. Since the beginning, there have been significant problems that have made it difficult for people to accept this technology because open security concerns exist. As technology developed, intrusion detection systems became more and more common to handle these security issues. Although wireless intrusion detection systems have advanced considerably, signature-based or anomaly-based location is still the fundamental method used to identify unusual patterns or attacks. Both methods make an effort to scan

the entire network for unauthorized access points, which can then be used to identify attacks.

The wireless intrusion detection system is often installed with a large number of sensors that thoroughly screen the communication channel and communicate the data at a consistent rate. Servers within the context of the database and administration servers are in charge of monitoring the sensor data that has been collected. The database server maintains the log data, while the administration server uses the log data to work with the administrators to address any unpleasant events on the network.

Our understanding from this implementation is that the management server's technique determines how well the wireless intrusion detection system can detect attacks. The management server uses anomaly-based detection as its primary methodology because signature-based attack identification is fully dependent on predetermined signatures and the system frequently lacks knowledge of how to respond to newer types of logs.

To put it simply, anomaly-based detection looks for unusual behavior. When an unusual traffic pattern is seen in the network, it is considered an attack and is most often referred to as an anomaly. The most typical type of traffic is something that occurs regularly. A wireless intrusion detection system makes an effort to spot anomalous traffic patterns, which can assist in deleting unauthorized access points and intruders from the network. The machine/deep/reinforcement-based algorithms power the majority of recent anomaly detectors. To understand the problems associated with machine, deep, and reinforcement anomaly detectors, let's review some of the recent works using these approaches in the next section.

### *Related Works in Machine/Deep/Reinforcement Learning for Anomaly Detection*

Hypothetically, a few machine learning-based approaches tried with the benchmark intrusion dataset like Network Service Laboratory-Knowledge Discovery Database (NSL-KDD), Defense Advanced Research Project Agency(DARPA), University of New South Wales (UNSW) and Aegean Wireless Intrusion Dataset (AWID) have given strategies that can progress the intrusion detection system exactness rate whereas keeping the false-negative rate to a least. The foremost utilized algorithms fall beneath distance or density-based and one of the common issues while managing with intrusion datasets is the most elevated number of unlabelled features. Feature building shapes a fundamental portion whereas building a proficient intrusion detection system and (Ambusaidi *et al*., 2016) feature selection based on support vector machine-made noteworthy decrease within the number of features both at the training and testing level. Kim and Cha's (2005), comparative work on support vector machines is suggested for identifying attacks using two different UNIX command sets. The whole observational research on the use of support vector machines in identifying attacks has given them an accuracy of around 87%.

The use of deep learning to increase intrusion detection systems' accuracy rate is enormous (Dalal *et al*., 2022). An 88% multiclassification precision rate for a multimodal location using deep autoencoders and LSTM (He *et al*., 2019) allows it to learn transitory data from an adjacent network relationship. The stacked autoencoder and deep neural network are used in Wang *et al*. (2018) to significantly improve intrusion detection. The self-learning characteristics of Kolias *et al*. (2015) are used to handle risky samples that are thought to be problematic. By using a ladder deep learning network with a focused loss function, which may be able to handle the challenging data well, the idiosyncrasies are identified. The focus of (Ran *et al*., 2019) is similar; it focuses on self-learning the features that increased the classification accuracy for the four classes of attacks that are present in the AWID intrusion dataset. In managing intrusion detection data, random forest-based classification is frequently used. According to Daesung (Thing, 2017), client-based behaviors are used to identify assaults. This more effectively fulfills the function of an intrusion detection system. The findings from the aforementioned research offer us a rough notion of the training set's significance.

When using a solid training set using machine learning and deep learning techniques, intrusion detection has produced results wi$^{th}$ good classification accuracy. The privacy of the training data is jeopardized by the storing and transfer of information because the processing must be done at the central server. Decentralized training is suggested to maintain this and some of the current decentralized learning strategies are reviewed. Federated learning to improve the click-through rate of the Gboard application is confirmed by Yang *et al*. (2018), with a progressed click-through rate of 33%. Federated learning is used to handle Firefox URL placement recommendations based on client interactions, as mentioned in Moon *et al*. (2017). Most of the time, users of browsers sort in and choose the suggestions made by the browser. By pooling this data, Firefox may employ suitable machine-learning-based algorithms to optimize search for users. Since bookmarks, histories, and search queries are completely private to the users, they are not included in this information gathering. By incorporating this data, the browser results for which federated learning is used can advance and the improvements will entirely be dependent on the data provided by the user.

Training takes place at the user's workplace and the federated learning initiative comprises data from many clients gathered there. Using this method generates more data points and selecting the relevant one from the enormous collection is crucial. Hartmann *et al*. (2019); Caldas *et al*. (2018) illustrate methods that can be

modified to recognize the benchmark picture dataset since federated learning is advantageous because picture training typically entails a larger set of images. Caldas *et al.* (2018) use a more condensed benchmark dataset to determine the importance of data at the client point. This analysis of a subgroup of unique clients produced an accuracy that was 25% more effective. It may be necessary as a result to carry out a comparison inside the AWID dataset under consideration. Coordination of the blockchain enables accountability of the integrated federated approach. Federated learning is described by Nguyen *et al.* (2019) as decentralizing data and reviewing pertinent materials while receiving incremental updates. The evaluation of the suggested method is put to the test using anomaly detection and the configuration suggested by Nguyen *et al.* (2019) may be used in various use cases. This study further demonstrated that the performance is not much impacted by the presentation of blockchain on federated learning. This thought should be carefully customized in recognizing the anomalies since the smaller benchmark dataset AWID could be an imbalanced dataset.

