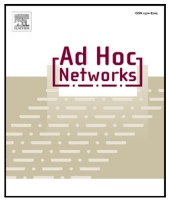
Ad Hoc Networks 162 (2024) 103540

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc



A federated learning-based zero trust intrusion detection system for Internet of Things

Danish Javeed [a], Muhammad Shahid Saeed [b], Muhammad Adil [c], Prabhat Kumar [d,*], Alireza Jolfaei [e]

[a] Software College, Northeastern University, Shenyang 110169, China
[b] School of Software Technology, Dalian University of Technology (DUT), Dalian 116024, Liaoning, China
[c] College of Intelligence and Computing, Tianjin University, Tianjin, China
[d] Department of Software Engineering, LUT School of Engineering Science, LUT University, 53850 Lappeenranta, Finland
[e] College of Science and Engineering, Flinders University, Adelaide, Australia

ARTICLE INFO

Keywords:
Cyber threats
Federated learning
Internet of Things
Intrusion Detection System

ABSTRACT

The rapid expansion of Internet of Things (IoT) devices presents unique challenges in ensuring the security and privacy of interconnected systems. As cyberattacks become more frequent, developing an effective and scalable Intrusion Detection System (IDS) based on Federated Learning (FL) for IoT becomes increasingly complex. Current methodologies struggle to balance spatial and temporal feature extraction, especially when dealing with dynamic and evolving cyber threats. The lack of diversity in datasets used for FL-based IDS evaluations further impedes progress. There is also a noticeable tradeoff between performance and scalability, particularly as the number of edge devices in communication increases. To address these challenges, this article introduces a horizontal FL model that combines Convolutional Neural Networks (CNN) and Bidirectional Long-Term Short Memory (BiLSTM) for effective intrusion detection. This hybrid approach aims to overcome the limitations of existing methods and enhance the effectiveness of intrusion detection in the context of FL for IoT. Specifically, CNN is used for spatial feature extraction, enabling the model to identify local patterns indicative of potential intrusions, while the BiLSTM component captures temporal dependencies and learns sequential patterns within the data. The proposed IDS follows a zero-trust model by keeping the data on local edge devices and sharing only the learned weights with the centralized FL server. The FL server then aggregates updates from various sources to optimize the accuracy of the global learning model. Experimental results using CICIDS2017 and Edge-IIoTset demonstrate the effectiveness of the proposed approach over centralized and federated deep learning-based IDS.

1. Introduction

The Internet of Things (IoT) has revolutionized digital connectivity by introducing smart communication methods, resulting in more efficient, reliable, and dynamic network communication. It facilitates intelligent networking operations among physical objects by using sensors and a communication protocol [1]. Sensors create a connectivity infrastructure by linking the participants, and the communication protocol regulates data flows across the network [2]. The scope of IoT is vast and can be observed in significant use cases such as smart grids [3], smart industries [4], smart surveillance systems [5], smart transportation, smart agriculture, smart healthcare, and more [6].While the extended reach of IoT offers many advantages, it also exposes it to various security challenges, including networking attacks, routing vulnerabilities, spoofing, sniffing, data breaches, and even network collapse in severe cases [7]. In this context, the concept of zero trust becomes crucial. Cyber exposure in IoT creates opportunities for various adversarial pursuits, such as Denial of Service (DoS) attacks, IoT botnet infiltrations, malware exploits, phishing schemes, Man-in-the-Middle (MITM) attacks, routing manipulations, and challenges related to cloud security [8]. In a zero-trust security model, the inherent vulnerabilities of IoT are acknowledged, and trust is systematically withheld. Verification is required for all attempts to access network resources, regardless of location or prior authentication, establishing a comprehensive defense against the diverse threats originating both externally and internally within the IoT network [8].

* Corresponding author.
E-mail addresses: 2027016@stu.neu.edu.cn (D. Javeed), ShahidSaeedRana@mail.dlut.edu.cn (M.S. Saeed), 6122000014@tju.edu.cn (M. Adil), prabhat.kumar@lut.fi (P. Kumar), alireza.jolfaei@flinders.edu.au (A. Jolfaei).

https://doi.org/10.1016/j.adhoc.2024.103540
Received 2 December 2023; Received in revised form 19 March 2024; Accepted 4 May 2024
Available online 9 May 2024

The security of IoT networks against cyber threats has become a paramount research domain in recent years. Artificial Intelligence (AI) based approaches, especially Federated Learning (FL), have garnered considerable attention for proposing security solutions due to their reliability and privacy-preserving mechanism [9,10]. The FL-based Intrusion Detection System (IDS) operates in a systematic architecture without actual data exchange between the central server and the participant nodes; instead, the model updates are shared between the communication parties. This characteristic defines novelty and declares FL as an exceptional choice over conventional AI techniques where the participant nodes are supposed to share actual data with the centralized server [11]. To achieve intrusion detection objectives, the central server initializes a global model, broadcasts it to the participants, and the model is then trained on the local data occupied by the individual nodes. Once the training is performed, each node computes relevant model updates and transmits them to the central server, where the secure accumulation is achieved based on a multi-party secure aggregation technique. The central server then updates the global model and distributes it among the subordinate nodes, where the model is employed for intrusion detection at the local level [12], [13]. FL substantially supports continuous learning processes in which iterative cycles are performed, following which the central server keeps spreading the global model. After training at the local data nodes, the model updates are sent back to the server. This way, a chain of processes is instituted to deeply intensify the training mechanism after each cycle [14].

There are two key aspects that highlight the effectiveness of FL-based IDS. The first is the privacy-preserving mechanism that precludes actual data sharing across the system, leading towards privileges such as localization of sensitive information, prevention of network data exposure, and efficient utilization of segregated data streams [15]. It reduces the risks of data breaches during the model training process and provides a dynamic roadmap for implementing security schemes more suitably for sensitive environments. The second significant aspect is the global model updating procedure, leveraged by the collective contribution of local nodes instead of overburdening the central server. This phenomenon reduces the data transmission among the participant modes and the central server, resulting in less consumption of network resources, i.e., energy, communication channel occupancy, and communication maintenance [16]. Furthermore, this approach strengthens the system's overall performance by promoting heightened robustness, improved responsiveness, and increased agility [17]. The FL setting also conforms to the zero-trust verification model, refraining from assuming trust in any device or central entity. Each device actively contributes to the model training, and the subsequent updates undergo validation and aggregation in a secure manner, eliminating reliance on any single point of trust. In crafting a communication architecture for IoT networks that prioritizes resilience against threats and safeguards privacy, an IDS oriented towards FL emerges as a prominent and highly recommended solution.

1.1. Contributions

The main contributions of this research work are as follows:

- We propose an innovative approach for threat detection in IoT systems. Our method leverages Deep Federated Learning (DFL), which enhances privacy while effectively countering different types of attacks on IoT networks. Specifically, we employ horizontal FL, ensuring that data remains secure on local devices. Additionally, our approach adheres to a zero-trust model for detecting attacks.
- To enhance the intrusion detection task, we combine two deep learning models: Convolutional Neural Networks (CNN) and Bidirectional Long-Term Short Memory (BiLSTM). This combination leverages CNN for extracting spatial features and BiLSTM for understanding temporal patterns, significantly improving our system's ability to detect and adapt to various types of cyberattacks.
- Rigorous evaluation is conducted using two diverse and real-world traffic datasets, namely CICIDS2017 and Edge-IIoTset. These datasets encompass a wide range of traffic, including both benign and various types of malicious activities, ensuring the comprehensive assessment of our classifiers.
- We provide an in-depth performance evaluation, offering insights into the efficacy of each classifier. Furthermore, we present a detailed comparative analysis of the performance of our privacy-preserving DFL-based framework against the traditional centralized learning approach.

2. Existing literature

In recent years, FL-based vulnerability detection systems have symbolically obtained significant consideration and become a trending research domain. For example, the researchers in [18] proposed a novel FL-based attack detection scheme for IoT communications governed by the Message Queue Telemetry Transport (MQTT) protocol. The MQTT is one of the most commonly used protocols in IoT that becomes vulnerable to various crucial security threats such as memory computation attacks, buffer overflow, Denial of Service (DoS) attacks, Man-in-the-middle (MiTM) attacks, etc. The model is trained on the MQTT-IoT-IDS2020 dataset containing two million impressions of relevant attacks in IoT communication environments. The model has shown above 80% attack detection accuracy; however, minor communication latencies are also experienced. To evaluate the reliability of FL-driven threat investigation approaches in uncertain IoT communication systems, researchers in [19] use a Generative Adversarial Network (GAN) to design a model poisoning attack entity. They aim to highlight the probabilities of biased decisions, false positive rates, and compromised decisions of FL-driven vulnerability detection schemes. Then, a Convolutional Neural Network (CNN) empowered intrusion detection scheme is presented that is trained on the UNSW-NB15 dataset equipped with a relevant variety of concerned attacks. The designed scheme Social Intrusion Detection System (SIDS) also addresses the social interconnection among the participant objects of an IoT network to ensure a trust-oriented and privacy-preserving collaborative architecture. The experimental results declare the performance of SIDS for reliable detection of poisoning attacks with higher accuracy. CNN is also implied in another FL-empowered intrusion detection mechanism designed to identify malicious elements in IoT [20]. The proposed model is trained on the CSE-CICIDS2018 dataset and has shown reliable performance regarding the timely detection of 15 swear attack classes, including DoS-Slowloris, DoS-SlowHTTPTest, DoS-HOIC, DoS-GoldenEye, and brute force-XSS attacks. The FL-inspired approach suggests a reliable pathway to secure IoT communications; nevertheless, the model is unsuitable for large-scale IoT networks.

