Nivaashini *et al.*
EURASIP Journal on Information Security (2024) 2024:8
https://doi.org/10.1186/s13635-024-00156-5

EURASIP Journal on Information Security

RESEARCH

Open Access

FEDDBN-IDS: federated deep belief network-based wireless network intrusion detection system

M. Nivaashini[1], E. Suganya[2], S. Sountharrajan[3], M. Prabu[3] and Durga Prasad Bavirisetti[4*]

Abstract

Over the last 20 years, Wi-Fi technology has advanced to the point where most modern devices are small and rely on Wi-Fi to access the internet. Wi-Fi network security is severely questioned since there is no physical barrier separating a wireless network from a wired network, and the security procedures in place are defenseless against a wide range of threats. This study set out to assess federated learning, a new technique, as a possible remedy for privacy issues and the high expense of data collecting in network attack detection. To detect and identify cyber threats, especially in Wi-Fi networks, the research presents FEDDBN-IDS, a revolutionary intrusion detection system (IDS) that makes use of deep belief networks (DBNs) inside a federated deep learning (FDL) framework. Every device has a pre-trained DBN with stacking restricted Boltzmann machines (RBM) to learn low-dimensional characteristics from unlabelled local and private data. Later, these models are combined by a central server using federated learning (FL) to create a global model. The whole model is then enhanced by the central server with fully linked SoftMax layers to form a supervised neural network, which is then trained using publicly accessible labeled AWID datasets. Our federated technique produces a high degree of classification accuracy, ranging from 88% to 98%, according to the results of our studies.

Keywords Federated learning (FL), Deep learning (DL), Wi-Fi attacks, Intrusion detection system (IDS)

1 Introduction

The advancement of Wi-Fi technology, such as mobility and cost-effectiveness, has made it highly popular for communication purposes, especially with the widespread use of mobile devices and Internet of Things (IoT) devices. However, compared to traditional computer networks, Wi-Fi networks are slower and demand additional security measures. This is because data packets are transmitted through the air, making them vulnerable to interception and manipulation, thereby increasing the susceptibility of Wi-Fi networks to various types of attacks. Consequently, there is an immediate need for Wi-Fi security defense strategies that are swift, affordable, and efficient. One valuable security technique in combating wireless intrusions during network transmissions is server-side intrusion detection.

As they keep track of system activities and proactively identify potential threats, intrusion detection systems (IDS) are essential for maintaining security in this context. New machine learning (ML) models and new threat detection methods have been created. The initial version of the IDS was very basic and used log events and tuples are updated from a database of signatures to mix them together. However, it soon became clear that

*Correspondence:
Durga Prasad Bavirisetti
durga.bavirisetti@ntnu.no
[1] Department of Artificial Intelligence and Data Science, Sri Eshwar College of Engineering, Coimbatore, Tamilnadu, India
[2] Department of Information Technology, Sri Sivasubramaniya Nadar College of Engineering, Chennai, Tamilnadu, India
[3] Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Chennai, India
[4] Department of Computer Science, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

these approaches have significant drawbacks, particularly rigidity. They were unable to identify new threats that had not yet been added to the signature database, which was one of their disadvantages, because of their lack of proactiveness.

Next-generation IDS began progressively adding a type of knowledge to find new threats as a moderating component. The normal subsequent stage is to make it workable for them to trade information about newfound risks, permitting everyone to rapidly distinguish new attack courses and diminish their general effects. As these systems continued to become more precise and capable of identifying new threats, this became a logical development.

Machine learning techniques, specifically wireless network intrusion detection systems (WNIDS), have been successfully employed in detecting wireless intrusions. Several studies have been conducted in the field, documenting the development of various WIDs using different machine-learning methodologies [1, 2]. The most well-known AI techniques are profound braindeep neural networks (DNN), artificial neural networks (ANN), and support vector machines (SVM). WNIDs created using conventional machine learning or deep learning methods are effective in recognizing such anomalous events. However, obtaining information about wireless networks for server-side machine learning training is both time-consuming and costly for customers.

To address these challenges, a cooperative machine-learning approach has been put forward as a solution. In federating learning, training is shared among multiple users while still taking place on the device, and user data never leaves the users' devices. The federated learning algorithms, originally developed by researchers at Google, involve the aggregation of models created on individual user devices, where these models are transferred from multiple devices to a centralized location. This approach ensures data privacy by only sharing the models and not the raw data.

FL has emerged as a potential technique for addressing the difficulties of sensitive data exploitation and the flow of information between many parties [3]. Because only local models and not local data are sent to the server, FL offers a workable solution to this issue. The FL approach was used because it facilitates data collection while maintaining privacy. We measured the classification accuracy and communication cost during training by using performance measures and the FEDDBN algorithm. The AWID intrusion detection dataset was utilized in our proposed model, and performance measures were employed to examine the outcomes.

Below is a breakdown of the remainder of this paper. The literature-based research on ML, DL, FL, and AI in IDS is the main topic of Section 2. Knowledge of the information and methods utilized in this study is provided in Section 3. For our suggested model, we employed a variety of methodologies, which are listed in Section 4, along with brief descriptions of each. The experiments covered in the methodology section are emphasized in Section 5, which also examines the findings. Section 6 concludes by summarizing the work done and addressing work that will be done in the future.

2 Knowledge about data and techniques

2.1 Intrusion detection system

In the realm of security and protection for hosts and networks, intrusion detection systems (IDS) play a vital role in automatically detecting and assessing unusual and hazardous behaviors. The primary objective of intrusion detection is to identify instances of intrusion, often establishing criteria to determine the validity of the evidence presented. An intrusion occurs when a computer or network deviates from its normal operations and becomes a platform for attacks aimed at compromising or manipulating network data [1, 4, 5]. In today's interconnected landscape, where private information is exchanged and stored, IDS serves as a cybersecurity tool alongside antivirus and firewall software, with the firewall additionally conducting preliminary inspections of web traffic [6]. IDS systems play a crucial role in managing, monitoring, and maintaining the normal operation of networks, promptly alerting network administrators upon the detection of abnormal activities or potential threats [7, 8].

Typically, an IDS consists of three fundamental phases: the initial phase involves the collection of evidence pertaining to cyberattacks from input data, which is then subjected to processing to identify the second phase, which is the cyberattacks themselves. The final phase of an IDS report provides comprehensive information about these identified attacks. Recent advancements in machine learning and deep learning have facilitated the ability to predict both typical and atypical network activities, as well as previously unknown network attacks [9, 10]. These technological advancements have significantly enhanced the capabilities of IDS systems, enabling more accurate detection and classification of network intrusions.

2.2 Deep learning

Deep learning (DL) has emerged as a prominent area of focus among security researchers and industry experts. It encompasses artificial intelligence (AI) models known as deep neural networks (DNNs), which draw inspiration from the workings of the human brain. DL, often

synonymous with deep learning, enables the automatic extraction of meaning from vast amounts of data without the need for domain expertise. An essential aspect of this process is featuring extraction, a critical step in feature engineering that requires prior knowledge [11]. Effective classification necessitates proper feature extraction. The utilization of neural networks (NNs) in machine learning (ML) can be traced back to the 1950s when NNs were able to automatically extract and classify features without human intervention. At that point, conventional neural networks served their purpose adequately.

However, the advent of advanced neural networks, commonly referred to as deep learning (DL), has obviated the need for feature engineering [12–14]. DL, ML, and NNs are frequently intertwined in discussions about artificial intelligence, leading to misconceptions. Neural networks (NNs) constitute the foundation of deep learning (DL) within the broader field of machine learning. The process of acquiring information and identifying patterns mirrors the functioning of the human brain. Deep learning heavily relies on neural networks (NNs), and the term "many NNs" often denotes a significant number of them [15]. As NNs become deeper, challenges such as vanishing and exploding gradients arise, necessitating robust computing systems.

Advancements in computing systems have facilitated notable progress in various aspects of deep learning (DL), including designs, optimizers, activation functions, loss functions, and addressing challenges like vanishing and exploding gradients. DL has found extensive applications in addressing diverse cybersecurity challenges, consistently outperforming traditional machine learning methods [16]. DL architectures can be broadly categorized into generative and discriminative models. In our approach, we leveraged innovative ideas by incorporating recurrent structures, deep belief networks, deep autoencoders, and deep Boltzmann machines. To differentiate these novel concepts from previous ones, we incorporated recurrent structures and convolutional neural networks (CNNs).

2.3 Federated learning

The underlying concept of federated learning revolves around constructing machine learning models using decentralized datasets from numerous devices while ensuring data privacy. Federated learning represents an innovative approach that involves computing an updated model by retrieving the latest model and employing local IoT data from IoT devices. After these privately trained models are transmitted from the IoT devices to the central server for aggregation, a unified and enhanced global model is disseminated back to the IoT devices, often through techniques like weight averaging. Effective data distribution plays a pivotal role in implementing federated learning, considering the associated technological and practical limitations [17].

At present, federated learning can be categorized into three distinct types. The first type is federated horizontal learning, which is applied when datasets share a common feature space but possess different sample spaces. The second type is vertical federated learning, employed when two datasets exhibit distinct feature spaces but share the same sample space. Finally, federated transfer learning comes into play when the feature and sample spaces of the datasets differ from each other.

3 Related works

3.1 ML and DL in IDS

The extensive body of literature extensively delves into the realm of intrusion detection, exploring a wide array of methodologies and algorithms grounded in the principles of machine learning and deep learning. Consequently, this particular section aims to shift its focus towards pre-existing solutions and methodologies that effectively harness the potential of deep learning techniques [18, 19]. One such pioneering approach is known as Auto-IF (Autoencoder and Isolation Forest), which has been meticulously crafted to discern anomalies within fog networks. Employing the mechanism of binary classification, this method scrutinizes inbound traffic to ascertain its legitimacy or malicious intent. Addressing the challenge of disparate datasets, the integration of an autoencoder (AE) comes into play, effectively extracting regular traffic data while eliminating training attacks. By capitalizing on the output of the AE, the isolation forest adeptly detects outliers, thus culminating in a highly efficient anomaly detection system. The evaluation of this method revolves around the utilization of the NSL-KDD dataset, whose findings unquestionably validate its exemplary performance, yielding an accuracy of 95.4%, precision of 95.81%, recall of 97.25%, and F-measure of 96.01%.

Subsequently, the adoption of the covering-based element extraction unit and feed-forwards profound brain organization strategies ensues, employing the WFEU extra-tree method to construct concise feature vectors. To comprehensively gauge the efficacy of the proposed intrusion detection system (IDS), two datasets, namely UNSW-NB15 and Aegean Wi-Fi intrusion dataset (AWID), are employed. Within the UNSW-NB15 dataset, the application of WFEU generates 22 and 26 attribute feature vectors for binary and multiclass classification, respectively, attaining accuracies of 87.10% and 77.16%. On the other hand, the proposed strategy culminates in a binary classification accuracy of 99.66% and a multiclass classification accuracy of 99.77%. In order to provide a

benchmark for comparison, the performance of the proposed methodology is juxtaposed against SVM, KNN, RF, DT, and NB within the realm of machine learning.