According to the studies, processing the vast amount of network traffic data in a centralized server takes time. The procedure of moving data to the central server for processing and storage takes time and compromises security. Therefore, a decentralized training method that can do the training on the user's device is essential. As a result, we suggest a federated learning configuration for decentralized data training. The majority of machine and deep learning-based intrusion detection systems have strong classification accuracy, but they struggle when faced with emerging types of attacks. This is because the labeled data is utilized for training (Agrawal *et al*., 2022), necessitating the need for a system that can analyze the data in real-time without using static data. Here, decentralized training for the real-time data is attempted. The next part explains the general principle of decentralized learning.

### *Federated Learning-Based Intrusion Detection System*

Federated learning breaks the concept of centralized training and model building. Collaborative model training performs distributed training at the user’s edge devices. The participating components in federated learning are the edge devices and central servers. Data collected at every node train and build the model at every node and the aggregation of it serves as a central model. Transfer level information from nodes to the central model is only the model parameters and not the raw data. This preserves the confidentiality of the messages generated at nodes. This methodology is used for intrusion detection to distinguish normal and attack patterns.

Either vertical or horizontal federated learning can be used to distribute data to the clients. When the client group is similar but different clients employ different numbers of features, vertical federated learning can be used. Banking and e-commerce are two common industries in which vertical federated learning is used. Because the feature space for all the clients is comparable, vertical federated learning is not employed in attack detection, hence horizontal federated learning is typically used instead. The AWID dataset was used in this instance to train the classifier for identifying intrusions using WLAN 802.11 statistics data. The feature space for this data is typically the same for all clients, however, the number of samples may vary slightly. Therefore, the federated learning utilized in this study is horizontal.

Some of the papers that look at network intrusion detection systems using the idea of horizontal federated learning are (Seo *et al*., 2021; Al-Jarrah *et al*., 2023; Li *et al*., 2020, Zhao *et al*., 2020, Preuveneers *et al*., 2018). Upon understanding the various works using machine learning and deep learning approaches, the accuracy results for classification were higher, however, they failed to address the problem of training overhead, scalability of the decentralized training model, and the classification happening considering the analysis of the traffic. So, the problem is thus formulated in section 3.1 based on these observations on conventional approaches. So, the proposed method considers achieving interpretability, scalability, and privacy in federated learning. The next section briefs the proposed methodology.

## Materials and Methods

This part outlines the problem at hand, provides background information on the study's motivation and goals, and then moves on to the system formulation.

### *Problem Formulation*

Consider a straightforward network with a client and a server. In this case, the client is a person who is receiving security-related services and the server is essentially the one who is providing those services. If the security service provider now chooses to offer an intrusion detection service on the network, this service can be produced by a model that can make choices based on traffic data and the behavior of the client. Therefore, each customer must supply information about their features and traffic for the service provider to construct a model. Privacy is jeopardized when this is transmitted via the network, though. Due to the training taking place on the server of the service provider in this case, only the security issue is raised. Therefore, to get around this, we can avoid privacy concerns if the training data isn't transferred to the central server and if training takes place at the client location. This is the primary reason for federated learning's adaption in this study.

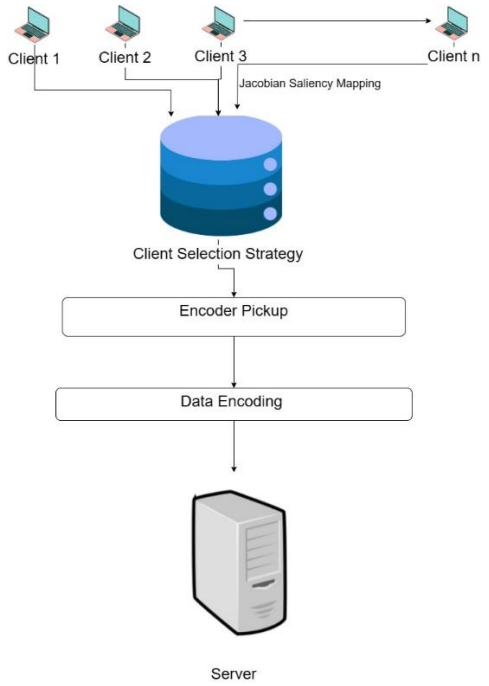


**Fig. 1:** System architecture of the proposed system

The second driving force behind this research is the belief that when a security service provider offers a service like attack detection, the trained model must have knowledge of the various categories of the attack and that the model shouldn't completely fail if a newer type of attack is discovered. Another crucial point is that manual help should be minimized to avoid the newest varieties of attacks. Deep CTRL is used in this study since the objective is to find a classification algorithm with strong interpretability.