Researchers in [21] designed a semi-supervised FL-based (SSFL) anomaly detection framework for IoT networks. The proposed scheme aims to interrogate frequently occurring cyber-attacks in large-scale IoT communication environments with uneven traffic distribution. The model is leaped by a CNN and trained on an N-BaIoT dataset that contains a comprehensive classification of traffic categories. The model effectively detects emerging cyber threats; however, its communicational overhead increases while evaluating its performance. The authors in [22] also used CNN to design another decentralized FL mechanism for anomaly detection in IoT. The model aims to protect IoT communications against frequently occurring anomalies in IoT, and for a comprehensive training and testing process, the IoT23 dataset is acquired. The system has achieved reliable accuracy regarding attack detection. However, the proposed scheme shows a higher ratio of false positives. In [23], the researchers proposed an FL-empowered unique threat classification scheme for large-scale IoT communication systems. The designed framework is interlaced with a Gated Recurrent Unit (GRU) classifier trained on the ToN-IoT dataset. While evaluating the

Table 1
Limitations of existing literature.

Ref	Year	Contribution	Strength	Limitation
[18]	2023	An FL-based novel attack detection framework is presented for MQTT-based IoT networks	Detection of frequently occurring anomalies in MQTT-based IoT environments	Communication latencies are experienced while evaluating the performance
[19]	2023	The reliability of FL-driven threat detection approaches is evaluated for IoT networks	Detection of poisoning attacks with higher accuracy	The scheme is only limited to a specific attack category
[20]	2023	FL-inspired intrusion detection model is presented for IoT systems	Investigation of DoS attack categories in an efficient manner	The model is not appropriate for large-scale IoT networks
[21]	2022	A semi-supervised FL-driven IDS is designed for IoT	Timely detection of emerging cyber threats	The framework suffers from communication overhead
[22]	2022	A decentralized FL-driven anomaly-hunting scheme is proposed	Detection of frequently occurring attacks in IoT	The designed model shows an increased ratio of false-positives
[23]	2022	A FL-empowered threat classification framework is presented	Generic classification between swear-attacking categories	The training time of the model is considerably high
[24]	2022	GAN and FL-based adversary detection model is proposed	Identification of emerging cyber-attacks in IoT	End-to-end communication delays increased
[25]	2021	A proactive threat-hunting scheme is proposed for edge-assisted IoT scenarios	Identification of DoS, U2R, R2L and probe attacks	The model requires considerable computational resources
[26]	2021	A multi-view FL-oriented threat detection model is devised	Reliable hunting of brute-force attacks	Higher false-positives alarms
[27]	2021	A novel attack investigation scheme is presented for IoT communications	Protection of IoT networks against potential vulnerabilities	The training and testing time is remarkably extended
[28]	2021	An unsupervised FL-based anomaly detection framework is proposed for IoT	Detection of fifteen unique categories of potential threats	No active resilience against adversarial activities in broadly-expanded IoT systems

performance, the model is noticed to provide meaningful classification query flood attack, ping DDoS, and SYN DDoS attack categories. Nevertheless, the training time of the designed model is considerably high. Authors in [24] proposed a generic privacy-preserving IDS using the systematic strength of GAN and FL called FEDGAN-IDS. The researchers have used three large datasets, i.e., NSL-KDD, KDD-CUP99, and UNSW-NB15, to influence the model with a cumulative of 24 attack classes. The model has achieved satisfactory performance in detecting potential vulnerabilities; however, end-to-end communication delay increases while executing the scheme. To safeguard edge-assisted IoT systems, the authors in [25] devised an FL-based intelligent anomaly hunting model that is capable of addressing DoS attacks, User-to-Root (U2R) attacks, Remote-to-Local (R2L) attacks, and probe attacks. The mechanism is strengthened by CNN, trained on the NSL-KDD dataset, enclosing the instances of admissible vulnerabilities in edge-based IoT scenarios. Researchers in [26] proposed another FL-oriented novel, Multi-View Federated Learning Intrusion Detection (MV-FLID) for smart IoT networks. The scheme encompasses Random Forest (RF) as a classifier trained on the MQTT protocol dataset. Researchers have used the Grey Wolf Optimization (GWO) technique for the feature extraction process, which provides a multi-view hunting mechanism to excerpt optimal features in three views, e.g., uniflow, bi-flow, and packet features. The model actively detects vulnerabilities with notable accuracy. However, the false-positives ratio is significantly high.

The authors in [27] proposed another FL-based threat detection framework for IoT communications to ensure timely interrogation of vulnerabilities. The model is equipped with a GRU and is trained on a Modbus-based network dataset. The model's performance is evaluated in accuracy, precision, recall, and F1-score. The model is capable enough to detect potential threats in IoT networks. However, the training and testing duration of the designed scheme is remarkably extended. To establish a trustworthy anomaly detection mechanism, researchers in [28] suggest an unsupervised FL-based approach for smart IoT networks. The unlabeled data entities of the network are individually stored on the devices to mitigate the risk of centralized server failure and privacy breaches. The system is strengthened by an Artificial Neural Network (ANN) classifier trained on the CICIDS2017 dataset. The model aims to find fifteen different attack classes, including DoS hulk, botnet attacks, portscan attacks, DDoS LOIT, etc. Experiments prove the efficiency of their scheme in terms of higher accuracy achieved for attack detection against minimal training and validation loss. However, the system does not provide active resilience against adversarial activities in large-scale IoT networks. The existing literature is summarized in Table 1.

3. Proposed mechanism

In this section, we first provide the details of the dataset and pre-processing, followed by deep learning-based attack detectors. We further discuss the proposed FL-assisted intrusion detection system, followed by a centralized intrusion detection system. Moreover, the details about the system architecture are presented. The overview of our proposed framework is shown in Fig. 1.

3.1. Dataset and pre-processing

In this study, we used the CICIDS2017 [29] and Edge-IIoTset [30] datasets, both are reliable and publicly accessible datasets for network intrusion detection. The CICIDS2017 dataset comprises benign network traffic and various common network intrusions patterns such as DDoS, Port Scan, FTP-patator, and SSH-patator, etc. While, the Edge-IIoTset consists of normal, DDoS UDP, DDoS ICMP, SQL injection, Backdoor, Ransomware, MITM, etc. Originally, there were 84 network traffic features in the CICIDS2017., which were obtained by analyzing raw network packets using publicly available CICFlowMeter software [31]. To better suit our problem, we decided to alleviate irrelevant features from the CICIDS2017, such as Flow ID, Source Port, Source IP, Destination IP, Timestamp, and Protocol, because the dataset was produced in an isolated environment and the values of these features may potentially differ from real-world scenarios considering the network topology perspective. Additionally, we identified and eliminated 288,602 rows from the dataset that lacked labels; as a result, the final dataset included 2,830,743 samples and 78 features. The CICIDS2017 dataset encompasses a total of fifteen different classes, comprising one for benign network traffic and the remaining fourteen categories representing various forms of intrusion traffic in the network. The Edge-IIoTset originally consisted of 61 features and 2219201 observations including both benign and malicious traffic. To better align with our problem, we conducted feature reduction, resulting in a final dataset comprising 46 features along with a label. The details of these datasets can be seen in Table 2.

Furthermore, we performed several pre-processing steps to prepare the CICIDS2017 and Edge-IIoTset datasets for our proposed FL-based IDS. These steps were necessary for two main reasons. First, our proposed framework uses CNN models for data classification, thus requiring pre-processing steps such as normalization. Furthermore, distributive training was not specifically intended for this dataset. We split this dataset into many batches based on the number of edge devices used in the training procedure in order to imitate the FL training process. The pre-processing steps are as follows:

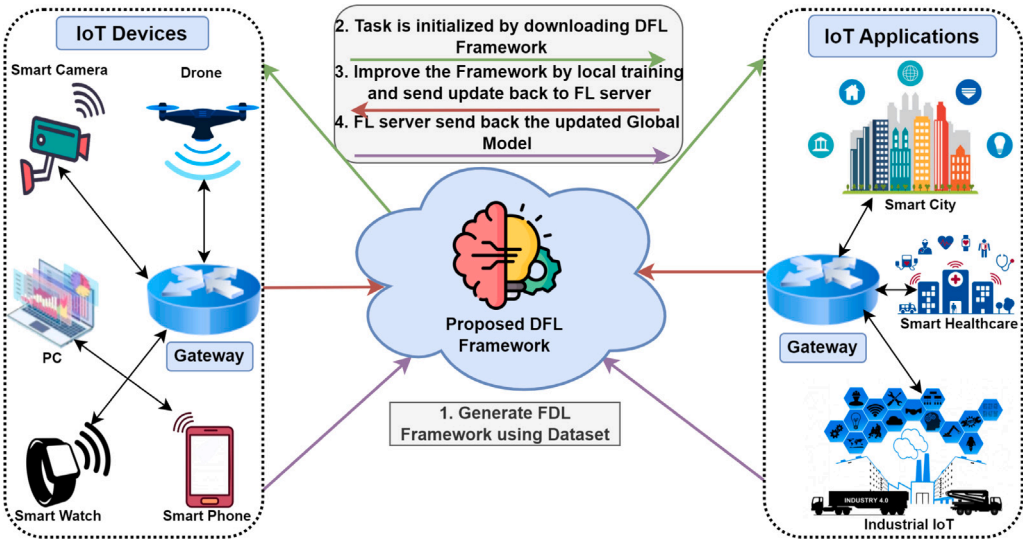


Fig. 1. Overview of the proposed framework.

Table 2
Datasets detail.

Dataset	Label	Total observations	Training samples	Testing samples
CICIDS2017	BENIGN	2273097	1818478	454619
	DoS Hulk	231073	184858	46215
	Brute Force	1507	1205	302
	XSS	652	522	130
	Infiltration	36	29	7
	Hearbleed	11	9	2
	Port_Scan	158930	127144	31786
	DDoS	128027	102422	25605
	DoS Golden_Eye	10293	8234	2059
	Dos Slow_loris	5796	4637	1159
	Dos Slow_http_test	5499	4399	1100
	FTP-Patator	7938	6350	1588
	SSH-Patator	5897	4717	1180
	Bot	1966	1573	393
Edge-IIoTset	BENIGN	404045	323,236	80,809
	DDoS_UDP	30515	24,412	6,103
	Fingerprinting	232	186	46
	Vulnerability_scanner	12428	9,942	2,486
	SQL_injection	12670	10,136	2,534
	Ransomware	2682	2,146	536
	DDoS_ICMP	29069	23,255	5,814
	Backdoor	6178	4,942	1,236
	DDoS_HTTP	12642	10,113	2,529
	DDoS_TCP	12593	10,074	2,519
	XSS	3970	3,176	794
	MITM	320	256	64
	Password	12536	10,028	2,508
	Uploading	9555	7,644	1,911
	Port_Scanning	5687	4,549	1,138

(1) Each feature's missing values were substituted with the class's average value.
(2) Each feature's infinite values have been replaced with the maximum value of the matching class.
(3) Replaced negative values in each feature with the minimum value of its corresponding class.
(4) Normalized the dataset features using unity-based normalization.
(5) Split the datasets up into many batches.
(6) Split the datasets into training and evaluation sets.