Empirical studies revolving around the AWID dataset reveal its versatile applicability across both wired and wireless network domains. However, it is essential to acknowledge that the suggested system might experience performance limitations stemming from the temporal constraints of the testing process and the intricacy of computational tasks. Exploring the prospect of utilizing robust hardware offers a potential avenue to circumvent these constraints [20, 21]. The deployment of recurrent neural networks (RNN) boasting multiple layers has emerged as a promising endeavor, warranting thorough evaluation against the reliable NSL-KDD dataset. This strategy encompasses two pivotal phases: traffic analysis and categorization. The initial phase entails preprocessing the data for deep neural network (DNN) processing, ultimately resulting in the classification of pre-processed data as either legitimate or malicious. Deep proportional recursive networks, in conjunction with various backpropagation algorithms, are harnessed to establish an optimized IDS through rigorous training. By virtue of its traffic analysis capabilities, this method engenders robust real-time security in the context of Internet of Things (IoT) scenarios, promptly sounding security alarms upon the detection of malicious activities. Noteworthy accomplishments include a DoS detection rate of 98.27%, R2L detection rate of 97.35%, U2R detection rate of 64.93%, and a commendable overall performance in accordance with the method's examination.

In a bid to capitalize on the spatial information embedded within traffic data, the gradient boosting decision tree (GBDT) strategy is parallelized, while the temporal data is handled through the utilization of a technique known as gated recurrent unit (GRU). By synergistically combining the spatial and temporal features extracted by the GBDT and GRU methods, a comprehensive IDS model is meticulously crafted. The evaluation of this model is conducted using the CAS2018 dataset, developed by a dedicated research project team. The outcomes of the evaluation illustrate the enhanced sensitivity of the proposed method in identifying benign instances of Distributed Denial of Service, port scan, infiltration, and web attack traffic, yielding impressive accuracies of 99.9%, 99.9%, and 99.9% respectively [20].

In order to address the immaturity of convolutional neural network (CNN) models in network traffic intrusion detection, a Fast Fourier Transformation (FFT) method has been devised to bolster their efficacy. This method employs FFT to convert each network connection into an image, enabling efficient classification. The characteristics of the data serve as the basis for constructing sequences necessary for the FFT, utilizing numerical representations rather than textual ones. By combining the real and imaginary components, a 64-bit image with three channels is generated. The results unequivocally showcase the efficacy of this conversion technique for both binary and multiclass data. The NSL-KDD dataset serves as the foundation for these experiments, although it should be noted that the limited data within the dataset poses challenges in accurately identifying U2R and R2L attacks. Consequently, the study delves into methods for real-time conversion of network communications into images and addresses the imbalances present within dataset samples [22, 23].

Within the realm of intrusion detection, various implementations are available, encompassing statistical methods, deep neural networks, and machine learning techniques [24, 25]. Notably, deep learning has been investigated as a cutting-edge approach for remote intrusion detection in recent studies [26, 27]. Wang et al. [2] studied attacks in a Wi-Fi environment, associating the consequences of two DNN assemblies and one SAE. The approach discussed in their study surpasses the methodology proposed by Thing [27]. By leveraging the AWID dataset, the network records were categorized into four classes: normal, attacks by injection, attacks by impersonation, and attacks by flooding. While the accuracy of the flooded attack class stands at 73%, the other three classes achieve accuracy rates exceeding 98.3%. Furthermore, an alternative architecture based on the standard ladder network implementation is presented, yielding an impressive overall accuracy of 98.54% [26].

In order to accomplish the reduction of dimensionality and the extraction of meaningful risk indicators, a novel feature extraction method has been devised for implementation within an intrusion detection system (IDS). This method introduces a unique process known as feature Vec2im (vectors to pictures) conversion, followed by a subsequent phase involving fuzzy allocation, which generates fuzzy class memberships from the raw data. The utilization of convolutional neural networks (CNN) in conjunction with a Siamese Aspect 1-d element space allows for the effective packing of these features. When subjected to a trial examination using the NSL-KDD dataset, this method exhibits an impressive accuracy of 86.64%. The study also places a strong emphasis on the application of visual analytics tools within IDS, drawing parallels with their successful implementation in the analysis of healthcare data [28–30].

3.2 Federated learning (FL)

Upon scrutinizing the evolution of intrusion detection systems (IDS), it is firmly established that federated learning (FL) will serve as the bedrock for future

generations of IDS. While FL is a relatively nascent approach with limited applicability to IDS technologies, it remains imperative to undertake a state-of-the-art review to consolidate the existing knowledge and stimulate further investigation by acknowledging the shortcomings within the literature. This study presents a roadmap of best practices to guide future innovations and also examines the apparent lack of uniformity in model evaluation.

In the realm of data centers, gathering data from edge devices and conducting centralized training proves to be impractical due to concerns regarding data privacy, network bandwidth limitations, and equipment availability. To address these challenges, the concept of federated learning has emerged [31]. Acknowledging that each participant may encounter local optima due to their limited data, the author of [32] demonstrates how federated learning enhances the accuracy of local participants' models. Additionally, participants can leverage models trained by others to effectively overcome local optima and obtain more precise models.

The employment of the federated averaging (FedAVG) technique is recommended, along with local stochastic gradient descent (SGD) [33]. Communication costs present a primary constraint in the federated learning context, and the FedAVG method significantly reduces the number of communication cycles required compared to synchronous stochastic gradient descent.

To address the issue of heterogeneous networks and the challenge of united learning calculation intermingling, the authors of [34] introduced FedProx, a federated optimization framework based on FedAVG. LoAdaBoost-FedAVG procedure has been anticipated in [35] to build a prototype founded on local data. The model underwent retraining before being aggregated with the global model, thus enhancing learning efficiency. Simulation results indicate that LoAdaBoost-FedAvg achieves rapid convergence.

Various frameworks and libraries are available for developing FL applications [36]. These frameworks can be categorized based on their primary objectives. Simulation-oriented libraries prioritize extensibility and evaluation goals, providing developers with a wide range of tools for development and benchmarking, such as emulation and virtual devices. On the other hand, production-oriented libraries focus on usability and efficiency, offering enterprise-level solutions for various FL scenarios.

Simulation-focused libraries include TensorFlow Federated (TFF), PySyft, LEAF, and FedML. Production-oriented libraries include FATE [37] and PaddleFL [38]. FedML and PySyft enable the development of adaptable systems through their FL architecture and programmable message exchange. FedAVG and FedProx, two common methods for central party disposal in FL, are incorporated into all the libraries. Some systems, such as PaddleFL, FedML, and FATE, solely rely on vertical data partitioning as their chosen method. Notably, FedML stands out as a highly comprehensive research-oriented library, particularly in terms of supporting diverse federated learning scenarios. Moreover, its modular worker/client design streamlines the coding and implementation processes.

3.3 FL in IDS

Federated learning, initially proposed by Google researchers, represents a novel paradigm [3]. The pioneering study [3, 39] showcased the application of deep CNNs in federated settings for tasks like image classification and next-word prediction. While our intention is to employ a comparable overall architecture, we opted for federated learning in the context of intrusion detection, which poses greater challenges due to data imbalance. Previous articles [40, 41] explored the use of the AWID dataset and KDD 1999 Cup data to investigate intrusion detection within a federated environment. The studies presented here employed SAE models for real use cases of intrusion detection and anomaly detection, demonstrating that the integration of blockchains had minimal impact on federated learning performance.

The convergence of these two technologies has emerged as a popular area of research. Considering that most IDS heavily rely on DL models on edge devices. Due to the prevalence of vanilla FL implementations and the existence of numerous fragmented custom implementations, it is challenging to create a taxonomy based on the type of FL algorithm used. Presently, two prominent groups of FL-IDS prevail: recurrent neural networks (RNNs) [42] and multilayer perceptron (MLP) [43]. Each larger group comprises two smaller subgroups. RNN models encompass Long Short-Term Memory (LSTM) [44] and Gated Recurrent Units (GRU) [45], which differ in terms of neuron subtype. Conversely, MLP models adopt the neural network (NN) architecture. Autoencoders (AE) [46], significant subtypes frequently encountered in the literature, play a notable role.

While Liu et al. [47] did not specifically address ID, they proposed a unique anomaly detection technique with potential application in the cybersecurity industry. Their study introduced attention mechanism-based convolutional neural network long short-term memory (AMCNN-LSTM) architecture, combined with a federated learning framework enabling collaborative training for anomaly detection among IoT devices. Another research effort proposed a method for detecting attacks in UNIX command sequences [48]. Furthermore, the utilization of FL models followed a recommendation

to initially employ centralized learning for two IDS approaches, LSTM and CNN models, before transitioning to FL. In the case of FLs engineering, the LSTM model outperformed the CNN model across all rules, yielding an impressive F1 score of 99.21%. Lin et al. [49] introduced an FL malware classification system that complements rather than replaces an IDS. This system combines federated learning with deep learning architectures such as LSTM and SVM, using a publicly available investigation dataset from Virus Total. Under ideal conditions, the system achieved a classification accuracy of 91.67%.

DeepFed, a federated deep learning scheme [50], integrates an FL framework, DL architecture, and privacy mechanism using the Paillier cryptosystem [51] to conduct identification in cyber-physical systems (CPS). Mothukuri et al. [52] devised a method for leveraging decentralized on-device data for intrusion detection in IoT networks. Another research effort introduced a decentralized FL method and ensemble [53]. When evaluated on the Modbus network dataset, LSTM and GRUs exhibited an average detection accuracy of 90.25%. FedAGRU [54] proposed an IDS for wireless edge networks (WENs) by combining a GRU and an SVM model with a unique FL algorithm. Among the various FL IDS designs, autoencoders (AE) emerged as the most prevalent FL-IDS architectures [55–59], particularly for performing intrusion detection through anomaly detection, leveraging their ability to reconstruct input data based on learned network traffic patterns.

Preuveneers et al. [56, 60] introduced a permissioned blockchain-based [61] federated learning system, where model modifications were securely recorded in the transaction ledger. However, the inclusion of blockchain transactions resulted in a decrease in overall performance by approximately 5% to 15%, even in scenarios where the training process remained transparent. Cetin et al. [58] proposed a wireless IDS that utilized FedAVG and a stacked autoencoder (SAE) for intrusion detection using the AWID dataset [1]. Autoencoders also provide a foundation for incorporating additional edge computing applications into FL-IDS. Notably, Cholakoska et al. [57] explored differential privacy in their study, given the sensitivity of clinical information. Other studies [62–64] employed distinct MLP NN architectures to deliver FL-IDS solutions, showcasing their versatility.

Therefore, federated learning, as introduced by Google researchers, has revolutionized the field by leveraging deep convolutional neural networks in federated settings. Our research builds upon this foundation, specifically applying federated learning to the challenging task of intrusion detection. The categorization of FL-IDS models revolves around the DL architectures used, such as RNNs and MLPs. Noteworthy advancements have been made in anomaly detection, attention-based CNN-LSTM architectures, and the fusion of FL with deep learning approaches like LSTM and SVM. Additionally, the integration of blockchain technology and decentralized FL methods has been explored, alongside the prevalent utilization of autoencoders in FL-IDS for anomaly detection tasks. The continuous exploration and refinement of FL-IDS architectures contribute to the advancement of intrusion detection systems and pave the way for future innovations in this domain. The summary of the existing AI-based IDSs has been grouped and presented in Table 1.