The creation of a robust, generalized, and scalable model is the third driving force for this research. The attack detection model should be able to accept and make judgments based on both attack growth and client growth because the number of attackers and the types of attacks that are generated daily are growing exponentially (Bonawitz *et al*., 2019). This means that a highly interpretable model can develop into a robust and scalable model.

The objective is further refined to provide a novel learning-based WIDS for multi-class classification based on these reasons. The resulting WIDS model should be comprehensible, scalable, reliable, and secure for the training set of data. The design objectives are summarised as follows:

- Data privacy: Every customer should be protected for privacy, along with their traffic data
- Accuracy: The performance score measured in accuracy for detecting cyberattacks should be high
- Interpretability: Every stage of prediction should be easy enough for humans to infer
- Scalability: The model should be able to handle newer categories of attacks without modifying the defined architecture

## *System Design and Implementation*

This section explains the system configuration and the overall architecture that is adopted for achieving decentralized training and attack classification.

## *Client Configuration*

The attack classification model that is designed here adopts horizontal federated learning and the setup is notated $c_1$, $c_2 \ldots c_n$ where the total number of clients is $n$. Suppose a client $c_2$ holds the data $x_2$ which is denoted as $y_2$, the client data $x_2 \in \mathbb{R}^{number\ of\ samples \times size\ of\ the\ feature\ space}$. This signifies that the clients in the horizontal federated learning setup share the same feature space.

## *Data Security*

Another form of configuration that is adopted in this methodology is to ensure data privacy the actual data is masked. This masking is carried out with the introduction of noise and this is carried out on $x_n$ *and* $y_n$. For the actual data, random samples are generated using the Jacobian-based saliency map attack ($M$) and $M$ is the hyperparameter for noise quantification. For the label $y_n$ random data masking is carried out.

## *Workflow Model*

The design of fed-deep CTRL includes four major steps and the overall flow is explained in this section. Figure 1 shows the system architecture of the proposed system.

## *Client Selection*

To prevent getting the same type of data from two distinct clients, this procedure is the initial stage in the federated learning process. The reason for performing this step is that data redundancy directly impacts training effectiveness (Wang *et al*., 2022). The client selection strategy takes a greedy approach, in which a preliminary survey is carried out for each client to determine the types of attacks and the quantity of data that each client has (Tuor *et al*., 2020). The main goal is to choose customers with sample sizes significantly lower than the training complexity value, yet even with lesser sample sizes, it should still be able to cover all the categories. In order for each client to be included in the fed-deep CTRL, the data distribution must contain samples that are both inclusive of all category samples and less difficult than the training complexity parameter. Algorithm 1 illustrates the algorithmic process used for client selection.

**Table 1:** Notations summary

| Notation | Description |
|---|---|
| $n$ | Number of participating clients |
| $C_i$ | The $i^{th}$ participating client |
| $x_i$ | Feature data |
| $y_i$ | Labelled data |
| $\in$ | Privacy data |

**Algorithm 1:** Client selection procedure

**Input:** All the clients $\{C_1, C_2, \cdots C_n\}$ and the data distribution expected / the number of data samples of a particular attack class belonging to the selected client is $D^i = \left[D_1^i, D_2^i, \cdots D_n^i\right]$

**Output:** Set of all selected clients $C_{selected} \leftarrow \{S_1, S_2 \ldots\ldots\ldots\ldots\ldots\ldots S_k\}$

**Parameter:** Training Complexity $T_t$

**Initialization**

$C_{selected} \leftarrow C_i^0$
$i_0 \leftarrow argmin[1, D] i \in [1, D] var(D^i)$
$C_{all} \leftarrow \{C_1, C_2, \cdots C_n\} C_i^0$
$D \leftarrow D_i^0$

**Procedure**

- do
  - minvar ← 0
  - for every client $C_i$ in $C_{all}$
    - do
      - temp ← D + $D^i$
      - if var(temp) < minvar ^ sum(temp)≤ $T_t$ then
      - minvar←var(temp)
      - $i_{min} \leftarrow i$
    - end
  - end
- if $i_{min}$ is none
  - break
- end
- /* parameter updation */
- D ← D + D$i^{min}$
- $C_{all} \leftarrow$ C$i^{min}$
- $C_{selected} \leftarrow C_{selected} \cup$ C$i^{min}$
- return $C_{selected}$

Because the objective is to choose a client who has a balance between the number of samples and attack category and because it should be less than the training complexity, Algorithm 1 computes variance. In light of this, the variance problem is defined as stated in Eq. (1):

$$Var(D) = Var[D_1^i, D_2^i, \cdots D_n^i] \quad (1)$$

The Algorithm 1 has been used to solve the variance problem using the greedy method. This algorithmic technique is carried out for every customer and the clients are selected in such a manner that minimizes the variance and minimizes the training complexity. The data distribution balance we aim for is achieved by solving variance as shown mathematically in Eq. (2):

$$Var(D) == \frac{\sum_{i=1}^{n}(D_i - \bar{D})x^2}{D} \quad (2)$$

In this manner, the required clients that are essential for training are selected and they are encoded as encoders in the deep CTRL. The labeled data and rules make up the trained encoders as a result. There are two separate encoders for data and rules and the possibility exists to modify the rule strength without performing the retraining. The construction of deep CTRL, with the incorporation of the tree structure, depicts the prediction process as illustrated in Fig. 1. Humans can easily understand this prediction method for any adjustment. All client's deep CTRL-trained encoders are uploaded to the server. Table 1 explains the variables and functions used in the algorithm.