Eq. (3) serves as a means to standardize features by confining their value within the range of 0 and 1. In this equation, x_i represents the value of a feature, while x_{min} and x_{max} denote the minimum and maximum values of that particular feature, respectively. This normalization process is important as it ensures that no single feature dominates or distorts the overall analysis. To simulate the proposed FL training approach, we partitioned the datasets into multiple batches, each encompassing various classes. Following this division, we further split them within each batch into training and evaluation sets. [32].

$$\mathfrak{x}_{\text{norm}} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{1}$$

3.2. Hybrid deep learning based attack detector

In this part, we present an overview of the hybrid deep learning model CNN-BiLSTM that we have used in our proposed approach. CNN is a type of Feedforward Neural Network (FNN) with the ability to

extract spatial features [33], while BiLSTM is an extension of RNN capable of handling temporal dependencies in both forward and backward directions [34]. Leveraging CNN's capability to extract high-level features from the dataset, the proposed model initiates with a CNN layer. Afterward, the input data undergoes the convolutional operation within the convolutional layer, where multiple filters extract the important features to generate a feature map. After batch normalization, this map goes through a max pooling layer to save the most prominent features. After that, the output is sent to a BiLSTM layer to extract temporal features, and to prevent overfitting, it is followed by a dropout layer. Finally, the classification task is carried out by a fully linked layer using the SoftMax activation function.

3.2.1. Convolutional Neural Network (CNN)

CNN comprises two main components: convolution and pooling. In the convolution layer, filters are applied to the input matrix, producing a feature map. This involves sliding a kernel over the input matrix, calculating dot products, and summing to create the feature map [35]. Formally, the process can be defined as:

$$y(x) = f(\sum_{j=1}^{M}\sum_{i=1}^{N} w_{ij} \times x_{ij} + \mathcal{B}ias) \tag{2}$$

where, $f(*)$ represents the activation function (AF), and in our case, AF is ReLU (Rectified Linear Unit). w_{ij} corresponds to the convolution kernel weight at position i, j within a kernel of size $m \times n$ and $i, j \in (R)^{m,n} \times x_{ij}$ denotes the input values. After convolution, the pooling layer reduces the matrix size to prevent overfitting and enhance learning. We use Max pooling, which reduces sample size without affecting weights.

3.2.2. Batch normalization (BN)

BN is primarily used to prevent covariance shifts that can occur when transitioning from one layer to another within a DNN. These changes may cause instability and reduce the effectiveness of the learning process. BN plays a pivotal role in accelerating optimization while reducing generalization errors [36]. Furthermore, it standardizes the output from the CNN layer by scaling the data within the input layer to a unit norm, before processing it through the LSTM layer. Formally, the BN process is as follows:

$$\mathcal{X} = \frac{X - \mu\beta}{\sqrt{\delta_{\beta}^2 + \epsilon}} \tag{3}$$

where, X represents the data obtained from the Max Pooling layer, $\mu\beta$ and $\delta\beta$ denote the batch mean and variance, respectively, and ϵ is introduced to ensure a non-zero denominator. The result of Eq. (3) is further processed with a scale γ and adding a shift β to generate an output $\mathbb{Z}_{norm}$. These parameters play a pivotal role in enhancing the learning process as they are fine-tuned during the training process.

$$\mathbb{Z}_{norm} = \gamma\mathcal{X} + \beta \tag{4}$$

3.2.3. Bi long-short term memory

Bi Long-short memory (BiLSTM) is fundamentally similar to its unidirectional counterpart, LSTM, with the key distinction being its ability to propagate information in both backward and forward directions. The bidirectional aspect of the BiLSTM enables to access information from the past and future in a sequence, making it exceptionally useful for tasks involving large temporal dependencies. The architecture of BiLSTM consists of three main components: an input gate ($\mathbb{IN}^t$), a forget gate ($\mathbb{FG}^t$), and an output gate ($\mathbb{OUT}^t$), alongside a cell state ($\mathbb{Z}^t$) and a candidate for the cell state ($\mathbb{C}^t$). Formally, the forward ($\rightarrow$) and backward ($\leftarrow$) processes for BiLSTM can be defined as [37]:

$$\overrightarrow{\mathbb{C}^t} = \tanh((\overrightarrow{\theta_{\mathcal{Z}_c} \cdot \mathcal{H}^{\langle t-1\rangle}}) \times (\overrightarrow{\theta_{\mathcal{Z}_c} \cdot \mathcal{X}^t}) + \overrightarrow{\mathcal{B}ias_c}) \tag{5}$$

$$\overrightarrow{\mathbb{Z}^t} = (\overrightarrow{\mathbb{FG}^t \cdot \mathcal{H}^{\langle t-1\rangle}}) + (\overrightarrow{\mathbb{IN}^t \cdot \mathbb{C}^t}) \tag{6}$$

$$\overrightarrow{\mathbb{IN}^t} = \alpha((\overrightarrow{\theta_{\mathcal{Z}_{in}} \cdot \mathcal{H}^{\langle t-1\rangle}}) + (\overrightarrow{\theta_{\mathcal{Z}_{in}} \cdot \mathcal{X}^t}) + \overrightarrow{\mathcal{B}ias_{in}}) \tag{7}$$

$$\overleftarrow{\mathbb{C}^t} = \tanh((\overleftarrow{\theta_{\mathcal{Z}_c} \cdot \mathcal{H}^{\langle t-1\rangle}}) \times (\overleftarrow{\theta_{\mathcal{Z}_c} \cdot \mathcal{X}^t}) + \overleftarrow{\mathcal{B}ias_c}) \tag{8}$$

$$\overleftarrow{\mathbb{Z}^t} = (\overleftarrow{\mathbb{FG}^t \cdot \mathcal{H}^{\langle t-1\rangle}}) + (\overleftarrow{\mathbb{IN}^t \cdot \mathbb{C}^t}) \tag{9}$$

$$\overleftarrow{\mathbb{IN}^t} = \alpha((\overleftarrow{\theta_{\mathcal{Z}_{in}} \cdot \mathcal{H}^{\langle t-1\rangle}}) + (\overleftarrow{\theta_{\mathcal{Z}_{in}} \cdot \mathcal{X}^t}) + \overleftarrow{\mathcal{B}ias_{in}}) \tag{10}$$

The $(\mathbb{FG}^t)$ is an important component, utilizing the sigmoid function (σ) to decide or control what information from the previous hidden state $(\mathcal{H}^{\langle t-1\rangle})$ and current input $(\mathbb{X}^t)$ should be retained, it controls the information memory will receive.

$$\overrightarrow{\mathbb{FG}^t} = \alpha((\overrightarrow{\theta_{\mathcal{Z}_{fg}} \cdot \mathcal{H}^{\langle t-1\rangle}}) + (\overrightarrow{\theta_{\mathcal{Z}_{fg}} \cdot \mathcal{X}^t}) + \overrightarrow{\mathcal{B}ias_{fg}}) \tag{11}$$

$$\overleftarrow{\mathbb{FG}^t} = \alpha((\overleftarrow{\theta_{\mathcal{Z}_{fg}} \cdot \mathcal{H}^{\langle t-1\rangle}}) + (\overleftarrow{\theta_{\mathcal{Z}_{fg}} \cdot \mathcal{X}^t}) + \overleftarrow{\mathcal{B}ias_{fg}}) \tag{12}$$

Similar to other gates, the output gate ($\mathbb{OUT}^t$) also use sigmoid function (σ) to determines the current timesteps hidden state $\mathcal{H}^t$ based on information gathered from both the previous hidden state $\mathcal{H}^{\langle t-1\rangle}$ and the current input $\mathcal{X}^t$. This computed hidden state $\mathcal{H}^t$ encapsulates crucial information from both past and future contexts and serves as the foundation for generating predictions. Formally, the process can be defined in two steps:

$$\overrightarrow{\mathbb{OUT}^t} = \alpha((\overrightarrow{\theta_{\mathcal{Z}_{out}} \cdot \mathcal{H}^{\langle t-1\rangle}}) + (\overrightarrow{\theta_{\mathcal{Z}_{out}} \cdot \mathcal{X}^t}) + \overrightarrow{\mathcal{B}ias_{out}}) \tag{13}$$

$$\overleftarrow{\mathbb{OUT}^t} = \alpha((\overleftarrow{\theta_{\mathcal{Z}_{out}} \cdot \mathcal{H}^{\langle t-1\rangle}}) + (\overleftarrow{\theta_{\mathcal{Z}_{out}} \cdot \mathcal{X}^t}) + \overleftarrow{\mathcal{B}ias_{out}}) \tag{14}$$

$$\overleftarrow{\mathcal{H}^t} = \tanh(\overleftarrow{\mathcal{Z}^t}) \odot (\overleftarrow{\mathbb{OUT}^t}) \tag{15}$$

The $\mathcal{X}^t$ represents the given input sample at a specific timestamp t. While the weight matrices for cell state candidate $(\mathbb{C}^t)$, input gate $(\mathbb{IN}^t)$, forget gate $(\mathbb{FG}^t)$, and output gate $(\mathbb{OUT}^t)$ are denoted by $\overrightarrow{\theta_{\mathcal{Z}_c}}$, $\overrightarrow{\theta_{\mathcal{Z}_{in}}}$, $\overrightarrow{\theta_{\mathcal{Z}_{fg}}}$, $\overrightarrow{\theta_{\mathcal{Z}_{out}}}$ for the forward process, and $\overleftarrow{\theta_{\mathcal{Z}_c}}$, $\overleftarrow{\theta_{\mathcal{Z}_{in}}}$, $\overleftarrow{\theta_{\mathcal{Z}_{in}}}$, $\overleftarrow{\theta_{\mathcal{Z}_{fg}}}$, $\overleftarrow{\theta_{\mathcal{Z}_{out}}}$ for the backward process respectively. The corresponding biases for these gates are represented as $(\overrightarrow{\mathcal{B}ias_c})$, $(\overrightarrow{\mathcal{B}ias_{in}})$, $(\overrightarrow{\mathcal{B}ias_{fg}})$, $(\overrightarrow{\mathcal{B}ias_{out}})$ and $(\overleftarrow{\mathcal{B}ias_c})$, $(\overleftarrow{\mathcal{B}ias_{in}})$, $(\overleftarrow{\mathcal{B}ias_{fg}})$, $(\overleftarrow{\mathcal{B}ias_{out}})$ for both directions. The final state is represented by $\mathcal{H}^t$.