The reasons for choosing the RBM as a deep learning solution are discussed below, and it serves as an overview of the literature review on the current state of wireless network intrusion detection.

The deep learning model can automatically extract superior representation characteristics from large-scale data, and it has strong benefits for handling complicated data. The model's parameters are first set with random weights in most of the works that are now in existence. On the other hand, an increasing amount of research [66, 67] has examined the effects of initialization and pre-training on FL and has proposed that beginning with a pre-trained model might lessen the influence of data heterogeneity as well as the amount of time needed to train the final model. By examining the effects of pre-training in FL-based IDS utilizing a semi-supervised DBN-based RBM deep learning approach, our study adds to the body of knowledge.

To overcome the shortcomings covered in the literature review, the current deep learning technique is improved with the help of the DBN-RBM framework. We tested several activation functions, iteration or epoch durations, and RBM layer topologies. With a 99% success rate, the recommended FEDDBN model-based WNIDS

Table 1 Summary of the survey of AI-based IDS

Study	IDS datasets	AI technique (ML/DL/FL)
[19–23, 28–30]	NSL-KDD dataset	DL
[65]	UNSW-NB15 and AWID datasets	ML
[24–27]	AWID dataset	ML and DL
[40, 41]	KDD 1999 cup and AWID datasets	DL and FL
[42–46]	–	ML, DL, and FL
[47–49]	Virus total malware dataset	ML, DL, and FL
[50–53]	Modbus network dataset	DL and FL
[54–59]	AWID dataset	ML, DL, and FL
[62–64]	AWID dataset	ML and FL

ML Machine learning, *DL* Deep learning, *FL* Federated learning

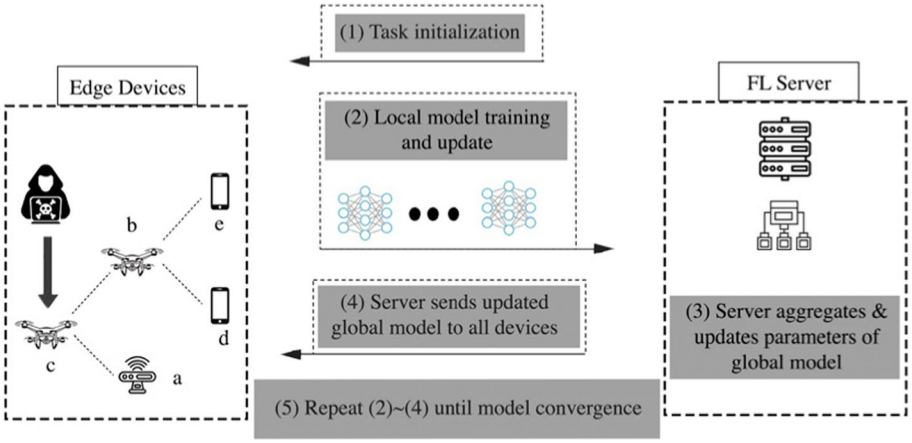

Fig. 1 WNIDS based on federated learning framework

with RBM pre-training model performed better. We also optimized the AWID dataset to increase performance. The optimization process includes both the identification of unique characteristics and the removal of superfluous ones. The proposed DBN-RBM model learns the feature set dynamically for each Wi-Fi network packet instance using optimized and enhanced hyperparameters, hence mitigating the shortcomings of different evasion techniques.

4 Proposed methodology

In this study, we adopted federated learning [3] as the recommended and evaluated approach for creating WNID models. By utilizing federated learning, wireless network devices were able to locally train models using their acquired data. These local models were then aggregated through averaging to form a global model. This methodology allowed edge devices to retain their raw training data, which could potentially contain sensitive information and confidential data. Figure 1 illustrates an approach and shares only the model parameters of the edge devices.

Our federated learning strategy involved classifying incoming network transfers as either legitimate or malicious. To accomplish this, we employed a specific type of deep learning neural network called the stacked restricted Boltzmann machine (RBM), which recorded a compressed representation of abnormal data for anomaly detection purposes.

Within the federated learning environment, algorithms continually updated both their local and global models based on new observations to identify emerging patterns. Implementing federated learning for intrusion detection posed methodological challenges, including feature selection, deep learning model selection, and parameter tuning specific to federated learning. To evaluate the effectiveness of our federated intrusion detection approach, we employed performance metrics such as detection accuracy, precision, recall, and F1 score. Furthermore, we considered the overall utilization of computing resources and communication requirements, including computational operations and the volume of downloaded/uploaded bytes, as crucial aspects of federated learning from edge devices.

4.1 Dataset collection and preprocessing

The proposed methodology was validated using the openly available AWID dataset collected using Wireshark [68, 69]. Table 2 provides the breakdown of the number of instances for each class, including both the normal and the three primary attack types, namely impersonation, injection, and flooding. It is a comprehensive and up-to-date publicly available dataset specifically focused on Wi-Fi networks.

This dataset contains a wide range of examples of typical and anomalous traffic patterns in 802.11 wireless networks [1]. It is divided into two forms, named after the number of samples, as indicated in the tables:

Table 2 Different forms of AWID datasets based on the sample size distribution

Dataset subtype name	Classes	Size	Type	Records
AWID-attack-full-training	10	Full	Train	162,375,247
AWID-attack-full-testing	17	Full	Test	48,524,866
AWID-class-full-training	4	Full	Train	162,375,247
AWID-class-full-testing	4	Full	Test	48,524,866
AWID-attack-reduced-training	10	Reduced	Train	1,795,575
AWID-attack-reduced-testing	15	Reduced	Test	575,643
AWID-class-reduced-training	4	Reduced	Train	1,795,575
AWID-class-reduced-testing	4	Reduced	Test	530,643

AWID-FULL and AWID-REDUCED datasets, each specifying the sample size for the training and testing sets within each partition. Additionally, based on the distribution of class labels, the AWID dataset is further categorized into two classes, as presented in the tables: AWID-ATTACK and AWID-CLASS. The testbed creation of the AWID dataset has been explained in detail in Section 5.1.

Due to its extensive coverage and realistic attributes, the dataset holds the potential to serve as a standardized benchmark for research focused on Wi-Fi networks. For our study, we utilized the AWID-CLS-R-Trn and AWID-CLS-R-tst datasets for training and testing purposes, respectively. The training dataset consisted of 1,795,575 instances, comprising 1,633,190 instances labeled as normal and 162,385 instances labeled as attacks. As for the test dataset, it encompassed a total of 575,643 instances, with 530,785 instances classified as normal and 44,858 instances classified as attacks. Table 2 provides an overview of the distribution of attack classes within both the training and test datasets.

In the dataset, the network's characteristics reflect a scenario where the number of normal instances surpasses the number of attack instances [65]. Specifically, the imbalanced training dataset had a normal-to-attack ratio of 10:1, while the imbalanced test dataset had a ratio of 11:1. Such imbalances in the training model can result in decreased classification accuracy. To mitigate this issue, we took proactive measures to balance the datasets beforehand.

Prior to conducting our tests, we followed Ran's method of preprocessing, normalization, and balancing, as outlined in their work [26]. By adhering to this approach, we successfully obtained a balanced dataset where the number of normal examples equaled the total count of attack instances. This outcome is illustrated in the last column of Table 3, showcasing the achieved result.

To accomplish data distribution, we employed the LEAF approach [39]. The dataset was initially segregated into different classes of devices, enabling us to generate varying example sizes and client counts.

To create a balanced training dataset, we randomly reduced the number of normal instances to 163,319 data instances. Similarly, the test dataset consisted of 53,078 data instances. This approach allowed us to train our proposed method on a balanced dataset while evaluating the learned model on an unbalanced dataset.

Considering the diverse range of value data types present in the AWID dataset [65], it was necessary to perform pre-processing steps prior to analysis. Two key pre-processing steps were employed: normalization and translation of symbolic-valued properties into numeric values. Symbolic qualities were represented using integer values ranging from 1 to *N*, where *N* represents the number of symbols associated with each property. Additionally, hexadecimal data types were converted into integer values. Some attributes in the dataset retained a continuous data type. Furthermore, any attribute without an available value was marked with a question mark ("?"), which was assigned a zero value [70]. Once all attribute values were transformed into integers, each attribute was linearly normalized between zero and one using the normalization equation outlined in Eq. (2).

Table 3 Sample size distribution based on different forms of AWID datasets

Dataset subtype name	Normal	Impersonation	Injection	Flooding
AWID-CLS-R-Trn	1,633,190	65,379	48,522	48,484
AWID-CLS-R-Tst	530,785	16,682	20,079	8,097
Total	21,63,975	82,061	68,601	56,581
Balanced	2,05,285	82,061	68,601	56,581

$$V = \frac{V - V_{MIN}}{V_{MAX} - V_{MIN}} \tag{1}$$

where V represents the normalized value and corresponds to the relevant attribute value, and V_{MIN} and V_{MAX} are the minimum and maximum values of the attribute set, respectively.

The selection of features plays a vital role in the effectiveness of an intrusion detection system (IDS). However, the presence of duplicated features poses a significant challenge in this regard. Such duplication is a major factor contributing to the high dimensionality of selected features, which can result in reduced detection accuracy and a higher rate of missing intrusions in a conventional IDS. To address this issue, the proposed system utilizes the deep belief network (DBN) technique, specifically employing a stacked restricted Boltzmann machine (RBM) to reduce the dimensionality of features.

The DBN technique is well-suited for feature selection in extensive wireless networks due to its unsupervised nature. This section provides an overview of the DBN and RBM approaches. The DBN is a common deep learning technique that allows for easy prediction and classification of samples transformed into a new feature space by selecting features in a layer-by-layer manner. It comprises multiple RBMs, where the RBM is a two-layer connectionist energy model with visible and hidden layer units represented by vectors m and n, respectively. In this context, the RBM consists of v visible units and h hidden units. The state of the d unit is denoted as nd, while the state of the c unit is denoted as mc. The energy for a

given state (M, N) is defined by Eq. (2), which captures the energy interaction within the RBM.

$$Energy(m,n|\mu) = -\sum_{c=1}^{v} a_c m_d - \sum_{d=1}^{h} b_c n_d - \sum_{c=1}^{v}\sum_{d=1}^{h} m_c w_{cd} n_d \quad (2)$$

Upon examining Eq. (2), it becomes evident that μ is defined as a function of {Wcd, ac, d}, where Wcd, ac, and bd represent the parameters of the RBM. Specifically, Wcd corresponds to the connection weights between the visible unit c and the hidden unit d. Additionally, ac signifies the bias of the visible unit, while bd represents the bias of a hidden unit. To further elaborate, the joint probability for both the visible and hidden layers can be expressed as Eq. (3), capturing the probabilistic relationship within the RBM.

$$P(m,n|\mu) = \frac{1}{Z(\mu)} \exp(-Energy(m,n|\mu)) \quad (3)$$

The normalization factor, denoted as Z(μ), is defined as the sum of the negative energy (*-Energy*) for all possible values of m and n, represented by Σm, n. This factor serves to normalize the probabilities within the RBM. Specifically, the observational data distribution for $P(m \mid \mu)$ is determined by the RBM. In turn, the joint probability $P(m, n \mid \mu)$ for the marginal distribution can be expressed through Eqs. (4) and (5), outlining the probabilistic relationships involved.

$$P(n|\mu) = \frac{1}{Z(\mu)} \sum_{m} \exp(-Energy(m,n|\mu)) \quad (4)$$

$$P(m|\mu) = \frac{1}{Z(\mu)} \sum_{n} \exp(-Energy(m,n|\mu)) \quad (5)$$

To optimize the RBM parameters and maximize the marginal distribution, the maximum likelihood function utilizes stochastic gradient descent. Given a sample N, the maximum likelihood function is expressed as Eq. (6). This function serves as a key component in estimating the parameters of the RBM model.

$$L(\mu) = \frac{1}{V} \sum_{v=1}^{V} \exp(-Energy(m,n|\mu)) \quad (6)$$

In accordance with Eq. (6), the term $P(m, n \mid \mu)$ represents the joint probability distribution between the hidden and visible units.