### *Encoder Pickup Procedure*

There are numerous clients and the server has access to their data as a trained encoder, as was the case in Step 1. With the addition of fresh data, the server is now required to process the data with the aid of numerous clients. Consequently, encoders who will participate in the training and testing procedure must be chosen among those encoders that are available. Assume a DoS attack is the fresh data for which the forecast is being made and the attack categories are Denial of Service (DoS), User to Root (U2R), and Remote to user (R2L). When choosing an encoder, make sure that it covers every attack category, either by itself or in combination with other encoders. The encoders picked up should satisfy the below conditions:

- $\forall_{category\ class} \in \{1,2,3 \ldots\ldots category_n\}$ the federated model $f_i$ includes all the classes in the class $category_n$

For the selection of the best encoders, the procedure is mentioned in algorithm 2. The procedure iteratively selects the encoder that possesses all the attack classes and that is notated as the best$_{encoder}$. The parameters are continuously updated till all the encoders are covered.

**Algorithm 2:** Best encoder selection strategy

- **Input:** $C_{selected}$ $\{c_1, c_{2,\ldots\ldots\ldots\ldots} c_i\}$encoders where the class set is $D_i$
- **Output:** Picked up encoders $Encoder_{samples}$
- **Parameters:** Set of attack classes $\{1,2,3 \ldots\ldots\ldots\ldots D\}$
- **Initialization variables**

- $Encoder_{sanples}$ = null
- $Class\ _{Samples}$ = null
- $Encoder_{left}$={ $c_1, c_{2,\ldots\ldots\ldots\ldots} c_i$ }

- Procedure (encoder find)
  - do
    - for $c_i$ in $Encoder_{left}$ do
      - if |c ∩ $c_i$| > 0 then
      - $E_{best} \leftarrow c_i$
      - end
    - end for
- end procedure
- procedure (parameter update)
  - $Class\ _{Samples}$ =$Class\ _{Samples}$ ∪ $Class\ _{best}$
  - $Encoder_{sanples}$=$Encoder_{sanples}$ ∪ $Encoder\ _{best}$
- Return $Encoder_{sanples}$

Once the best encoder is selected using algorithm 2 this selected encoder is sent from the server to the other participating clients for the data encoding process that will be performed by each of the clients.

### *Client Data Encoding*

Now that the clients have the encoder they downloaded from the server, they try to encrypt it using their own data. Since there is no information loss when softmax is included in the encoding process, it joins a softmax probability. The number of data columns is decreased depending on the type of problem being evaluated, however, this reduction shouldn't have an impact on the function, hence softmax is utilized. As a result, each participating client generates both the label data and the encoding information. For the purpose of the training process, these details are provided back to the server. The data is homomorphically encrypted before being sent back to the server to ensure data privacy.

### *Deployment*

The Deep CTRL classifier is combined with the chosen encoders on the server side, resulting in the generation of soft max probabilities first, followed by the deep CTRL classifier's final predictions. Anyone can comprehend how the predictions are made because the deep CTRL has been adjusted to behave in a manner resembling that of a tree. The training procedure of deep CTRL is shown in Algorithm 3.

**Algorithm 3:** Deep CTRL training procedure

**Input:** Training data of all participating clients { $c_1, c_{2,\ldots\ldots\ldots\ldots} c_i$ }, Deep = {($x_i$,$y_i$): i={1,2,…….C} in which x represents the private data and y denotes the label.
**Output:** Trained Model = Encoder ∩ classifier

**Parameters:** Training Complexity $T_t$, encoder complexity $E_c$

- Call client selection procedure (Algorithm 1)
- {$C_1$ , $C_2$, ……$C_k$ } ← $C_{selected}$←{$C_1$ , $C_2$, ……$C_i$, $T_t$}
- for every i in 1,2…..k do
  - $E_i$= deep CTRL($x_i$,$y_i$)
  - Initialize Rule encoder $\emptyset_r$ data encoder$\emptyset_d$, decision block$\emptyset$ , and distribution $p(\propto)$
  - Check if convergence does
    - Obtain minibatch $Deep_{mini}$ from Deep and from $p(\propto)$
    - $Z_r = \emptyset_r(Deep_{mini})$
    - $Z_d = \emptyset_d(Deep_{mini})$
    - $Z = \alpha_b Z_r \oplus (1 - \alpha_b Z_d)$
    - Compute Loss and update gradients
  - end if
  - end for
- $Encoder_{sanples}$ = call Encoder Selection (Algorithm 2)
  - for 1 ≤ i ≤ k do
    - homo encryption = $Encoder_{sanples}(x_i)$ + $truncated\ Laplacian\ (\frac{2}{\epsilon})$
  - end for
  - end

The rule and data encoder is trained using the latent representation, which is extracted from the labeled data and encoder. These combined taught representations show that $Z$ is an amalgam of and, with the symbols for these two learned representations being and. Here, random is used to enhance learning and the model's ability to understand new types of input. Therefore, to make the model simpler to control and govern, modifications may cause it to behave differently.