3.2.4. Dropout

In neural networks, dropout is a regularization method used to prevent overfitting. [38], whereby each neuron, except the output neurons, is randomly dropped with a probability p during a given training step, the process can be defined as:

$$\mathbb{M}_i \sim \text{Bernoulli}(1 - p) \tag{16}$$

where $\mathbb{M}_i$ is a binary mask for neuron i, sampled from a Bernoulli distribution with a probability of $1 - p$. During the training process, some neurons are deactivated, and do not perform any calculations, as their activations are set to zero using the mask:

$$\hat{a}_{drop} = a_i \times \mathbb{M}_i \tag{17}$$

Here, $\hat{a}_{drop}$ represents the modified activation of neuron i, and a_i is the original activation. Consequently, the learning model exhibits a slower convergence rate. Nevertheless, this procedure results in a reduction in inter-dependency between neurons across layers, improving the model's ability to generalize more effectively on unseen data. We have used a 0.2% dropout rate in this work.

Table 3
Implementation details of federated learning models.

Model	Hyperparameters	
DNN	Hidden layers	64, 64, 32
	Activation function	ReLU
	Edge device learning rate	0.01
	Edge device optimizer	Adam
	Server learning rate	0.05
	Server optimizer	Adam
	Loss function	Binary crossentropy
	No. of epochs	100
	Batch size	1024
	Training rounds	10
CNN-BiLSTM	Convolutional layers	1, filter size (3, 3), 32 filters
	Pooling layers	Max pooling, size (2, 2)
	Recurrent layers	BiLSTM, 32 units, BiLSTM, 16 units
	Dense layers	Dense, 1 unit
	Activation function	Sigmoid
	Loss function	Binary crossentropy
	No. of epochs	100

3.2.5. Fully connected layer

The final layer of the learning model processes the extracted feature map. A Fully Connected (FC) layer means that each of its neurons is connected to all neurons of the previous layer. The responsibility of this layer is to use the Softmax activation function to carry out classification tasks. [39]:

$$\sigma(\overrightarrow{\mathbb{Z}})_i = \frac{e^{\mathbb{Z}_i}}{\sum_{j=1}^{K} e^{\mathbb{Z}_j}} \tag{18}$$

where σ is the softmax, $\overrightarrow{\mathbb{Z}_i}$ is the input vector, $e_i^{\mathbb{Z}}$ is the standard exponential function applied to the input Z_i, and i ranges from 1 to K, where K is the number of classes. In order to categorize the provided input data into the right class and assign output probabilities, it is converted into a one-dimensional layer. The final output is represented by the output from this layer. Lastly, we compute the loss using categorical cross-entropy loss, which is defined as:

$$\mathcal{L}oss(\hat{y}_c, y_c) = -\sum_{j=0}^{M-1}\sum_{i=0}^{N-1} y_c^{x_i} \times \log(p(\hat{y}_{ic} = y_{ic}|x_i)) \tag{19}$$

where $\hat{y}_c$ corresponds to the model predicted output for class c, while y_c represents the actual true label for that particular class. The variable x signifies the input sequence, n denotes the total number of observations in the dataset, and p signifies the probability that the predicted output matches the actual label when conditioned on the input x. (See Table 3.)

3.3. FL-assisted intrusion detection

In the proposed privacy-preserved framework the process of horizontal FL paradigm is broadly divided into four phases. First, the coordinating server selects edge devices (also referred to as clients) and initializes a global model for all devices in the federated network, the selection process is important as it is considered the first line of defence against removing rogue devices. Second, each device starts training its local model with a private dataset and then uploads the updated model parameters to the coordinating server. Third, the server aggregates parameters from all selected devices and performs averaging. Finally, the new parameters are broadcast back to each device for the next round of communication and the same process repeats for a number of iterations until model convergence or pre-defined accuracy is achieved.

More specifically, we suppose that there are N edge devices participating in FL process and each device n holds $\mathbb{D}^n = \{(\mathbf{x}_i^n, \mathbf{y}_i^n)|i = 1, 2, \ldots, m\}$ training dataset, where $\mathbf{x}_i^n$ and $\mathbf{y}_i^n$ represents the feature space of a sample and the corresponding one-hot label of the sample respectively, with $m^n = |\mathbb{D}^n|$ is the size of device n training dataset. For a classification problem, $\mathbf{y}_i^n$ is a one-hot vector, i.e., $\mathbf{y}_i^n = [y_{i,0}^n, y_{i,1}^n, y_{i,2}^n, \ldots, y_{i,L-1}^n]$.

In our proposed framework for optimization, we utilize gradient updates to enhance model performance during the training process. These gradient updates are performed both at the coordinating server (global level) and on the edge devices (local level). The gradient update process in federated learning closely resembles a standard gradient update step, as described below

$$\theta^{t+1} = \theta^t = \eta_s \times f(\mathfrak{gradient}) \tag{20}$$

To break it down, θ^{t+1} represents the next iteration, while $f(\mathfrak{gradient})$ is a function of the edge device gradients computed for each round of FL training. Any machine learning algorithm necessitates an objective function for optimization. In our case, as it pertains to the classification problem, the objective function is the loss function, which is subject to minimization. This loss function essentially aggregates individual losses from each sample used for training. Formally, the standard definition of minimizing a loss function can be expressed as follows [40]:

$$\min_{\theta \in \mathbb{R}^d} f(\theta) \tag{21}$$

where

$$f(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}oss(\theta; x_i^n, y_i^n) \tag{22}$$

$\mathcal{L}oss(\times)$ represents the loss for a specific input sample (x_i^n, y_i^n), given model parameters θ. A gradient step involves the calculation of the gradient of $f(\theta)$, denoted as:

$$\mathfrak{g}(\theta) = \nabla f(\theta) = \frac{1}{m}\sum_i \mathfrak{g}_i(\theta) \tag{23}$$

where

$$\mathfrak{g}_i(\theta) = \frac{\partial \mathcal{L}oss((\theta; x_i^n, y_i^n))}{\partial \theta} \tag{24}$$

During optimization of the aforementioned loss function, instead of using the entire dataset in each iteration. we select only a subset of samples, whose indices are denoted by $\mathcal{B} \subset \{1, \ldots, m\}$ and the loss function would be:

$$\min_{\theta \in \mathbb{R}^d} f_{\mathcal{B}}(\theta) \tag{25}$$

where

$$f_{\mathcal{B}}(\theta) = \frac{1}{m_{\mathcal{B}}}\sum_{i \in \mathcal{B}} \mathcal{L}oss(\theta; x_i^n, y_i^n) \tag{26}$$

where $m_{\mathcal{B}}$ represents the total samples in a subset $m_{\mathcal{B}} = |\mathcal{B}|$. The gradient associated with the loss function for the subset is given by:

$$\mathfrak{g}_{\mathcal{B}}(\theta) = \nabla f_{\mathcal{B}}(\theta) = \frac{1}{m_{\mathcal{B}}}\sum_{i \in \mathcal{B}} \mathfrak{g}_i(\theta) \tag{27}$$

this standard loss function can be reformulated to represent our proposed FL setting for N edge devices, such that:

$$f(\theta) = \arg\min_{\theta \in \mathbb{R}^d} \sum_{n=1}^{N} \frac{\mathbb{D}^n}{\mathbb{D}} f_n(\theta) \tag{28}$$

where

$$f_n(\theta) = \frac{1}{\mathbb{D}^n}\sum_{i \in m^n} \mathcal{L}oss(\theta; x_i^n, y_i^n) \tag{29}$$

In this reformulated function, rather than computing the average loss function as an average over m available samples from a centralized training dataset as $\frac{1}{m}\sum_{i=1}^{m} f_i(\theta)$, we compute the average loss $f_n(\theta)$ for a specified edge device n as $\frac{1}{\mathbb{D}^n}\sum_{i=1}^{m} \mathcal{L}oss(\theta; x_i^n, y_i^n)$ and then aggregate the loss of all selected edge devices N, by computing a weighted average loss depends on the number of samples $\mathbb{D}^n$ the each device holds. For instance, in case of CICIDS2017 dataset we used, the input sample $\mathbf{x}_i^n$ is a d number of features thus $\mathbf{x}_i^n \in \mathbb{R}^d$. The $\mathbf{y}_i^n$ is an output vector

Algorithm 1 Federated-Averaging.
The $\mathbb{N}$ edge devices are indexed by $\mathfrak{n}$