To reduce the dimensionality of the pre-processed feature set, we employed the DBN deep-learning technique. This process consists of two stages: the initial stage involves pre-training the RBM using a predefined group composed of visible elements denoted as m= mv, and a set of hidden elements denoted as n= nh, where 'h' denotes the hidden neural hubs. In this case, 'v' neural hubs are utilized at the visible level. The contrastive divergence (CD) method employing Gibb's random sampling technique is employed to initialize the neural states of each feature component in a random manner. The hubs are divided into stochastic portions, namely visible and hidden hubs, without any direct connections between the visible and hidden hubs, but only interlink associations.

The visible-level hubs contain some of the original values from the pre-processed feature sets, while the hidden-level hubs contain the processed values obtained through level-by-level training of the entire DBN network using Eqs. (2)–(6). The DBN network is constructed by stacking RBMs, with the initial RBM level utilizing pre-trained feature values as inputs for the higher layers of the RBM [71]. To ensure the proper unfolding of the RBM layers and minimize reconstruction errors, adjustments to the limits (W_{cd}, a_c, b_d) are introduced in the secondary stage of DBN training.

The DBN training procedure is depicted in Fig. 2. In this prototype, the Wi-Fi network data was organized into 154 feature components arranged on a single visible layer. The ascending projectiles represent the generative model, while the descending bullets represent the recognition model. To enhance the stacked RBMs, the class names of the information vectors were incorporated as an additional level at the topmost layer. Similar to the configuration of a backpropagation (BP) neural network, the complete DBN is modified at this stage. By employing the Wi-Fi attack identification mechanism, reduced feature sets for future use are obtained.

In stochastic gradient descent (SGD) models, particularly deep neural networks (DNNs), federated learning techniques are commonly utilized. However, DNNs are not well-suited for capturing changes in time series data. Recurrent neural networks (RNNs) are often employed for analyzing sequence data, including wireless network traffic data. On the other hand, DBN offers a simpler structure, fewer parameters, and a faster training rate compared to RNNs.

To get classification results and reduce cross-entropy loss, deep learning classification models frequently use the Softmax activation function in their final neural network layer. The probabilities of each category are output using the Softmax algorithm.

FEDDBN, a federated learning (FL) system, comprises two primary components: the data owner (participant) and the model owner (FL server). The FL server, typically located at the cloud center, solely receives local model parameters from edge devices, as depicted in Fig. 3, which are then aggregated to form a global model. Edge

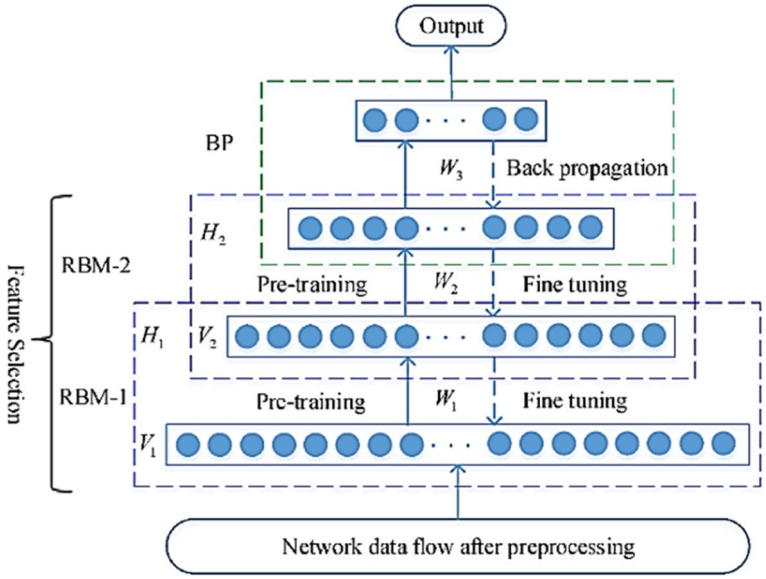

Fig. 2 DBN feature learning model with stacked RBM

devices train local intrusion detection models using private datasets.

In existing devices, the typical FL technique is used, where each device imports the central model and executes calculations using its local data. Each device thereafter returns a new iteration of the core model. These local models are combined by the server to provide a more accurate overall model. Distributed intrusion detection is another objective of the suggested federated structure. Generators and discriminators are the two basic categories involved. The initial parameters for the central models include weights, learning rate, batch size, penalty coefficient, and decay rate. After that, to obtain these parameters, a ping message is broadcast to all connected active devices. Transmission of the generator and discriminator's parameter sets starts the training phase. Following the first global epoch, local generators and discriminators' basic setup and design can be changed.

The effectiveness of our model's training encounters various obstacles. It necessitates well-organized associations between clients (devices) to establish connections between a generator and a discriminator. Additionally, ensuring a reasonable computational burden on the devices is crucial. By examining the probability distribution of the device's local traffic, each generator first produces synthetic data. In order to train its intrusion detection network, the discriminator uses input from the local device and generator. The gradients are sent to the core networks once the two deep neural networks have finished their local iterations. The central model, which is then sent to the end devices for use, is updated with the local model's parameters. The central generator and discriminator compile each local gradient of Gn and Dn. The local networks of these devices then receive the latest gradients once more.

The modified gradients are sent to local generators so they may create a new model and boost the caliber of the samples they produce. As seen in Figs. 4 and 5, the updated gradients from the central discriminator help to improve the local discriminators' detection accuracy.

Edge networks require interactions with a variety of edge nodes for federated learning to operate. Because wireless connections are sometimes unstable or inconsistent, there are fewer communication cycles between the client and server. Due to the comparatively quick processors on edge clients, the dataset for each client is less than the dataset. The learning framework's performance is hence constrained by communication delay, although computing costs are frequently insignificant in contrast. It is crucial to guarantee that the learning process is convergent while also reducing the overall number of communication cycles.

This paper suggests an improved federated learning approach called FEDDBN that uses the attention mechanism notion to evaluate the importance of uploaded model parameters. The importance criteria are based on optimizing the global model's classification performance to rate the relevance of various clients. When bandwidth resources are limited, clients with high relevance—identified by larger weights of their uploaded parameters—are given precedence over clients with lower relevance because they have a bigger influence on the performance

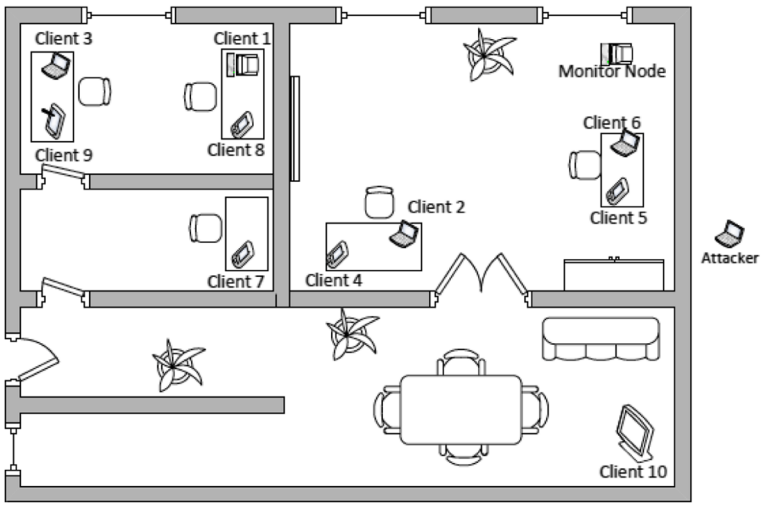

Fig. 3 Prototype of the AWID testbed model

of the model. The local model is no longer calculated, and only the global model parameters are permitted to update after the relevance of model parameters drops below a specific threshold and exceeds a specific number of repetitions. As a result, transmission and calculation costs are greatly decreased while ensuring model convergence. The FedAVG algorithm is improved by the FEDDBN algorithm, which is described in detail on this page:

In a manner like the previous federated learning technique, the main server initializes and issues model parameters on the server, waits for the client model parameters to be uploaded, and then executes aggregate updates. In order to take into account, the wireless edge network environment, we present an asynchronous updating technique for the server. Without awaiting the completion of other clients' calculations, the central server modifies the model parameters in response to a client update. This considers situations in which several clients finish their local calculations at the same time, resulting in many updates.

Attentional processes are crucial to feature representation because they help us determine the value of parameters. The attention mechanism can pick out the most important information from a large volume of input, which was first used for picture recognition.

The importance of uploaded model parameters is assessed by utilizing the attention mechanism and CMFL algorithm to improve the performance of the global model and give greater weights to client-submitted parameters that contribute more to the model. The attention mechanism on the main server is used by the FEDDBN to give various weights to various client model characteristics.

5 Experimental results and analysis

We assess the recall, precision, accuracy, F1-score, and detection rate of the FEDDBN-IDS model in this section. We also show the outcomes of categorizations that were accomplished using the suggested IDS model for binary and multiclass data. The performance of an FL-based IDS (FED-IDS) built without a DBN architecture but with the same hyperparameters is also compared. We keep the total number of communication rounds at the global level constant to achieve a fair comparison. We execute all essential preprocessing steps for model training on a typical AWID remote dataset before running our experiments.

Our studies show that the FEDDBN algorithm works better than the FedAvg approach when it comes to intrusion detection [72]. It works well to combine local models by averaging their weights on a central server, even for a basic neural network model. The overall number of features in the dataset (154) is equivalent to the number of neurons in the input layer and the output layer. The loss function used by these neurons is sparse Softmax cross-entropy, and they use sigmoid activation functions.

Each of our studies has 20 rounds overall, an 8-round round size, and a learning rate of 0.8. The results Wang [2] presented are consistent with the training and test accuracies.

In the performance analysis of intrusion detection, the following commonly used metrics are employed:

- True positive (TP): this metric quantifies the number of attack patterns correctly classified as attacks.
- False positive (FP): it measures the proportion of normal patterns mistakenly labeled as attacks.