### *Experimentation and Evaluation*

This section presents the experimental evaluation of the proposed system as well as the robustness research that was conducted using the AWID dataset.

### *Experimental Setup*

The AMD Ryzen 5900 CPU and 64 GB of memory are used for the experiment. The construction of client and server setups is done using the FL Sim simulation framework, while training is done using Federated Averaging (FedAvg) (Liu *et al*., 2020).

### *Dataset*

Since the security of wireless networks is a key component of our discussion, a suitable dataset that accurately represents the real wireless network serves

as the main focus of our study (Anish, 2015). Since this data is gathered using a well-defined wireless setup that includes 10 terminals, including notepads, cell phones, and smart TVs, the AWID dataset (Wireless Datasets, 2022) is the most widely used wireless intrusion dataset. Internet surfing, leaking, and other methods are used to gather the common attack plans, and a specialized Kali-based Linux was used to generate attack activity. Due to the practical difficulties in organizing the enormous dataset, AWID creates a subsection with 155 features called AWID-ATK-R. 1,765,000 records are included in here, of which 1,62,358 are adversaries from 17 distinct classifications. Figures 3-4 shows the AWID-ATK-R preparation and test dataset's course conveyance plot.

### *Baseline Model*

Federated Matched Averaging (FedMA) (BMcMahan *et al*., 2017) is the baseline model used in this study. In this model, different neurons are added to the baseline model's 72 similar neurons. The aggregation is more significant because permutation invariance is used for the neurons in this. 13 neurons make up the output layer. Three classifiers-extreme gradient boosting (XgBoost), Light Gradient Boosting Machine (LightGBM), and deep CTRL-are used to confirm the centralized training's outcomes. The optimal time to use deep CTRL is after the client data has been delivered to the server. Since the quality and amount of the data have a significant impact on the neural network's performance, empty values are initially replaced with 0, then scaling and normalization operations are carried out. The total, the squared sum, and the number of samples are utilized in the client setting to calculate the average as well as the standard deviation. The neural network employs client batch sizes of 32, 64, 128, and 256 with a learning rate of 0.01. There are between 1000 and 1500 epochs. Since the raw data is unaffected by noise or magnitude, it is utilized to evaluate the deep CTRL.

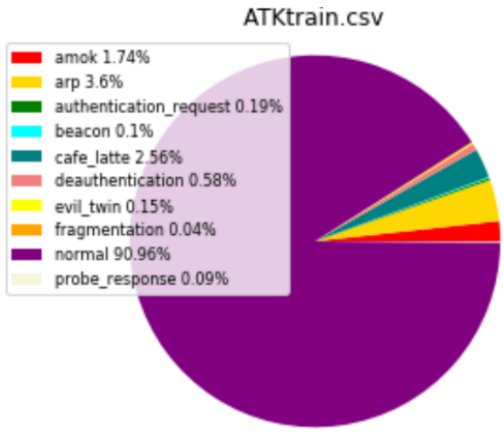


**Fig. 3:** Class distribution plot of the training dataset

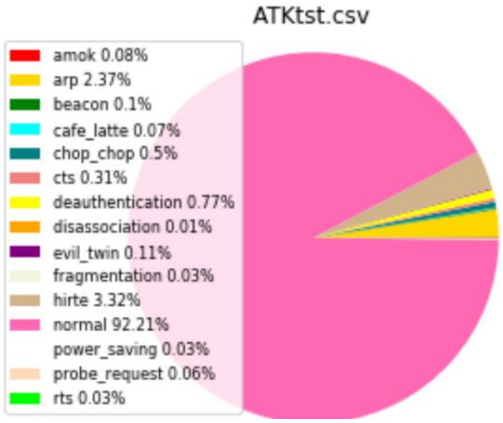


**Fig. 4:** Class distribution plot of the test dataset

### *Evaluation Metrics*

The number of correctly classified samples is obtained using accuracy. Accuracy is the most common performance indicator that identifies the built model's classification result. This value is obtained from the ratio of the correct data to the considered total data:

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_P + F_n} \quad (3)$$

The attack detection performance is measured using the *F*1-*score* serves as one of the most important evaluation metrics that gives a direct indication if the value is high, true positive is also high and there is a diminishment of false:

$$F1 - Score = \frac{2T_p}{2T_p + F_P + F_n} \quad (4)$$

## Results

By contrasting it with other models, the suggested method, fed-deep CRTL, is evaluated in terms of classification accuracy. As the overall objective of this study was to achieve high accuracy, interpretability, scalability, and privacy, its efficiency in terms of training complexity and training overhead is also significant.