Procedure SERVER EXECUTES:
initialize global model parameters θ_0
for each communication round $\mathbb{C} = \{0, 1, \ldots, n-1\}$ **do**
  $\mathfrak{m} \leftarrow \mathbb{MAX}(\mathbb{K} \times \mathbb{N}, 1)$ ▷ $\mathbb{K}$ = *fraction of devices*
  $\mathbb{S}_t \leftarrow$ *(random set of* $\mathfrak{m}$ *devices)*
  for each device $\mathfrak{n} \in \{0, 1, \ldots \mathbb{S}_t - 1\}$ **in parallel do**
    $\theta^{\mathfrak{n}}_{t+1} \leftarrow \mathbb{LocalUpdate}(\mathfrak{n}, \theta_t)$
  end for
  $\theta_{t+1} \leftarrow \sum_{\mathfrak{m}=1}^{\mathbb{N}} \frac{\mathbb{D}^{\mathfrak{m}}}{\mathbb{D}} \theta^{\mathfrak{m}}_{t+1}$
end for

Algorithm 2 Local Update $(\mathfrak{n}, \theta_t)$.
$\mathbb{B}$ is the local minibatch size, $\mathbb{E}$ is the number of local epochs, and $\mathfrak{n}$ is the learning rate

Procedure EDGE DEVICE EXECUTES:
$\mathbb{B} \leftarrow$ split $\mathfrak{m}^{\mathfrak{n}}$ into batches of size $\mathbb{B}$
for each local epoch $\mathfrak{i} \in \{0, 1, \ldots, \mathbb{E}-1\}$ **do**
  for batch $\mathfrak{b} \in \mathbb{B}$ **do**
    $\theta \leftarrow \theta - \beta \times \nabla \mathbb{Loss}(\theta; \mathfrak{b})$
  end for
end for
return θ to SERVER

represents $K = 2$ possibilities (i.e., Benign and Malicious) in the given case $\mathbf{y}^n_i \in \mathbb{R}^K$. The training dataset per device e.g. contains 100,000 input samples in total, so here $m^n = 100,000$. Each device n is required to compute the objective function in order to minimize the empirical risk over the local dataset $|\mathbb{D}^n|$, i.e., for simplicity, the total empirical risk for all participants can be written as:

$$f(\theta_1, \ldots, \theta_n) = \arg \min_{\theta_n \in \mathbb{R}^{D_n}} \sum_{n=1}^{N} \frac{\mathbb{D}^n}{\mathbb{D}} f_n(\theta) \tag{30}$$

As aforementioned, in our proposed framework before the first round of communication begins, a coordinating server initializes a global model and shares it with all selected devices within a federated network, such as

$$\mathfrak{g}_n \leftarrow \mathfrak{g}^* \tag{31}$$

where $\mathfrak{g}^*$ and $\mathfrak{g}_n$ represent the initial gradient information of the global model and initial gradient information of device n. Along with computing the average loss function in Eq. (29), we are also required to compute the gradients of the learning model in order to update the model parameters (also referred to as weights). In the proposed federated setting, each device computes the average gradient $\mathfrak{g}_n$ on the local dataset as follows:

$$\mathfrak{g}_n = \nabla f_n(\theta) = \frac{1}{\mathbb{D}^n} \sum_i \mathfrak{g}_i(\theta). \tag{32}$$

similarly the global gradient for n number of devices can be defined as:

$$\mathfrak{g}^* = \nabla f(\theta) = \sum_{n=1}^{N} \frac{\mathbb{D}^n}{\mathbb{D}} \nabla f_n(\theta) \tag{33}$$

These steps are iteratively performed for a predefined number of iterations. Our proposed framework pseudo-code is comprehensively outlined in Algorithm 3. Additionally, detailed algorithms for both the local level (at the edge side) and the global level (coordinating server side) are provided in Algorithms 1 and 2, respectively.

In the proposed FL-Assisted IDS we also employed a variation to the FedAvg approach to further facilitate the communication between edge devices and the coordinating server. Instead of uploading their

Algorithm 3 Proposed Hybrid FL-based IDS

Procedure Hybrid IDS:
INPUT: Read the Datasets (CICIDS 2017 and Edge-IIoTset)
OUTPUT: Classify and identify attack types
Data Pre-processing:
- ◇ Handling missing values by removing NaN entries
- ◇ Imputing infinity values
- ◇ Converting categorical features into numeric data
- ◇ Normalizing data within the range of 0 and 1

Split Dataset: Divide CICIDS 2017 and Edge-IIoTset into CIDS17_{train}, Edgeset_{train}, CIDS17_{Val}, Edgeset_{Val}, CIDS17_{test}, and Edgeset_{test}.
Model Initialization: SERVER EXECUTES [1]
Feature Extractor:

$$y(x) = f(\sum_{j=1}^{M} \sum_{i=1}^{N} w_{ij} \times x_{ij} + Bias)$$

Add BiLSTM layers and Softmax classifier:
Add CNN-BiLSTM layer using (Eq. 2) to (Eq. 15)
Add softmax layer

$$\sigma(\vec{\mathbb{Z}})_i = \frac{e^{\mathbb{Z}_i}}{\sum_{j=1}^{K} e^{\mathbb{Z}_j}}$$

calculate categorical cross-entropy loss

$$\mathcal{L}oss(\hat{y}_c, y_c) = -\sum_{j=0}^{M-1} \sum_{i=0}^{N-1} y_c^{x_i} \times \log(p(\hat{y}_{ic} = y_{ic}|x_i))$$

Model Distribution: Share the hybrid model with N number of edge devices
Model Training: EDGE DEVICE EXECUTES [2]
Perform Training, Validation, and Testing

$\mathbb{T}\text{raining} \leftarrow \text{CIDS17}_{train}$and Edgeset_{train}

$\mathbb{V}\text{alidation} \leftarrow \text{CIDS17}_{val}$and Edgeset_{val}

$\mathbb{T}\text{esting} \leftarrow \text{CIDS17}_{test}$and Edgeset_{test}

Assess performance using diverse metrics
End procedure

complete updated model weights θ^n_{t+1} to the server, each device sends only the difference between their updated θ^n_{t+1} and previous weights θ_t. This difference denoted as $\theta^n_{t+1} - \theta_t$, is calculated as follows:

$$\theta^n_{t+1} - \theta_t = -\eta \sum_{\mathfrak{b}} \nabla \mathcal{L}oss^n(\theta; \mathfrak{b}) \tag{34}$$

where $\nabla \mathcal{L}oss^n(\theta; \mathfrak{b})$ represents the gradient of the loss function for device n with batch $\mathfrak{b}$. The aggregator then computes $f(\mathfrak{gradient})$ as the weighted sum of device updates:

$$f(\mathfrak{gradient}) = \sum_n \frac{\mathbb{D}^n}{\mathbb{D}} (\theta^n_{t+1} - \theta_t) \tag{35}$$

This aggregated gradient is used in Eq. (20) to update the global model weights for the next round of communication as follows:

$$\theta_{t+1} = \theta_t - \eta \times \sum_n \frac{\mathbb{D}^n}{\mathbb{D}} (\theta^n_{t+1} - \theta_t) \tag{36}$$

When the learning rate is set to $\eta = 1$, it makes the update process equivalent to $\theta_{t+1} \leftarrow \sum_{n=1}^{N} \frac{\mathbb{D}^n}{\mathbb{D}} \theta^n_{t+1}$ the aggregation method used in FedAvg, where each device's contribution to the global model is directly included. This ensures that all edge devices' updates are fully considered during global model training. The advantage of this optimization, as described in Eq. (36) and the local updates as in Eq. (34),

is that it allows for the flexibility to apply various gradient optimization techniques, such as Adam [41], SGD [42], or RMSProp [43], at both the local-level and global-level. This flexibility is particularly useful in scenarios where edge devices have non-identically distributed data (non-IID), as it provides more control over fine-tuning the model weights to accommodate varying data distributions among different devices.

4. Experimental details

In this section, we provide complete details about the experimentation setup and the evaluation metrics. The details are provided as follows:

4.1. Experimental setup

We performed the experiments using a Windows 10-based Lenovo Legion Pro Core i9-13900HX PC processing @ 3.90 GHz with 32 GB RAM. We further utilized the NVIDIA GeForce RTX 4070 GPU for faster processing. We used numerous libraries of Python, i.e., Pandas, Numpy, Tensorflow, and Keras.

4.2. Evaluation metrics

Several performance evaluation metrics have been employed in this work to thoroughly evaluate the performance of the proposed FL-based IDS. The complete details about these evaluation metrics are as follows [44]:

4.2.1. ACCuracy (ACC)

indicates the ratio of correctly identified cases (both positive and negative) to the total number of input samples, and is given by:

$$ACC = \frac{TPR + TNR}{TPR + TNR + FPR + FNR} \tag{37}$$

4.2.2. PREcision (PRE)

indicates the ratio of correctly identified attacks to the total number of input samples predicted as attacks and is given by:

$$PRE = \frac{TPR}{TPR + FPR} \tag{38}$$

4.2.3. RECall (REC)

shows the proportion of attacks that were successfully detected to all of the input samples that were supposed to be classed as attacks. It is calculated as:

$$REC = \frac{TPR}{TPR + FNR} \tag{39}$$

4.2.4. Area under the curve (AUC)

It is a metric commonly used for binary classification problems, representing the area under the ROC curve. At various classification thresholds, the trade-off between a true positive rate (sensitivity) and a false positive rate (specificity) is depicted by the ROC curve. The AUC is calculated by integrating the ROC curve and is given by the integral of the ROC curve:

$$AUC = \int_0^1 TPR(FPR^{-1}(t)), dt \tag{40}$$

where FPR^{-1} is the inverse of the false positive rate function.

5. Performance evaluation

In this section, we evaluate the performance of the proposed FL-based IDS. The objectives of our experiments are broadly divided into the following parts: (1) Build, train, and evaluate a privacy-invasive centralized model, where the dataset is located at the centralized location; (2) Evaluate the effectiveness of our proposed FL-based IDS, a privacy-preserving approach; (3) Comparative analysis between centralized model and the FL-based model in terms of privacy and performance.

Table 4
Implementation details of centralized learning models.