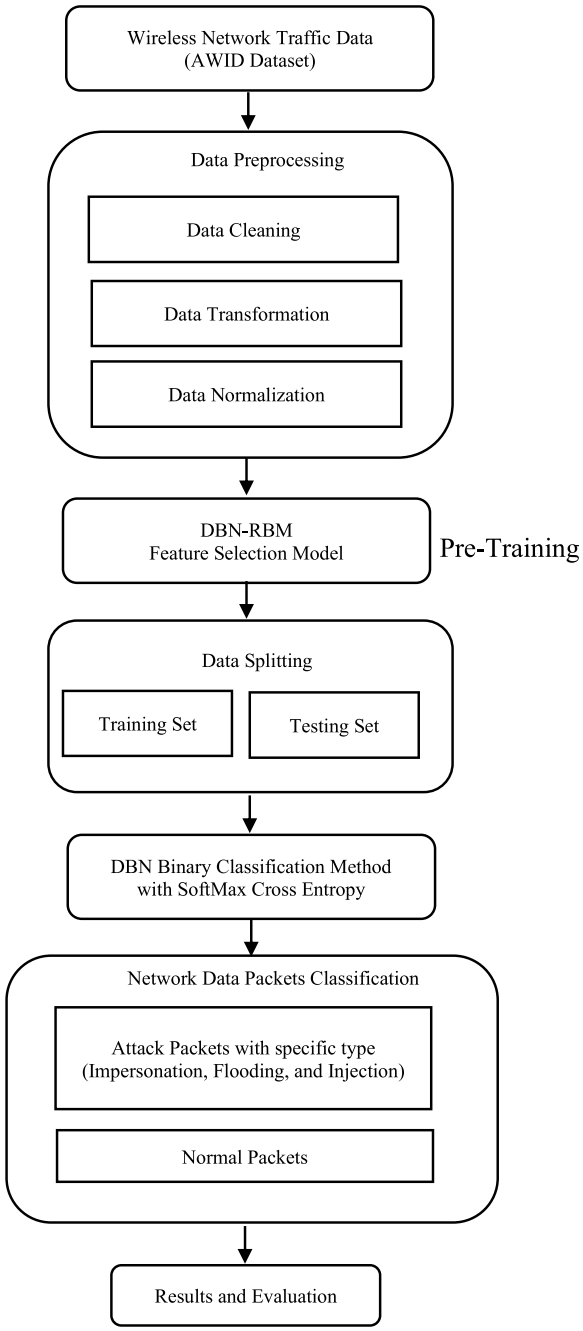

Fig. 4 Proposed scheme of the DBN method for training the local models

- True negative (TN): the TN metric calculates the proportion of normal patterns correctly categorized as such.
- False negative (FN): it counts the number of attack patterns incorrectly labeled as normal.

Additionally, accuracy is employed as a measure to assess the ratio of correctly classified entries to the total number of entries, as provided by Eq. (7).

$$\text{Accuracy} = \frac{TP + TN}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{7}$$

Precision is the ratio of correct intrusion classes to all anticipated intrusion outcomes and can be calculated as follows in (8):

$$\text{Precision} = \frac{TP}{\text{TP} + \text{FP}} \tag{8}$$

Recall that the proportion of accurate attack classifications compared to the total number of samples that should have been classified as attacks is shown by formula (9):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{9}$$

F1-Score: reports the harmonic mean between precision and recall, which is given by (10).

$$\text{F1} - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

The ratio of properly detected abnormal samples to all truly abnormal samples is known as the detection rate, and can be computed as follows in (11):

$$\text{Detection Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{11}$$

5.1 Experimental model of the testbed

To assess the outcomes of the suggested FEDDBN model, the suggested approach employs an authentic testbed-based AWID dataset, as seen in Fig. 3.

The following procedures are followed in creating the testbed for the dataset. A physical lab that faithfully replicates a standard SOHO architecture has been established for data collection reasons. A single mobile attacker was launching many assaults, as seen in Table 4, while a few mobile and stationary STAs served as the network's legitimate client devices. More precisely, there was one desktop computer, two laptops, two cell phones, one tablet, and one smart TV that were part of the legitimate network. Throughout every trial, the desktop computer and smart TV stayed in the same locations. Over the course of the trials, the smartphone devices demonstrated considerable mobility, as shown by their frequent joining, and leaving of the network and changing of positions inside the lab's facilities. Finally, the laptop computers

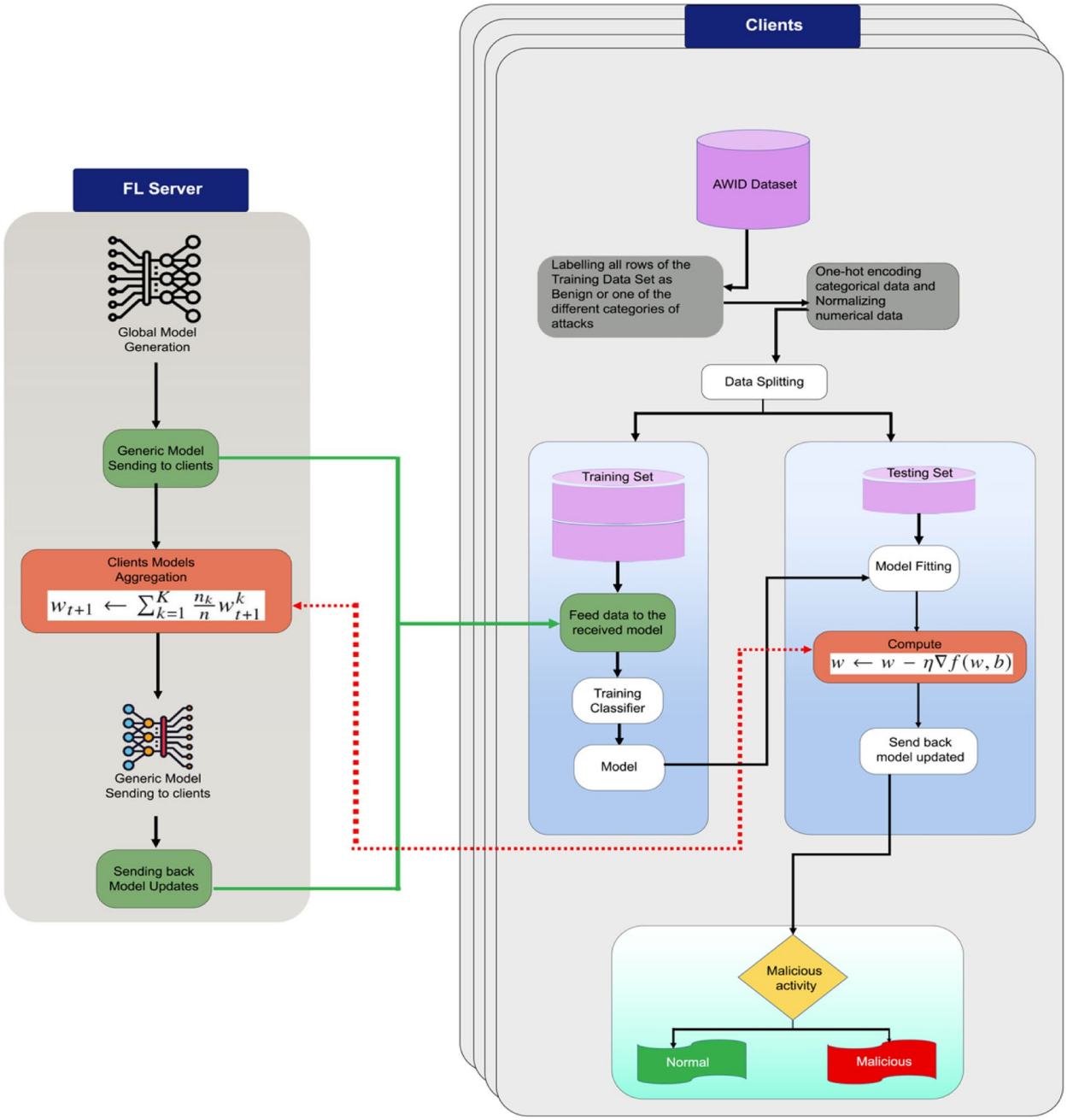

Fig. 5 Architecture of the proposed FEDDBN model-based WNIDS

were semi-static—that is, they hardly ever moved. Internet surfing, VoIP, and file downloads were the services that were operating on the clients and oversaw generating traffic. The network was secured by a single AP, a Netgear N150 WNR1000 v3 device (Firmware Version V1.0.2.5460.0.82), with a 54-Mbps transfer rate and poor WEP encryption.

With a single access point (AP) that supported up to 54 Mbps transfer rates and was secured by the unstable WEP encryption, the network was protected by a Netgear N150 WNR1000 v3 device (Firmware Version V1.0.2.5460.0.82). Using an Acer Aspire 5750G laptop running Kali Linux 1.0.6 64 bit, a single attacker node launched all the assaults. The attacker has a

Table 4 Description of the devices of the AWID testbed model

Node	Type	Brand	OS	Network Card	CPU
Client 1	Desktop	Custom	Ubuntu Linux 12.04 LTS	Netgear WNA3100 N300	Intel Core i7 3.2GHz
Client 2	Laptop	Fujitsu-Siemens	Ubuntu Linux 12.04 LTS	Intel 3945ABG	Intel Core Duo T2050 1.6GHz
Client 3	Laptop	Acer	Ubuntu Linux 12.04 LTS	Qualcom Atheros AR9462	Intel Core i5 1.7GHz
Client 4	Smartphone	iPhone 3G	iOS 4.2	NA	Samsung 32-bit RISC ARM 620MHz
Client 5	Other	iPod Touch	iOS 3.1	NA	Samsung 32-bit RISC ARM 533MHz
Client 6	Laptop	Acer Aspire 5750G	Windows 7	Broadcom BCM943227HM4L	Intel Core i5 2.8GHz
Client 7	Smartphone	HTC Diamond	Windows Phone 6.1	NA	528 MHz ARM 11
Client 8	Smartphone	Samsung Nexus	Android 4.2	NA	dual-core ARM Cortex-A9 1.2 GHz
Client 9	Tablet	Samsung Galaxy Tab	Android 2.2	Na	Cortex-A8 1 Ghz
Client 10	Smart TV	LG 42LM7600S	Linux	NA	NA
Attacker	Laptop	Acer Aspire 5750G	Kali Linux 1.0.6	D-Link DWA-125/Linksys WUS-B54GC	Intel Core i5 2.8GHz
Monitor Node	Desktop	Custom	Linux Debian 7.3	Alpha AWUS036H	Core i7 2.4Ghz

D-Link DWA-125 card that could be used in promiscuous mode to inject packets. The attacker switched her MAC address a lot while carrying out several assaults. The Aircrack-ng suite, the MDK3 tool, the Metasploit framework, and bespoke ones created in C using the Lorcon2 library were among the tools used to implement the assaults. The mobile invader was operating outside the boundaries of the laboratory's premises. A different gadget was added as a monitor node to record the wireless communication. Although it was situated within the network's service area, this node was never connected to it.

Running Linux Debian 7.3, the monitor node was a desktop computer with an Alpha AWUS036H card in promiscuous mode and a Samsung 840 series SSD hard drive with a writing speed of 130 MB/s. On that node, the Tshark program—the terminal version of Wireshark—was installed. It was used to record traffic in many smaller-sized PCAP files, each of which included information collected over the course of an hour. Keep in mind that not every packet will be collected using this method. However, we stress the method's affordability and ease of use, contending that it is the most dependable way to collect data in resource-constrained contexts, like SOHO ones. A technique of normalization for certain fields (e.g., MAC addresses represented as integers) was applied to the generated CSV files in an interim phase that followed the monitoring phase.