### *Baseline Model Comparison*

The length of the dataset is the training complexity and to show the client and server model variation if at the client XG Boost is used and the server is deep CTRL then X + D is used. Table 2 shows the F1-scores various models. Understanding how well the model detects threats is essential to determining how many false alarms are produced by the system. Table 2, the effectiveness of attack-wise detection is assessed using F1-scores. As can be seen, the FedMA reported a greater missing rate and comparable results are obtained with other federated averaging methods as well.

**Table 2:** F1-scores for the attacks in the test set for fed MA, XgBoost, light GBM, and fed-deep CRTL

| WIDS model | Amok % | ARP % | Beacon % | Caffe latte % | Chop chop % | CTS % | Deauth % | Disasso % | Evil twin % |
|---|---|---|---|---|---|---|---|---|---|
| FedMA 3 layer | 92.13 | 91.21 | 95.42 | 93.21 | 92.18 | 90.100 | 90.20 | 92.31 | 94.21 |
| FedMA 7 layer | 93.41 | 95.21 | 94.31 | 92.12 | 92.46 | 90.310 | 93.21 | 92.41 | 90.10 |
| XGBoost | 94.11 | 94.03 | 93.96 | 90.12 | 95.32 | 94.690 | 94.32 | 94.21 | 94.32 |
| LightGBM | 95.21 | 94.96 | 94.21 | 94.22 | 95.01 | 95.360 | 95.56 | 95.42 | 94.32 |
| Fed Deep CTRL | 97.21 | 96.42 | 98.31 | 96.57 | 97.81 | 1.000 | 98.92 | 97.99 | 98.46 |

**Table 3:** Accuracy scores of different federated and non-federated learning models

| WIDS model | Architecture | AWID dataset % |
|---|---|---|
| Semi boosted nested Mananayaka and Chung (2023) | Ensembled | 95.26 |
| DRL + RSFNN Reyes *et al*. (2020) | Deep belief | 95.40 |
| RL Lopez-Martin *et al*. (2021) | Reinforcement | 95.72 |
| AFS-RF Wang *et al*. (2020) | ANN, DT, RF, SVM | 99.39 |
| FedMA | 3-layer | 65.27 |
| | 5-layer | 68.17 |
| | 7-layer | 72.13 |
| Fed-deep CTRL (proposed) | XG + XG | 89.90 |
| | LG + LG | 99.60 |
| | XG + LG | 92.31 |
| | LG + XG | 95.61 |

**Table 4:** Training overhead values for 3, 5- and 7-layer FEDMA is compared with fed-deep CTRL

| WIDS Model | 3-layer | 5-layer | 7-layer | Fed-deep CTRL |
|---|---|---|---|---|
| One client | 16.75s | 32.47s | 56.82s | 0.82s |
| Multiple client (4) | 67s | 129.88s | 227.28s | 3.28s |
| Server | 120.23s | 240.46s | 480.92s | 330.22s |
| Simulation time | 820.27s | 1640.54s | 3281.08s | 453.27s |

The detection performance of fed-deep CTRL is higher than that of the other baseline models. The CTS attack, which is not present in the AWID-ATK-train, may be identified with 100% accuracy as a de-authentication attack, in Table 3. As a result, the suggested fed-deep CTRL can handle hostile samples and produce alerts for an assault. This in turn effectively reduces the number of false alarms. Classification accuracy scores for different models are reported in Table 3. As obtained from the table, the fed-deep CTRL exhibits a higher accuracy value compared to all other models. This was tested with different partitions with homogenous and heterogenous data and achieved good accuracy, thus showing scalability.

The fed-deep CTRL with Light GBM classifier produces improved accuracy that is simple to understand and the training complexity is also decreased, as can be seen from Table 3. Fed-deep CTRL is chosen as the foundation model. The training overhead is also been observed for the proposed fed-deep CTRL in terms of single, and multiple clients, and servers and is tabulated in Table 3.

With CPU used for training, the overhead for fed-deep CTRL is very low at the client because the client encoders are very small and the hyperparameter updation in fed-deep CTRL is less, making it lightweight as well.

## Discussion

The experimentation revealed that the proposed FedDeepCtrl classifier results especially about the server are sensitive to the hyperparameters. Tuning the hyperparameters had an influence on the performance of the server classifier. In this study, the hyperparameters are tuned manually which consumes a lot of time thus auto hyperparameter tuning would be a great option and this is left as a future work.

Though this study can achieve generalizability, scalability, and privacy, when the attack category class is higher this system consumes heavy cost. This is because the time consumption is linear in the number of attack classes. So, the only way to bring down the cost is to make the attack classes to a minimum. The cost consumption can be made much less by grouping similar attack classes in a single group manually.

## Conclusion

Privacy protection is essential for intrusion detection. To maintain this anonymity, a novel fed-deep CTRL setup that combines federated learning with a modified deep CTRL and incorporates a tree-like structure using the light GBM is used. If the training takes place in a centralized location, the federated learning model simply behaves the same in terms of accuracy. These simulation results appear to be more accurate since this is localized training, which makes it easier to detect invasive behaviors locally. The suggested fed-deep CTRL has superior scalability, interpretability, and efficiency to the conventional federated learning algorithms as a result of considerable experimentation. Though it promises privacy, the privacy can still be enhanced by incorporating a client-side homomorphic encryption technique and this is left to be pursued as a future work.