Model	Hyperparameters	
CNN	Convolutional layers	32, 64
	Pooling layers	Max pooling, size (2, 2)
	Dense layers	64
	Activation function	ReLU
	Loss function	Binary crossentropy
	Optimizer	Adam
	No. of epochs	10
	Batch size	1024
DNN	Hidden layers	64, 64, 32
	Activation function	ReLU
	Edge device learning rate	0.01
	Edge device optimizer	Adam
	Server learning rate	0.05
	Server optimizer	Adam
	Loss function	Binary crossentropy
	No. of epochs	10
	Batch size	1024
CNN-BiLSTM	Convolutional layers	1, filter size (3, 3), 32 filters
	Pooling layers	Max pooling, size (2, 2)
	Recurrent layers	BiLSTM, 32 units, BiLSTM, 16 units
	Dense layers	Dense, 1 unit
	Activation function	Sigmoid
	Loss function	Binary crossentropy
	No. of epochs	10

5.1. Intrusion detection based on centralized learning

In the initial set of experiments, we developed a centralized IDS and employed three distinct classifiers: CNN, DNN, and the hybrid CNN-BiLSTM. The specific details for each classifier used in the centralized IDS are provided in Table 4. Initially, experiments were conducted following a centralized learning approach, where the model was constructed and trained using data collected and stored on a single node. This experimental setup mirrors the conventional detection frameworks in use today. In this approach, a central computational entity aggregates training data from various distributed sites and derives insights from it. In the context of IoT 4.0, where these sites can extend across wide geographical areas, the central entity often assumes the role of a 'trusted' and 'highly secure' entity. The centralized learning models—CNN, DNN, and CNN-BiLSTM—were evaluated on two distinct datasets, CICIDS2017 and Edge-IIoTset. The training and testing accuracies for each model on the CICIDS2017 dataset are detailed in Table 5, while the corresponding results for the Edge-IIoTset dataset are presented in Table 6. These accuracy metrics offer valuable insights into the models' performance across various benign and attack types. To further illustrate the performance, Figs. 2(a) and 2(b) showcase the accuracy trends of CNN, DNN, and CNN-BiLSTM models over 10 epochs for the CICIDS2017 and Edge-IIoTset datasets, respectively. Additionally, Fig. 3 presents comparative histograms, offering a visual comparison of the models' performances on the two datasets. These visualizations enhance our understanding of how each model performs relative to the others in different scenarios. The results highlight the highest accuracies achieved using the Edge-IIoT dataset, with DNN and CNN-BiLSTM models both attaining 99.99%, and the CNN model achieving 99.98%. Conversely, the lowest accuracies were observed with the CICIDS2017 dataset, where DNN recorded 99.91%. However, CNN-BiLSTM and CNN exhibited strong performances with accuracies of 99.99% and 99.98%, respectively.

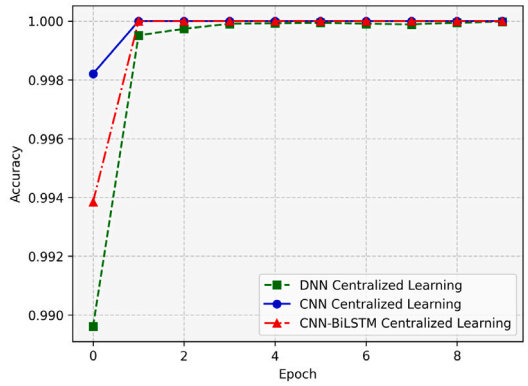
Although the centralized model has demonstrated strong performance on both datasets, it is essential to address the prolonged concerns and challenges pertaining to privacy, network latency, and various other issues associated with this approach, as these challenges have the potential to undermine the effectiveness of the entire system.

Table 5
Performance of different models in centralized learning on CICIDS2017 dataset.

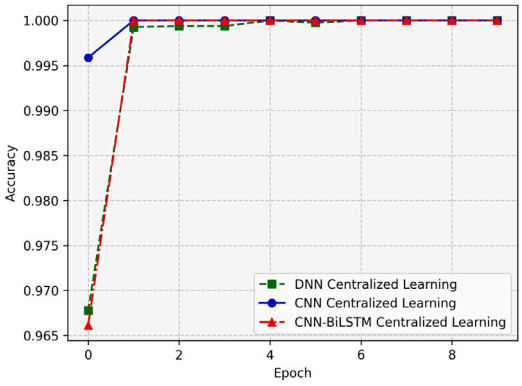
Model type	Train Acc (%)	Train loss	Test Acc (%)	Test loss
DNN	99.88	$1.341e^{-08}$	99.91	$1.713e^{-09}$
CNN	99.98	$1.754e^{-09}$	99.98	$9.885e^{-10}$
CNN-BiLSTM	99.97	$3.774e^{-08}$	99.99	$1.341e^{-08}$

Table 6
Performance of different models in centralized learning on Edge-IIoTset dataset.

Model type	Train Acc (%)	Train loss	Test Acc (%)	Test Loss
DNN	99.98	$3.500e^{-07}$	99.99	$1.043e^{-09}$
CNN	99.98	$1.215e^{-06}$	99.9	$1.114e^{-06}$
CNN-BiLSTM	99.96	$2.831e^{-05}$	99.99	$1.559e^{-10}$



(a) Using CICIDS2017 Dataset



(b) Using Edge-IIoTset Dataset

Fig. 2. Different models performance in centralized learning approach.

5.2. Intrusion detection based on federated learning

This experiment emulates the FL-based intrusion detection framework. In this approach, edge devices collaboratively share knowledge with each other without compromising local data privacy. It eliminates the need for third-party authorization, which often demands high levels of trust and robustness, potentially exceeding the affordability of some participants seeking to train a learning model.

In this series of experiments, we have designed a privacy-preserving FL environment. This setup includes one aggregation server, and multiple sets of edge IoT devices denoted as N. Specifically, in the initial experiment stage we set $N = 5$, in the second stage, N is increased to 10, in the third stage $N = 15$, and in the final stage N is expanded to 20.

Each edge device has its own local training dataset partitioned from the global dataset, which simulates individual private data. To ensure privacy and prevent data leakage, we have structured the experiments

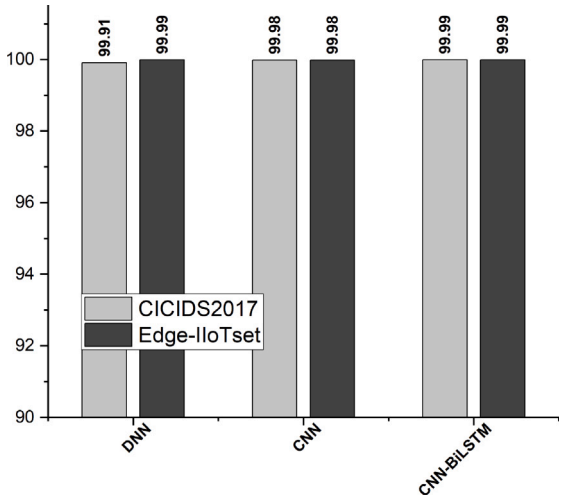
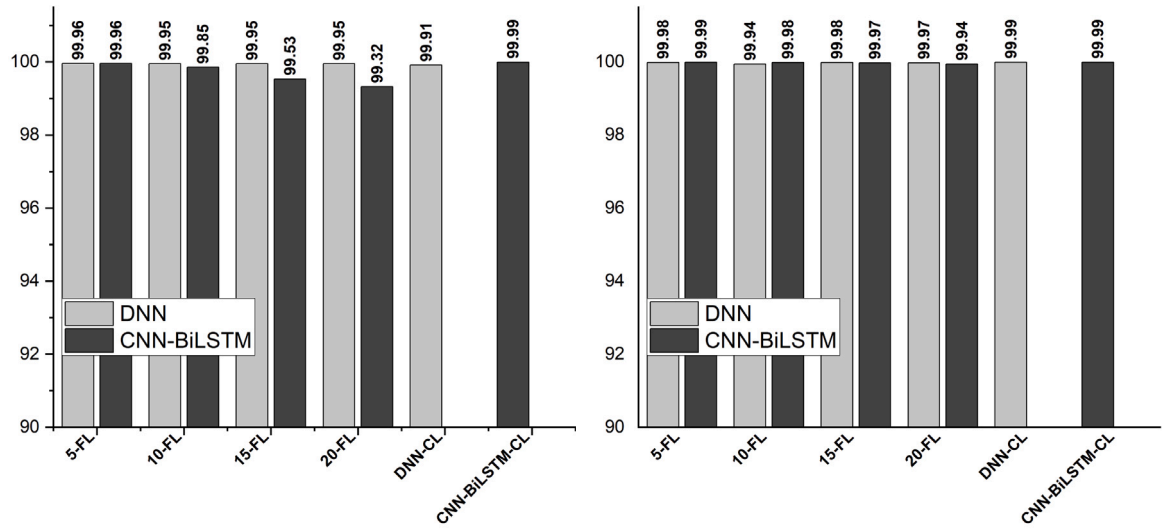


Fig. 3. Comparison of DNN, CNN, and CNN-BiLSTM Models on CICIDS 2017 and Edge-IIoTset datasets in centralized learning approach.

such that each device operates in entire isolation within the framework, and only communicates with the aggregation server through secured channels. During the training process, each edge device completes one local epoch and subsequently transmits the updated model parameters back to the aggregation server. In order to simulate real-world IoT scenarios, all selected edge devices participate in each communication round.