5.2 Operational scenarios on the AWID testbed dataset

We develop three scenarios using the AWID testbed dataset to assess how various data distributions affect our multiclass classifier's ability to identify assaults. We quantify the imbalance of the various local datasets of each FL client using Shannon entropy [70]. The following formula determines the balance between the classes given a dataset of length n and k classes of size c_i:

$$\text{Entropy} = \frac{-\sum_{i=1}^{k} \frac{c_i}{n} \log \frac{c_i}{n}}{\log k} \tag{12}$$

where all classes are equal to 0 except for one, and all c_i=n/k, in which case the function equals 1. It is also important to remember that we see every FL client as having a single IP address. Here, d is the attack class count, w is the number of network flows, and h is the attack class size.

5.2.1 Basic scenario

The wireless network traffic in this case serves as the basis for each FL client's dataset. In this instance, there is a significant imbalance in the distribution of samples and classes across the various nodes. In fact, there are only two instances of impersonation attacks on devices 1 and 3, compared to seven benign traffic samples. This hypothetical circumstance exemplifies a common occurrence in a particular wireless network, whereby some devices may fall prey to many assaults, while other devices carry out their intended functions unaffected by attacks. As said, however, the simple use of FL in this case can lead to poor convergence and performance problems. In this instance, there is a significant imbalance in the distribution of classes and samples across the various nodes, as Table 5 illustrates.

5.2.2 Balance scenario

Here, we distribute a subset of our dataset across the ten devices, ensuring that every device receives an equal number of samples from each class. As a result, each party has the same Shannon entropy value, as seen in Table 5.

Table 5 Description of the basic, balanced, and mixed scenarios

Scenario	Device	Entropy
Basic	1	0.520
	2	0
	3	0.286
	4	0
	5	0.388
	6	0.472
	7	0.119
	8	0
	9	0.087
	10	0.085
Balanced	1–10	0.761
Mixed	1	0.662
	2	0.664
	4	0.668
	6	0.698
	8	0.702

As said, a balanced situation like this offers superior performance; nevertheless, depending on the scenario under consideration, each FL client may have samples of other nodes, which might lead to privacy concerns.

5.2.3 Mixed scenario

To strike a compromise between the two earlier options, a mixed scenario is created where each side keeps its own samples, but they are balanced locally. The devices that have a Shannon entropy value, as determined by (12), greater than a predetermined threshold (0.2) are the ones that we choose: parties 1, 2, 4, 6, and 8. Since the classes of the devices are not evenly distributed, we apply a basic instance selection technique to eliminate some of the samples from the prominent classes until we obtain the Shannon entropy within a range of values after this first filtering step. With this set falling between 0.66 and 0.71, we are left with a dataset that is a reasonable middle ground between the default situation, in which balancing was not applied, and the balanced scenario, in which the dataset was arbitrarily split across the ten devices.

Our studies have been conducted utilizing IBM-based federated learning (IBMFL), which uses a federated architecture for learning, in a distributed and virtual testbed. It is configured with a central server and ten client devices, or parties, each of which runs a separate FL process. The simulation divides the learning into many discrete processes or threads, each of which does the federated learning job in parallel, even if all the federated learning processes are carried out on one physical system. For our investigation, an HP laptop equipped with a 3 GHz Intel(R) Core (TM) i7-10750H CPU and a 4-GB NVIDIA GeForce RTX2060 graphics card was used to replicate the federated environment, parties, and server. After every round of global communication, the central edge server receives all the local model parameters and compiles and updates them.

In terms of the percentage of correctly detected cases for each class label in the AWID dataset, the proposed FEDDBN model performed better than the existing FedAVG technique as shown in Fig. 6.

The performance of the FEDDBN model was verified. Table 6 summarizes the findings for binary classification using the FEDDBN-IDS and AWID datasets. Additionally, it illustrates how FED-IDS (adapted) and FEDDBN-IDS (proposed) compare across all the measures. According to the table, the FEDDBN-IDS binary

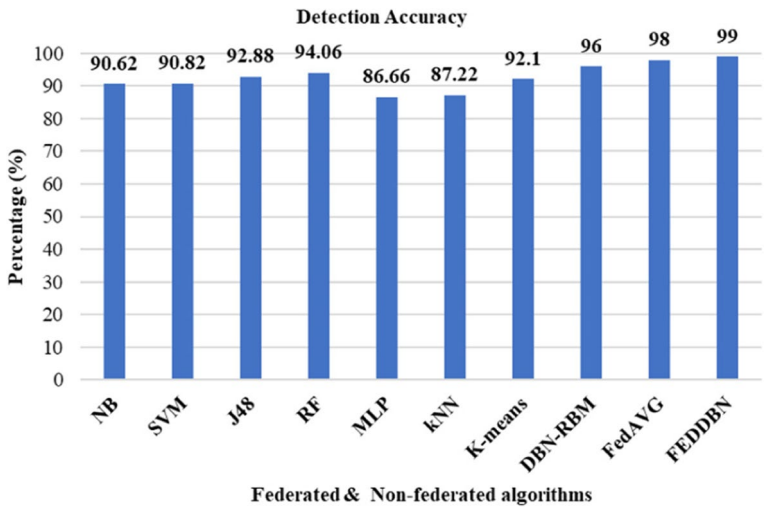

Fig. 6 Comparative analysis of federated vs. non-federated algorithms using detection accuracy

Table 6 Comparison of binary classification of the AWID dataset using FED-IDS and FEDDBN-IDS

Metrics	AWID	
	FED-IDS	FEDDBN-IDS
Accuracy	96.1	99.2
Precision	95.8	99.3
Recall	95.1	99
F1-Score	96	98.9
Detection rate	96.3	99.5

classification model performed 98% to 99.5% correctly when utilizing the AWID dataset. Table 7 displays the outcomes of the final global upgrades. The FEDDBN-IDS multiclass classification model's performance for various metrics using the AWID dataset ranged between 98.5% and 99%.

In our research, we adjusted parameter *D*, or the number of wireless network devices, during each training session. For instance, D =1 denotes that all wireless network devices were utilized for training, D = 0:5 denotes that only half of the devices were used, and D = 0 denotes that an algorithm round employed a random number of devices. Every linked device was utilized for every training cycle in the initial experiment, with D = 1. By calculating the detection accuracy rate, Fig. 7 illustrates the performance of the FEDDBN-IDS model. The metrics were averaged over the 20 runs. As seen from the data in the figure, the detection accuracy rate increased as the number of wireless devices increased.

The values of *D* are then modified to 0.5, where training uses half of the wireless network device group, and to 0, where a random number of device groups is chosen for each round, with a minimum of one device. The results in Fig. 8 show that even when only half of a group's devices are used in a round, detection accuracy is not hampered. According to this research, it is possible to train a model with comparable performance using fewer training devices. Therefore, we can use half as many devices for each training session and spend less on

Table 7 Comparison of multiclass classification of the AWID dataset using FED-IDS and FEDDBN-IDS

Metrics	AWID	
	FED-IDS	FEDDBN-IDS
Accuracy	96	99.1
Precision	96.2	98.7
Recall	95.5	99.3
F1-Score	96.1	99
Detection Rate	95.8	99.2

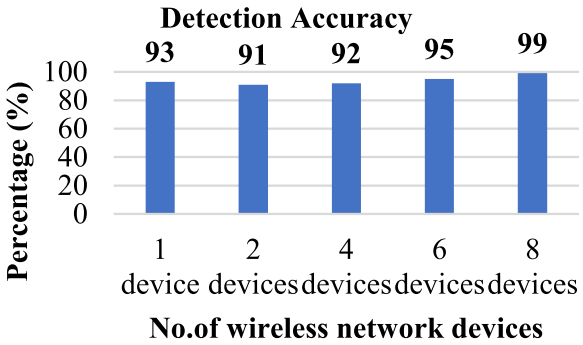

Fig. 7 Performance evaluation of FEDDBN-IDS with D = 1

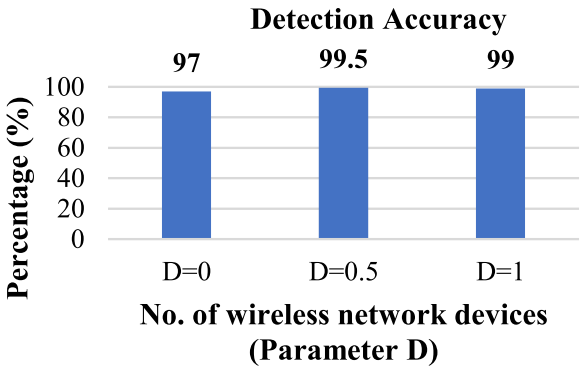

Fig. 8 Eight devices regrouped together, with the number of devices utilized in each training cycle changing (parameter *D*)

both connectivity and power. Figures 9 and 10 show the performance of the clients and server for CPU usage and memory consumption.

As shown in Fig. 11, when four or more devices were utilized in the training phase of the federated algorithm with D = 0, the accuracy of the model increased for all attacks.

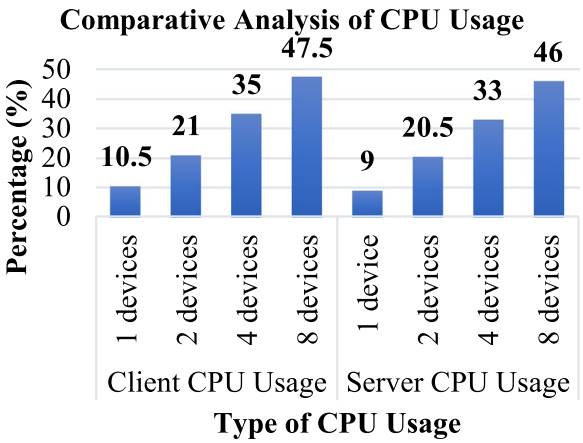

Fig. 9 Performance evaluation of CPU usage by client and server devices of FEDDBN-IDS model with D = 1

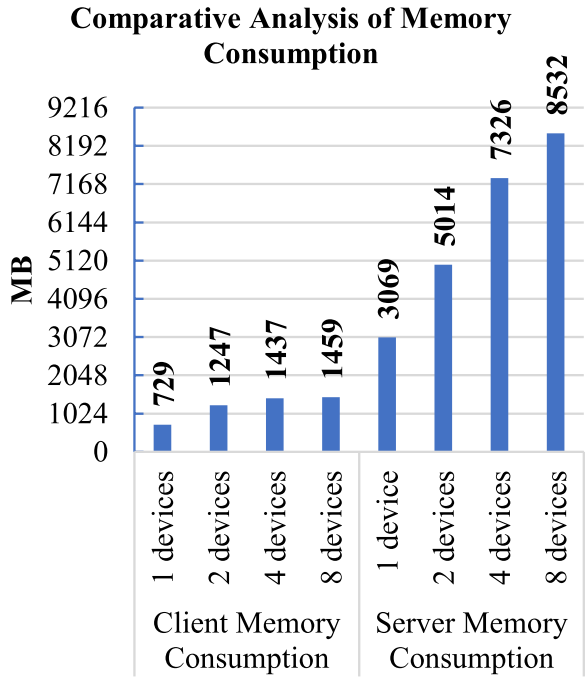

Fig. 10 Performance evaluation of memory consumption by client and server devices of FEDDBN-IDS Model with $D = 1$

The average detection accuracy increased between one and eight devices. Our efforts had a statistically significant and favorable influence on the rate of attack detection accuracy across the board. The detection accuracy rate was lower when the two devices were employed. This is because the federated algorithm achieves an equilibrium point where each wireless network device starts with the federated model, calculates the update, and receives the identical federated model back from the server. Because of the greater variety of devices, this scenario is avoided when more devices are registered in a cluster.