## Acknowledgment

I would like to thank Dr. Ganapathy Sannasi for his advice and mentoring. His knowledge of security was quite helpful in determining the course of our study and improving our techniques. Their astute recommendations and kind criticism were helpful in improving the overall caliber of this study.

## Funding Information

The authors did not receive support from any organization for the submitted work. No funding was received to assist with the preparation of this manuscript.

## Author's Contributions

**Sudaroli Vijayakumar:** Worked on the results and paper written.

**V. Muthumanikandan:** The whole paper was reviewed and suggestions were given.

## Ethics

Hereby, I D. Sudaroli Vijayakumar consciously assure that for this manuscript the following is fulfilled:

1) This material is the authors' own original work, which has not been previously published elsewhere
2) The paper is not currently being considered for publication elsewhere
3) The paper reflects the authors' own research and analysis in a truthful and complete manner
4) The results are appropriately placed in the context of prior and existing research
5) All authors have been personally and actively involved in substantial work leading to the paper, and will take public responsibility for its content

## References

Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Alazab, M., ... & Gadekallu, T. R. (2022). Federated learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications*. https://doi.org/10.1016/j.comcom.2022.09.012

Aldweesh, A., Derhab, A., & Emam, A. Z. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy and open issues. *Knowledge-Based Systems*, *189*, 105124. https://doi.org/10.1016/j.knosys.2019.105124

Al-Jarrah, O. Y., El Haloui, K., Dianati, M., & Maple, C. (2023). A novel detection approach of unknown cyber-attacks for intra-vehicle networks using recurrence plots and neural networks. *IEEE Open Journal of Vehicular Technology*, *4*, 271-280. https://doi.org/10.1109/OJVT.2023.3237802

Ambusaidi, M. A., He, X., Nanda, P., & Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Transactions on Computers*, *65*(10), 2986-2998. https://doi.org/10.1109/TC.2016.2519914

Anish, N. (2015). Packet analysis with wireshark. *Google Books*. https://books.google.com/books/about/Packet_Analysis_with_Wireshark.html?id=UzzlCwAAQBAJ

Bhattacharya, S., Maddikunta, P. K. R., Kaluri, R., Singh, S., Gadekallu, T. R., Alazab, M., & Tariq, U. (2020). A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU. *Electronics*, *9*(2), 219. https://doi.org/10.3390/electronics9020219

BMcMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. *In Artificial Intelligence and Statistics*, 1273-1282. PMLR.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., ... & Roselander, J. (2019). Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, *1*, 374-388.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečný, J., McMahan, H. B., ... & Talwalkar, A. (2018). Leaf: A benchmark for federated settings. *ArXiv Preprint ArXiv: 1812.01097*. https://doi.org/10.48550/arXiv.1812.01097

Cetin, B., Lazar, A., Kim, J., Sim, A., & Wu, K. (2019). Federated wireless network intrusion detection. *In 2019 IEEE International Conference on Big Data (Big Data)*, 6004-6006. IEEE. https://doi.org/10.1109/BigData47090.2019.9005507

Dalal, N., Akhtar, N., Gupta, A., Karamchandani, N., Kasbekar, G. S., & Parekh, J. (2022, January). A wireless intrusion detection system for 802.11 WPA3 networks. In *2022 14th International Conference on Communication Systems and Networks (COMSNETS)*, 384-392. IEEE. https://doi.org/10.1109/COMSNETS53615.2022.9668542

Hartmann, F., Suh, S., Komarzewski, A., Smith, T. D., & Segall, I. (2019). Federated learning for ranking browser history suggestions. *ArXiv Preprint ArXiv: 1911.11807*. https://doi.org/10.48550/arXiv.1911.11807

He, H., Sun, X., He, H., Zhao, G., He, L., & Ren, J. (2019). A novel multimodal-sequential approach based on multi-view features for network intrusion detection. *IEEE Access*, *7*, 183207-183221. https://doi.org/10.1109/ACCESS.2019.2959131

Khan, F. A., & Gumaei, A. (2019). A comparative study of machine learning classifiers for network intrusion detection. *In Artificial Intelligence and Security: 5th International Conference, ICAIS 2019, New York, NY, USA, July 26-28, 2019, Proceedings, Part II 5*, 75-86. Springer International Publishing. https://doi.org/10.1007/978-3-030-24265-7_7

Kim, H. S., & Cha, S. D. (2005). Empirical evaluation of SVM-based masquerade detection using UNIX commands. *Computers and Security*, *24*(2), 160-168. https://doi.org/10.1016/j.cose.2004.08.007

Kolias, C., Kambourakis, G., Stavrou, A., & Gritzalis, S. (2015). Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Communications Surveys and Tutorials*, *18*(1), 184-208. https://doi.org/10.1109/COMST.2015.2402161