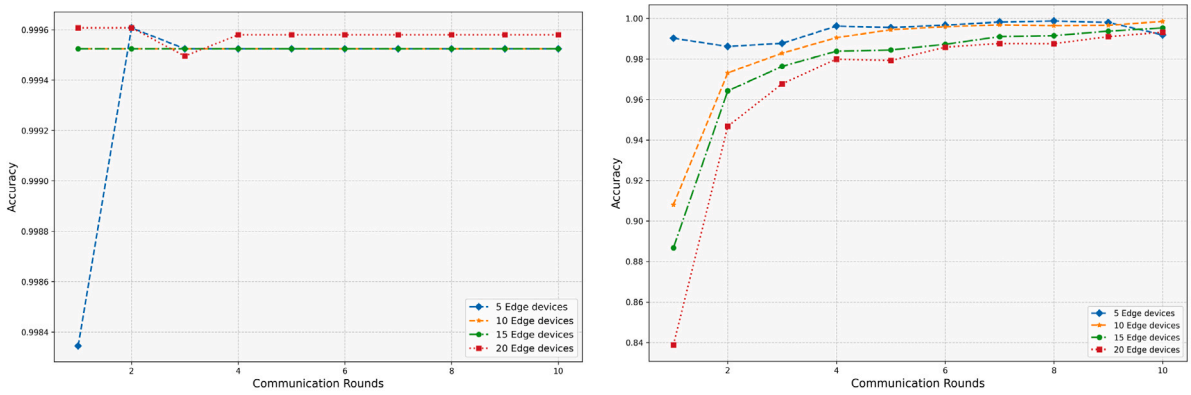
As aforementioned in FL settings, we conducted experiments involving different sets of edge devices (i.e., $N = 5, N = 10, N = 15$, and $N = 20$) over 10 communication rounds. Tables 7 and 8 present the performance of DNN and CNN-BiLSTM models for both CICIDS2017 and Edge-IIoTset datasets in terms of ACC, PRE, REC, AUC and loss, these tables provide a detailed interpretation of the models' across various benign and attack types. For a detailed assessment of DNN and CNN-BiLSTM models, we present accuracy results after each communication round on both datasets in Tables 9 and 10 respectively. A key finding from our study is that, with an increasing number of communication rounds in federated settings, the accuracy of both DNN and CNN-BiLSTM improves consistently across all sets of edge devices. This indicates that the collaborative training process benefits all edge devices simultaneously. Moreover, there is a discernible difference in the performance of all learning models between the first and final communication rounds, which is quite normal since all edge devices have to progressively learn from the collective knowledge in each subsequent round.

After completion of communication rounds, the results show the highest accuracies for both CICIDS2017 and Edge-IIoTset datasets were attained with 5 edge devices using CNN-BiLSTM models, achieving an impressive 99.99%. The DNN model also demonstrated strong performance with an accuracy of 99.98%. In contrast, the lowest accuracies were observed in the CICIDS2017 dataset, where the CNN-BiLSTM model recorded 99.32% accuracy under 20 edge devices. Based on the provided tables, it appears that the DNN model reaches high accuracy levels relatively quickly, and there is not much change in accuracy after the first few communication rounds. This suggests a faster convergence rate for the DNN model compared to the CNN-BiLSTM model. In contrast, the CNN-BiLSTM model continues to show changes in accuracy even after a few communication rounds, indicating a potentially slower convergence rate. Further, in some cases, the DNN and CNN-BiLSTM models trained in federated settings have demonstrated performance levels that are comparable to those of centrally trained models as shown in Fig. 4. In other words, our FL-based approach has enabled the global models to match or closely approach the accuracy of a



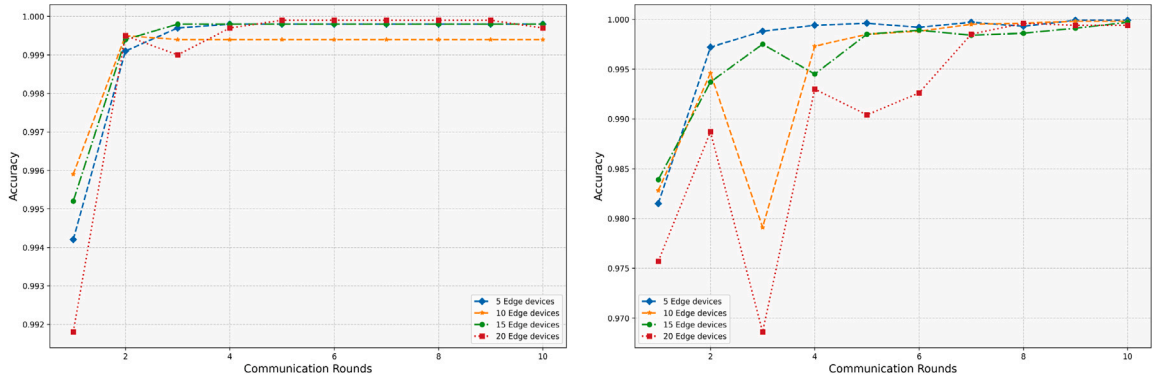
(a) Using CICIDS 2017 Dataset (b) Using Edge-IIoTset Dataset

Fig. 4. Performance comparison of different numbers of devices in FL against centralized model.



(a) Using DNN Model (b) Using CNN-BiLSTM Model

Fig. 5. Accuracy of different number of devices in FL setting on CICIDS2017 dataset.



(a) Using DNN Model (b) Using CNN-BiLSTM Model

Fig. 6. Accuracy of different number of devices in FL setting on edge-iiotset dataset.

Table 7
Performance on different numbers of devices using CICIDS2017 dataset.

Model	No. of devices	ACC (%)	PRE (%)	REC (%)	AUC (%)	Loss
CNN-BiLSTM	05 edge devices	99.19	98.99	98.58	99.92	0.021
	10 edge devices	99.85	99.75	99.80	99.95	0.006
	15 edge devices	99.53	99.32	99.27	99.89	0.017
	20 edge devices	99.32	99.12	98.86	99.85	0.024
DNN	05 edge devices	99.96	99.92	99.95	99.96	0.006
	10 edge devices	99.95	99.96	99.89	99.93	0.007
	15 edge devices	99.95	99.90	99.94	99.95	0.007
	20 edge devices	99.95	99.94	99.93	99.95	0.006

Table 8
Performance on different numbers of devices using Edge-IIoTset dataset.

Model	No. of devices	ACC (%)	PRE (%)	REC(%)	AUC (%)	Loss
CNN-BiLSTM	05 edge devices	99.99	99.99	1.0	99.99	0.0001
	10 edge devices	99.98	99.96	99.99	1.0	0.0003
	15 edge devices	99.97	99.92	99.99	99.99	0.001
	20 edge devices	99.94	99.88	99.96	99.97	0.003
DNN	05 edge devices	99.98	99.98	99.98	99.98	0.017
	10 edge devices	99.94	99.94	1.0	1.0	0.006
	15 edge devices	99.98	99.96	1.0	0.999	0.002
	20 edge devices	99.97	1.0	99.93	99.96	0.003

Table 9
Models accuracy after each communication round on the CICIDS2017 dataset in FL.

Model	No. of devices	1st round	2nd round	3rd round	4th round	5th round	6th round	7th round	8th round	9th round	10th round
CNN-BiLSTM	05 edge devices	95.02	98.61	98.77	99.62	99.55	99.67	99.82	99.87	99.80	99.99
	10 edge devices	90.80	97.31	98.28	99.05	99.44	99.59	99.68	99.64	99.66	99.85
	15 edge devices	88.67	96.42	97.63	98.38	98.44	98.72	99.10	99.15	99.37	99.53
	20 edge devices	83.87	94.67	96.77	97.99	97.92	98.58	98.76	98.75	99.1	99.32
DNN	05 edge devices	98.83	99.39	99.68	99.73	99.83	99.75	99.85	99.86	99.94	99.96
	10 edge devices	98.55	99.56	99.60	99.75	99.87	99.89	99.89	99.94	99.96	99.95
	15 edge devices	98.44	99.48	99.72	99.90	99.93	99.92	99.93	99.95	99.95	99.95
	20 edge devices	99.96	99.96	99.94	99.95	99.95	99.95	99.95	99.95	99.95	99.95

Table 10
Models accuracy after each communication round on the Edge-IIoTset dataset in FL.

Model	No. of devices	1st round	2nd round	3rd round	4th round	5th round	6th round	7th round	8th round	9th round	10th round
CNN-BiLSTM	05 edge devices	98.15	99.72	99.88	99.94	99.96	99.92	99.97	99.93	99.99	99.99
	10 edge devices	98.28	99.46	97.91	99.73	99.85	99.88	99.95	99.96	99.98	99.98
	15 edge devices	98.39	99.37	99.75	99.45	99.85	99.89	99.84	99.86	99.91	99.97
	20 edge devices	97.57	98.87	96.86	99.30	99.04	99.26	99.85	99.96	99.94	99.94
DNN	05 edge devices	99.42	99.91	99.97	99.98	99.98	99.98	99.98	99.98	99.98	99.98
	10 edge devices	99.59	99.95	99.94	99.94	99.94	99.94	99.94	99.94	99.94	99.94
	15 edge devices	99.52	99.94	99.98	99.98	99.98	99.98	99.98	99.98	99.98	99.98
	20 edge devices	99.18	99.95	99.90	99.97	99.99	99.99	99.99	99.99	99.99	99.97

centralized model in specific cases, which can be seen in Figs. 4(a) and 4(b) for CICIDS2017 and Edge-IIoTset datasets respectively. This highlights the effectiveness of our proposed method in leveraging the collective knowledge of distributed edge devices to achieve nearly identical results.

To comprehensively evaluate the effectiveness of our proposed framework, we present the accuracy trends of both learning models on the CICIDS2017 and Edge-IIoTset datasets for all sets of edge devices in Figs. 5 and 6. An intriguing observation from our study is a temporary decline in performance experienced by the CNN-BiLSTM model for both datasets, as evident in Figs. 5(b) and 6(b). However, the model exhibited a swift recovery after a few communication rounds. This issue can be effectively mitigated by opting for a smaller number of epochs during the training phase. In contrast, the DNN consistently demonstrated stable performance throughout all communication rounds, as illustrated in Fig. 5(a) for the CICIDS2017 dataset and Fig. 6(a) for the Edge-IIoTset dataset. This detailed assessment provides valuable insights into the dynamics of model performance over the course of communication rounds, highlighting specific considerations for optimizing the training process and ensuring stable and effective intrusion detection in the proposed framework.