The amount of data sent to and received from the wireless network devices serves as a proxy for the communication cost of the federated algorithm. We sent 1621 bytes per model exchange using the proposed FEDDBN.

During each training cycle, a contributing device trains its model, changes its weights, and sends these modifications to the server. The server subsequently receives the weights that have been averaged, resulting in a total of two communications, each round of 1621 and 3242 bytes in size. In every round, a total of 3251 bytes are transmitted, and each patient receives nine bytes of command data. With eight rounds employed, our FEDDBN communication cost was 26,008 bytes. 98,034 bytes is the usual size of data saved on a device. We can reduce the cost of communication by 3.8 and save 72,026 bytes during transmission by using the proposed method.

The energy consumption analysis of the federated algorithm involved measuring the USB e-meter multiplied by the duration of each algorithmic step. During the training process, the overall energy consumed by the federated model was 265 J (joules), while the algorithm-free device consumed 218 J. Consequently, training the federated model required an additional 46 J, adding a new device to the network consumed 5.5 J, and testing new samples required 5.24 mJ. Thus, our FEDDBN model consumed 17.5% more energy during training and 3.4% more energy during prediction compared to the non-federated approach.

Next, we conducted a performance comparison between our proposed federated FEDDBN algorithm and various non-federated learning algorithms, including support vector machine (SVM), naive Bayes (NB), J48, multilayer perceptron (MLP), random forest (RF), k-nearest neighbor (kNN), K-means, and DBN-RBM, as depicted in Fig. 6. Figure 6 also showcases the training times of federated and non-federated models trained with eight device

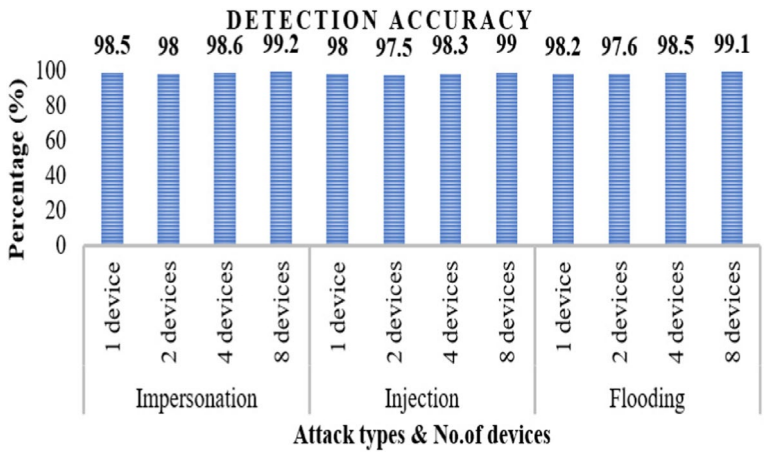

Fig. 11 Attack-specific detection accuracy of FEDDBN-IDS with $D = 0$

records. The results indicate that classifiers such as KNN, NB, K-means, and DBN-RBM exhibit greater time efficiency compared to SVM, RF, J48, MLP, and FEDDBN algorithms. However, in terms of training times, both the deep learning-based DBN-RBM and the federated learning-based FEDDBN models perform well when compared to the conventional MLP neural network technique.

Furthermore, for the AWID dataset, Fig. 8 presents a comparison of detection accuracy for various Wi-Fi attacks using federated learning and other learning techniques. We observed that among the non-federated learning algorithms, RF and DBN-RBM achieved the highest detection accuracy rates for the system trained on data from eight devices, with an average detection accuracy. In contrast, the federated learning model FEDDBN exhibited a higher detection accuracy rate of 99% compared to RF and DBN-RBM models. Due to its superior training efficiency compared to non-federated algorithms, FEDDBN offers the best detection accuracy.

Finally, the scalability of the proposed system was assessed using the same architecture and data. Instead of tracking the amount of time spent on the server, we tracked the amount of time spent by clients in each training round. All the models reached convergence at approximately 6.5 s. Round 8 was reached by the two-device model in 6.79 s, whereas it took the 64-device model the same amount of time. Generally, increasing the number of devices results in a linear increase in the amount of time spent at the server but not at the expense of lengthening training sessions or individual rounds.

6 Conclusion and future work

Using the substantial AWID dataset, which has previously been made accessible for federated learning experiments, we demonstrate how to construct the federated learning framework, FEDDBN, to identify intrusions in wireless networks. With a decreased communication overhead, the FEDDBN algorithm achieved a detection accuracy of 99%. We discovered that FEDDBN centralized learning outperformed the conventional DBN model in terms of classification performance by 3%. The detection accuracy of the FEDDBN algorithm was approximately 1% higher than that of FedAVG as shown in Fig. 6, and the method required up to 70% fewer communication rounds than that of FedAVG.

The experimental results also show that utilizing an RBM-based DBN to train an FL-based IDS on heterogeneous data may significantly reduce FEDDBN model performance by as much as 50%. There is some evidence to suggest that pre-training both the local and global models may somewhat reduce data heterogeneity. The suggested DBN-RBM model also addresses the drawbacks of various evasion strategies by dynamically learning the feature set for each occurrence of the Wi-Fi network packet with improved and optimized hyperparameters. According to the operational cybersecurity perspective, the suggested FEDDBN model uses an FL model to assure data privacy and a dynamically pre-trained DBN based RBM to protect wireless network devices, systems, and users' data from adversarial assaults. Soon, a non-ML cyber security researcher may use the decentralized hardware IDS design of the wireless network components to build the suggested FL model.

The following are the future directions of research that must be considered under the proposed FEDDBN model: (i) to solve the issue of resource heterogeneity. In other words, the primary difficulty is in connecting every device inside the diverse wireless networks because these heterogeneous devices operate on different platforms and frameworks. This includes the Internet of Things (IoT) devices that are connected via wireless networks, the large number of different devices, network complexity, heterogeneity at the device and network levels, multiple communication protocols, and the enormous volume of actions that these sensors naturally produce. All these factors make the development of IDS extremely difficult, (ii) to utilize datasets encompassing more attack types by pre-training the datasets to detect different attack types using DL-based models like SAE, RNN, etc, than the proposed DBN-RBM model, and (iii) to model self-security new-generation FL-IDS to prevent model poisoning and malicious parameter update packets. For instance, incorporating blockchain technology can increase a task's transparency and traceability [73] and the use case of the same has started its implementation using a permissioned blockchain-based federated learning method where incremental updates to an anomaly detection machine learning model are chained together on the distributed ledger [56, 60].

Abbreviations

IDS	Intrusion detection system
DBN	Deep belief networks
FDL	Federated deep learning
FL	Federated learning
DL	Deep learning
RBM	Restricted Boltzmann machine
ML	Machine learning
WNIDS	Wireless network intrusion detection system
AI	Artificial intelligence
DNN	Deep neural networks
SVM	Support vector machine
ANN	Artificial neural networks
AWID	Aegean Wi-Fi intrusion dataset
kNN	k-nearest neighbours
RF	Random Forest
DT	Decision tree
NB	Naïve Bayes
IoT	Internet of Things

AE	Autoencoders
RNN	Recurrent neural networks
GBDT	Gradient boosting decision tree
GRU	Gated recurrent unit
FFT	Fast Fourier Transformation
FedAVG	Federated averaging
SGD	Stochastic gradient descent
MLP	Multilayer perceptron
LSTM	Long short-term memory
TFF	Tensor flow federated
CPS	Cyber physical systems
WEN	Wireless edge networks
SAE	Stacked autoencoders
TP	True positive
FP	False positive
TN	True negative
FN	False negative

Authors' contributions
Nivaashini M has contributed in the design, implementation, and experimental result analysis of the proposed work. Suganya E and Prabu M have contributed in the survey of the existing research work using the AWID dataset. Sountharrajan S and Durga Prasad Bavirisetti have contributed in the technical writing and finetuning of the manuscript. Each author had participated sufficiently in the work to take public responsibility for appropriate portion of the content. All authors read and approved the final manuscript.

Funding
Open access funding provided by Norwegian University of Science and Technology

Availability of data and materials
The datasets analyzed during the current study are available in the AWID - Awid-wireless security datasets project data set repository, https://icsdweb.aegean.gr/awid/.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 9 July 2023 Accepted: 20 March 2024
Published online: 04 April 2024