Li, B., Wu, Y., Song, J., Lu, R., Li, T., & Zhao, L. (2020). DeepFed: Federated deep learning for intrusion detection in industrial cyber-physical systems. *IEEE Transactions on Industrial Informatics*, *17*(8), 5615-5624. https://doi.org/10.1109/TII.2020.3023430

Liu, Y., Garg, S., Nie, J., Zhang, Y., Xiong, Z., Kang, J., & Hossain, M. S. (2020). Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, *8*(8), 6348-6358. https://doi.org/10.1109/JIOT.2020.3011726

Lopez-Martin, M., Sanchez-Esguevillas, A., Arribas, J. I., & Carro, B. (2021). Network intrusion detection based on extended RBF neural network with offline reinforcement learning. *IEEE Access*, *9*, 153153-153170. https://doi.org/10.1109/ACCESS.2021.3127689

Mananayaka, A. K., & Chung, S. S. (2023). Network intrusion detection with Two-Phased hybrid ensemble learning and automatic feature selection. *IEEE Access*. https://doi.org/10.1109/ACCESS.2023.3274474

Merity, S., Keskar, N. S., & Socher, R. (2017). Regularizing and optimizing LSTM language models. *ArXiv Preprint ArXiv:1708.02182*. https://doi.org/10.48550/arXiv.1708.02182

Moon, D., Im, H., Kim, I., & Park, J. H. (2017). DTB-IDS: An intrusion detection system based on decision tree using behavior analysis for preventing APT attacks. *The Journal of Supercomputing*, *73*, 2881-2895. https://doi.org/10.1007/s11227-015-1604-8

Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., & Sadeghi, A. R. (2019). DÏoT: A federated self-learning anomaly detection system for IoT. *In 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 756-767. IEEE. https://doi.org/10.1109/ICDCS.2019.00080

Preuveneers, D., Rimmer, V., Tsingenopoulos, I., Spooren, J., Joosen, W., & Ilie-Zudor, E. (2018). Chained anomaly detection models for federated learning: An intrusion detection case study. *Applied Sciences*, *8*(12), 2663. https://doi.org/10.3390/app8122663

Ran, J., Ji, Y., & Tang, B. (2019). A semi-supervised learning approach to IEEE 802.11 network anomaly detection. *In 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 1-5. IEEE. https://doi.org/10.1109/VTCSpring.2019.8746576

Reyes, A., D. Vaca, F., Castro Aguayo, G. A., Niyaz, Q., & Devabhaktuni, V. (2020). A machine learning based two-stage Wi-Fi network intrusion detection system. *Electronics*, *9*(10), 1689. https://doi.org/10.3390/electronics9101689

Seo, S., Arik, S., Yoon, J., Zhang, X., Sohn, K., & Pfister, T. (2021). Controlling neural networks with rule representations. *Advances in Neural Information Processing Systems*, *34*, 11196-11207.

Tang, F., Mao, B., Fadlullah, Z. M., & Kato, N. (2018). On a novel deep-learning-based intelligent partially overlapping channel assignment in SDN-IoT. *IEEE Communications Magazine*, *56*(9), 80-86. https://doi.org/10.1109/MCOM.2018.1701227

Thing, V. L. (2017). IEEE 802.11 network anomaly detection and attack classification: A deep learning approach. *In 2017 IEEE Wireless Communications and Networking Conference (WCNC)*, 1-6. IEEE. https://doi.org/10.1109/WCNC.2017.7925567

Tuor, T., Wang, S., Ko, B. J., Liu, C., & Leung, K. K. (2020). Data selection for federated learning with relevant and irrelevant data at clients. *ArXiv Preprint ArXiv: 2001.08300*, 64. https://doi.org/10.48550/arXiv.2001.08300

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, *7*, 41525-41550. https://doi.org/10.1109/ACCESS.2019.2895334

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., & Khazaeni, Y. (2020). Federated learning with matched averaging. *ArXiv Preprint ArXiv: 2002.06440*. https://doi.org/10.48550/arXiv.2002.06440

Wang, S., Li, B., Yang, M., & Yan, Z. (2018). Intrusion detection for Wi Fi network: A deep learning approach. *In International Wireless Internet Conference*, 95-104. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-06158-6_10

Wang, W., Jian, S., Tan, Y., Wu, Q., & Huang, C. (2022). Representation learning-based network intrusion detection system by capturing explicit and implicit feature interactions. *Computers and Security*, *112*, 102537. https://doi.org/10.1016/j.cose.2021.102537

Wireless Datasets. (2022). Wireless technologies have become prevalent in the last few years. *University of the AEGEAN*. https://icsdweb.aegean.gr/awid/

Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., ... & Beaufays, F. (2018). Applied federated learning: Improving google keyboard query suggestions. *ArXiv Preprint ArXiv:1812.02903*. https://doi.org/10.48550/arXiv.1812.02903

Zhao, R., Yin, Y., Shi, Y., & Xue, Z. (2020). Intelligent intrusion detection based on federated learning aided long short-term memory. *Physical Communication*, *42*, 101157. https://doi.org/10.1016/j.phycom.2020.101157