References

1. C. Kolias, G. Kambourakis, A. Stavrou, S. Gritzalis, Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset. IEEE Commun. Surv. Tutor. **18**(1), 184–208 (2016)
2. S. Wang, B. Li, M. Yang, and Z. Yan, "Intrusion detection for Wi-Fi network: A deep learning approach," in Wireless Internet. Springer International Publishing, 2019, pp. 95–104.
3. H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Ag¨uera y Arcas, "Communication-efficient learning of deep networks from decentralized data," Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, vol. 54, 2017.
4. J. Yang, T. Li, G. Liang, W. He, Y. Zhao, A simple recurrent unit model-based intrusion detection system with dcgan. IEEE Access **7**, 83286–83296 (2019)
5. R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, A. Abuzneid, Features dimensionality reduction approaches for machine learning-based network intrusion detection. Electronics **8**(3), 322 (2019)
6. B. Liao, Y. Ali, S. Nazir, L. He, H.U. Khan, Security analysis of IoT devices by using mobile computing: a systematic literature review. IEEE Access **8**, 120331–120350 (2020)
7. M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the KDD CUP 99 dataset," in Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6, Ottawa, ON, Canada, July 2009.
8. N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive dataset for network intrusion detection systems (UNSW-NB15 network dataset)," in Proceedings of the 2015 Military Communications and Information Systems Conference (Mil-CIS), pp. 1–6, IEEE, Canberra, ACT, Australia, 2015 November.
9. N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: bot-IoT dataset. Future Generation Comput. Syst. **100**, 779–796 (2019)
10. M.F. Elrawy, A.I. Awad, H.F.A. Hamed, Intrusion detection systems for IoT-based smart environments: a survey. J. Cloud Comput. **7**(1), 21 (2018)
11. A.A. Hamad, G.N. Nguyen, D.-N. Le, Efficient dual cooperative bait detection scheme for collaborative attackers on mobile ad hoc networks. IEEE Access **8**, 227962–227969 (2020)
12. M.K. Shahoodh, The adjacency matrix of the compatible action graph for finite cyclic groups of p-power order. Tikrit J. Pure Sci. **26**(1), 123–127 (2021)
13. F.J. Suhae, A.I. Hussain, Suitability evaluation of mudstone of Injana formation for dam filling materials in Taq Taq area/Erbil/Iraq. Tikrit J.Pure Sci. **25**(3), 49–56 (2020)
14. M.L. Thivagar, A.A. Hamad, A theoretical implementation for a proposed hypercomplex chaotic system. J. Intell. Fuzzy Syst. **38**(3), 2585–2590 (2020)
15. S.R. Thanoon, A comparison between Bayes estimation and the estimation of the minimal unbiased quadratic standard of the bidivision variance analysis model in the presence of interaction. Tikrit J. Pure Sci. **25**(2), 116–123 (2020)
16. S.A. Wuhaib, N.F. Abd, Control of prey disease in stage structure model. Tikrit J. Pure Sci. **25**(2), 129–135 (2020)
17. M.A. Ferrag, O. Friha, L. Maglaras, H. Janicke, L. Shu, Federated Deep Learning for Cyber Security in the Internet of Things: Concepts, Applications, and Experimental Analysis. IEEE Access **9**, 138509–138542 (2021). https://doi.org/10.1109/ACCESS.2021.3118642
18. A.R. Khan, Facial emotion recognition using conventional machine learning and deep learning methods: current achievements, analysis and remaining challenges. Information **13**(6), 268 (2022)
19. S. Sengan, R. H. Jhaveri, V. Varadarajan, R. Setiawan, and L. Ravi, "A secure recommendation system for providing context-aware physical activity classification for users," Security and Communication Networks, vol. 2021, 2021.
20. R. H. Jhaveri, N. M. Patel, Y. Zhong, and A. K. Sangaiah, "Sensitivity analysis of an attack-pattern discovery based trusted routing scheme for mobile ad hoc networks in industrial IoT," in Proceedings of the IEEE Access, vol. 6, pp. 20085–20103, IEEE, April 2018.
21. T. Saba, K. Haseeb, I. Ahmed, A. Rehman, Secure and energy-efficient framework using Internet of Medical Things or e-healthcare. J. Infect. Public Health **13**(10), 1567–1575 (2020)
22. F.A. Khan, A. Gumaei, A. Derhab, A. Hussain, TSDL: a two-stage deep learning model for efficient network intrusion detection. IEEE Access **7**, 30373–30385 (2019)
23. A. Andalib and V. T. Vakili, "An autonomous intrusion detection system using an ensemble of advanced learners," in Proceedings of the 2020 28th Iranian Conference on Electrical Engineering (ICEE), pp. 1–5, IEEE, Tabriz, Iran, 2020.
24. S. Agrawal, J. Agrawal, Survey on anomaly detection using data mining techniques. Proc. Comput. Sci. **60**, 708–713 (2015)
25. M. Ahmed, A. Naser Mahmood, J. Hu, A survey of network anomaly detection techniques. J. Netw. Comput. Appl. **60**, 19–31 (2016)
26. J. Ran, Y. Ji, and B. Tang, "A Semi-Supervised learning approach to IEEE 802.11 network anomaly detection," in 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Apr. 2019, pp. 1– 5.
27. V. L. L. Thing, "IEEE 802.11 network anomaly detection and attack classification: A deep learning approach," in 2017 IEEE Wireless Communications and Networking Conference (WCNC). ieeexplore.ieee.org, Mar. 2017, pp. 1–6.
28. I. Abunadi, A.A. Albraikan, J.S. Alzahrani et al., An automated glow worm swarm optimization with an inception based deep convolutional neural network for COVID-19 diagnosis and classification. Healthcare **10**(4), 697 (2022)
29. A. R. Gupta and J. Agrawal, "A comprehensive survey on various machine learning methods used for intrusion detection system," in Proceedings of

the 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), pp. 282–289, IEEE, Gwalior, India, 2020.
30. K. Farhana, M. Rahman, M.T. Ahmed, An intrusion detection system for packet and flow-based networks using deep neural network approach. Int. J. Electr. Comput. Eng. **10**(5), 5514 (2020)
31. W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, ``Federated learning in mobile edge networks: A comprehensive survey,'' IEEE Commun. Surveys Tuts., vol. 22, no. 3, pp. 2031_2063, 3rd Quart., 2020.
32. P. Pu Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency, ``Think locally, act globally: Federated learning with local and global representations,'' 2020, arXiv:2001.01523. [Online]. Available: http://arxiv.org/abs/2001.01523
33. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, ``Communication-efficient learning of deep networks from decentralized data,'' in Proc. Artif. Intell. Statist., 2017, pp. 1273_1282.
34. T. Li, A. Kumar Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, ``Federated optimization in heterogeneous networks,'' 2018, arXiv:1812.06127. [Online]. Available: http://arxiv.org/abs/1812.06127
35. L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, ``LoAdaBoost: Loss-based AdaBoost federated machine learning with reduced computational complexity on IID and non-IID intensive care data,'' 2018, arXiv:1811.12629. [Online]. Available: http://arxiv.org/abs/1811.12629
36. C. He, S. Li, J. Therefore, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, X. Zhu, J. Wang, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "Fedml: A research library and benchmark for federated machine learning," 2020.
37. Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, Federated Learning, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2019. [Online]. Available: https://doi.org/10.2200/S00960ED2V01Y201910AIM043
38. Y. Ma, D. Yu, T. Wu, and H. Wang, "Paddlepaddle: An open-source deep learning platform from industrial practice," Frontiers of Data and Domputing, vol. 1, no. 1, p. 105, 2019. [Online]. Available: http://www.jfdc.cnic.cn/EN/abstract/article 2.shtml
39. S. Caldas, P. Wu, T. Li, J. Kone˘cn´y, H. Brendan McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," 2018.
40. J. Schneible and A. Lu, "Anomaly detection on the edge," in MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), Oct. 2017, pp. 678–682.
41. S. Xu, Y. Qian, and R. Q. Hu, "Data-driven edge intelligence for robust network anomaly detection," IEEE Transactions on Network Science and Engineering, pp. 1–1, 2019.
42. S. A. Marhon, C. J. F. Cameron, and S. C. Kremer, Recurrent Neural Networks. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 29–65. [Online]. Available: https://doi.org/10.1007/978-3-642-36657-4 2
43. F. Murtagh, "Multilayer perceptrons for classification and regression," Neurocomputing, vol. 2, no. 5, pp. 183–197, 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0925231291900235
44. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735
45. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modelling," 2014.
46. D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2021.
47. Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, M.S. Hossain, 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 (2021)
48. R. Zhao, Y. Yin, Y. Shi, and Z. Xue, "Intelligent intrusion detection based on federated learning aided long short-term memory," Physical Communication, vol. 42, p. 101157, 2020. [Online]. Available: https://doi.org/10.1016/j.phycom.2020.101157
49. K.Y. Lin, W.R. Huang, Using Federated Learning on Malware Classification. International Conference on Advanced Communication Technology, ICACT **2020**, 585–589 (2020)
50. B. Li, Y. Wu, J. Song, R. Lu, T. Li, L. Zhao, DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber-Physical Systems. IEEE Transactions on Industrial Informatics **17**(8), 5615–5624 (2021)
51. P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," vol. 5, 05 1999, pp. 223–238.
52. V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated Learning-based Anomaly Detection for IoT Security Attacks," IEEE Internet of Things Journal, vol. 4662, no. c, pp. 1–10, 2021.
53. I. Fraz˜ao, P. Henriques Abreu, T. Cruz, H. Araujo, and P. Simoes, "Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process:" 12 2018, pp. 230–235.
54. Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, "Intrusion Detection for Wireless Edge Networks Based on Federated Learning," IEEE Access, vol. 8, pp. 217 463–217 472, 2020.
55. Y. Qin and M. Kondo, "Federated Learning-Based Network Intrusion Detection with a Feature Selection Approach," in 3rd International Conference on Electrical, Communication and Computer Engineering, ICECCE 2021, 2021.
56. D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," Applied Sciences (Switzer-land), vol. 8, no. 12, pp. 1–21, 2018.
57. A. Cholakoska, B. Pfitzner, H. Gjoreski, V. Rakovic, B. Arnrich, and M. Kalendar, "Differentially Private Federated Learningfor Anomaly Detection in eHealth Networks," no. Ml, pp. 514–518, 2021.
58. B. Cetin, A. Lazar, J. Kim, A. Sim, and K. Wu, "Federated Wireless Network Intrusion Detection," Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019, pp. 6004–6006, 2019.
59. P. Tian, Z. Chen, W. Yu, and W. Liao, "Towards asynchronous federated learning-based threat detection: A DC-Adam approach," Computers and Security, vol. 108, p. 102344, 2021. [Online]. Available: https://doi.org/10.1016/j.cose.2021.102344
60. D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, E. Ilie-Zudor, "Chained anomaly detection models for federated learning: an intrusion detection case study", NATO. Adv. Sci. Inst. Ser. E Appl. Sci. **8**(12), 2663 (2018)
61. C. Cachin and M. Vukolic, "Blockchain consensus protocols in the wild," in Proc. 31st Intl. Symposium on Distributed Computing (DISC 2017), 2017.
62. B. Weinger, J. Kim, A. Sim, M. Nakashima, N. Moustafa, and K. J. Wu, "Enhancing IoT anomaly detection performance for federated learning," Proceedings - 2020 16th International Conference on Mobility, Sensing and Networking, MSN 2020, pp. 206–213, 2020.
63. N. A. Al-Athba Al-Marri, B. S. Ciftler, and M. M. Abdallah, "Federated Mimic Learning for Privacy Preserving Intrusion Detection," 2020 IEEE International Black Sea Conference on Communications and Networking, Black SEACOM 2020, 2020.
64. Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multitask network anomaly detection using federated learning," Pervasive Health: Pervasive Computing Technologies for Healthcare, pp. 273–279, 2019.
65. Y. Ali, H. Ullah Khan, GTM approach towards engineering a features-oriented evaluation framework for secure authentication in IIoT environment. Computers & Industrial Engineering **168**, 108119 (2022)
66. H.-Y. Chen, C.-H. Tu, Z. Li, H. W. Shen, and W.-L. Chao, "On the importance and applicability of pre-training for federated learning," in International Conference on Learning Representations, 2023.
67. J. Nguyen, K. Malik, M. Sanjabi, M. Rabbat, *Where to begin? exploring the impact of pre-training and initialization in federated learning* (2022)
68. Anish Nath. Packet Analysis with Wireshark. Packt Publishing Ltd, 2015.
69. AWID, "Awid-wireless security datasets project dataset," 2015. [Online]. Available: http://icsdweb.aegean.gr/awid/features.html
70. J.A. Bonachela, H. Hinrichsen, M.A. Munoz, Entropy estimates of small datasets. J. Phys. A **41**(20), 202001 (2008)
71. Hinton GE, Simon O, Whye TY. A fast learning algorithm for deep belief nets. Neural Comput **18**, 1527–1554 (2006)
72. B. Cetin, A. Lazar, J. Kim, A. Sim and K. Wu, "Federated Wireless Network Intrusion Detection," 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 6004-6006, https://doi.org/10.1109/BigData47090.2019.9005507.
73. H. B. Desai, M. S. Ozdayi, and M. Kantarcioglu, "Blockfla: Accountable federated learning via hybrid blockchain architecture," 2020.

Publisher's Note