6. Conclusion

In conclusion, this research focuses on addressing the challenges presented by the rapid growth of Internet of Things (IoT) devices, with an emphasis on the need for strong security measures. The increasing frequency of cyberattacks on interconnected systems highlights the difficulty of creating a scalable and zero-trust Intrusion Detection System (IDS) based on Federated Learning (FL) for IoT. The proposed hybrid deep learning model, which combines Convolutional Neural Networks (CNN) and Bidirectional Long-Term Short Memory (BiLSTM), achieves high accuracy and recall when tested with 20 edge devices across 10 communication rounds using the CICIDS2017 and Edge-IIoTset datasets. By locally training the data of each participant, the original data is kept secure from potential cyberattacks, and only the local training outcomes are shared for collaborative learning. In the future, our aim is to create our own FL environment using Raspberry Pi to assess the performance of the proposed approach. This setup will enable us to thoroughly evaluate and improve the performance of our proposed approach in a practical, real-world context, ensuring its effectiveness and applicability in various IoT scenarios.

CRediT authorship contribution statement

Danish Javeed: Methodology, Conceptualization, Visualization, Writing – original draft. **Muhammad Shahid Saeed:** Conceptualization, Data curation, Formal analysis, Writing – original draft. **Muhammad Adil:** Writing – original draft, Conceptualization, Data curation, Formal analysis, Visualization. **Prabhat Kumar:** Writing – original draft, Visualization, Supervision, Formal analysis, Data curation, Conceptualization. **Alireza Jolfaei:** Writing – original draft, Supervision, Formal analysis, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the Research Council of Finland with CHIST-ERA, grant agreement no - 359790, Di4SPDS-Distributed Intelligence for Enhancing Security and Privacy of Decentralized and Distributed Systems.

References

[1] A. Thakkar, R. Lohiya, Attack classification of imbalanced intrusion data for IoT network using ensemble learning-based deep neural network, IEEE Internet Things J. (2023).
[2] M. Malik, M. Dutta, et al., Feature engineering and machine learning framework for DDoS attack detection in the standardized internet of things, IEEE Internet Things J. (2023).
[3] P. Kumar, R. Kumar, A. Aljuhani, D. Javeed, A. Jolfaei, A.N. Islam, Digital twin-driven SDN for smart grid: A deep learning integrated blockchain for cybersecurity, Sol. Energy 263 (2023) 111921.
[4] D. Javeed, T. Gao, M.S. Saeed, M.T. Khan, FOG-empowered augmented intelligence-based proactive defensive mechanism for IoT-enabled smart industries, IEEE Internet Things J. (2023).
[5] N. Cheng, S. Wu, X. Wang, Z. Yin, C. Li, W. Chen, F. Chen, AI for UAV-assisted IoT applications: A comprehensive review, IEEE Internet Things J. (2023).
[6] M. Zawish, N. Ashraf, R.I. Ansari, S. Davy, Energy-aware AI-driven framework for edge-computing-based IoT applications, IEEE Internet Things J. 10 (6) (2022) 5013–5023.
[7] G.W. de Oliveira, M. Nogueira, A.L. dos Santos, D.M. Batista, Intelligent VNF placement to mitigate DDoS attacks on industrial IoT, IEEE Trans. Netw. Serv. Manag. (2023).
[8] Y.R. Siwakoti, M. Bhurtel, D.B. Rawat, A. Oest, R. Johnson, Advances in IoT security: Vulnerabilities, enabled criminal services, attacks and countermeasures, IEEE Internet Things J. (2023).
[9] J. Cui, H. Sun, H. Zhong, J. Zhang, L. Wei, I. Bolodurina, D. He, Collaborative intrusion detection system for SDVN: A fairness federated deep learning approach, IEEE Trans. Parallel Distrib. Syst. (2023).
[10] A. Muhammad, K. Lin, J. Gao, B. Chen, Robust multi-model personalized federated learning via model distillation, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2021, pp. 432–446.
[11] P. Singh, G.S. Gaba, A. Kaur, M. Hedabou, A. Gurtov, Dew-cloud-based hierarchical federated learning for intrusion detection in IoMT, IEEE J. Biomed. Health Inform. 27 (2) (2022) 722–731.
[12] P. Ruzafa-Alcázar, P. Fernández-Saura, E. Mármol-Campos, A. González-Vidal, J.L. Hernández-Ramos, J. Bernal-Bernabe, A.F. Skarmeta, Intrusion detection based on privacy-preserving federated learning for the industrial IoT, IEEE Trans. Ind. Inform. 19 (2) (2021) 1145–1154.
[13] R.R. dos Santos, E.K. Viegas, A.O. Santin, P. Tedeschi, Federated learning for reliable model updates in network-based intrusion detection, Comput. Secur. 133 (2023) 103413.
[14] M. Sarhan, S. Layeghy, N. Moustafa, M. Portmann, Cyber threat intelligence sharing scheme based on federated learning for network intrusion detection, J. Netw. Syst. Manage. 31 (1) (2023) 3.
[15] F. Naeem, M. Ali, G. Kaddoum, Federated-learning-empowered semi-supervised active learning framework for intrusion detection in ZSM, IEEE Commun. Mag. 61 (2) (2023) 88–94.
[16] V.T. Truong, L.B. Le, MetaCIDS: Privacy-preserving collaborative intrusion detection for metaverse based on blockchain and online federated learning, IEEE Open J. Comput. Soc. (2023).
[17] J. Li, X. Tong, J. Liu, L. Cheng, An efficient federated learning system for network intrusion detection, IEEE Syst. J. (2023).
[18] A. Omotosho, Y. Qendah, C. Hammer, IDS-MA: Intrusion detection system for IoT MQTT attacks using centralized and federated learning, in: 2023 IEEE 47th Annual Computers, Software, and Applications Conference, COMPSAC, IEEE, 2023, pp. 678–688.
[19] M. Amiri-Zarandi, R.A. Dara, X. Lin, SIDS: A federated learning approach for intrusion detection in IoT using social Internet of Things, Comput. Netw. 236 (2023) 110005.
[20] N. Hamdi, Federated learning-based intrusion detection system for internet of things, Int. J. Inf. Secur. 22 (6) (2023) 1937–1948.
[21] R. Zhao, Y. Wang, Z. Xue, T. Ohtsuki, B. Adebisi, G. Gui, Semi-supervised federated learning based intrusion detection method for internet of things, IEEE Internet Things J. (2022).
[22] Z. Lian, C. Su, Decentralized federated learning for Internet of Things anomaly detection, in: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, 2022, pp. 1249–1251.
[23] T.A. Ahanger, A. Aldaej, M. Atiquzzaman, I. Ullah, M. Yousufudin, Federated learning-inspired technique for attack classification in IoT networks, Mathematics 10 (12) (2022) 2141.
[24] A. Tabassum, A. Erbad, W. Lebda, A. Mohamed, M. Guizani, Fedgan-ids: Privacy-preserving ids using gan and federated learning, Comput. Commun. 192 (2022) 299–310.
[25] D. Man, F. Zeng, W. Yang, M. Yu, J. Lv, Y. Wang, Intelligent intrusion detection based on federated learning for edge-assisted Internet of Things, Secur. Commun. Netw. 2021 (2021) 1–11.
[26] D.C. Attota, V. Mothukuri, R.M. Parizi, S. Pouriyeh, An ensemble multi-view federated learning intrusion detection for IoT, IEEE Access 9 (2021) 117734–117745.
[27] V. Mothukuri, P. Khare, R.M. Parizi, S. Pouriyeh, A. Dehghantanha, G. Srivastava, Federated-learning-based anomaly detection for IoT security attacks, IEEE Internet Things J. 9 (4) (2021) 2545–2554.
[28] K. Yadav, B.B. Gupta, C.-H. Hsu, K.T. Chui, Unsupervised federated learning based IoT intrusion detection, in: 2021 IEEE 10th Global Conference on Consumer Electronics, GCCE, IEEE, 2021, pp. 298–301.
[29] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, in: International Conference on Information Systems Security and Privacy, 2018, [Online]. Available: https://api.semanticscholar.org/CorpusID:4707749.
[30] M.A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, H. Janicke, Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning, IEEE Access 10 (2022) 40281–40306.
[31] A.H. Lashkari, G. Draper-Gil, M.S.I. Mamun, A.A. Ghorbani, Characterization of tor traffic using time based features, in: International Conference on Information Systems Security and Privacy, 2017, [Online]. Available: https://api.semanticscholar.org/CorpusID:38122314.
[32] R. Kumar, D. Javeed, A. Aljuhani, A. Jolfaei, P. Kumar, A.N. Islam, Blockchain-based authentication and explainable AI for securing consumer IoT applications, IEEE Trans. Consum. Electron. (2023).
[33] L.O. Chua, T. Roska, The CNN paradigm, IEEE Trans. Circuits Syst. I 40 (3) (1993) 147–156.
[34] K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber, LSTM: A search space odyssey, IEEE Trans. Neural Netw. Learn. Syst. 28 (10) (2016) 2222–2232.
[35] K. O'Shea, R. Nash, An introduction to convolutional neural networks, 2015, ArXiv, arXiv:1511.08458, [Online]. Available: https://api.semanticscholar.org/CorpusID:9398408.
[36] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How does batch normalization help optimization? Adv. Neural Inf. Process. Syst. 31 (2018).
[37] M. Schuster, K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (1997) 2673–2681, http://dx.doi.org/10.1109/78.650093.
[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[39] M. Wang, S. Lu, D. Zhu, J. Lin, Z. Wang, A high-speed and low-complexity architecture for softmax function in deep learning, in: 2018 IEEE Asia Pacific Conference on Circuits and Systems, APCCAS, IEEE, 2018, pp. 223–226.
[40] Y. Tian, D. Su, S. Lauria, X. Liu, Recent advances on loss functions in deep learning for computer vision, Neurocomputing 497 (2022) 129–158.
[41] Z. Zhang, Improved adam optimizer for deep neural networks, in: 2018 IEEE/ACM 26th International Symposium on Quality of Service, IWQoS, Ieee, 2018, pp. 1–2.
[42] S.-i. Amari, Backpropagation and stochastic gradient descent method, Neurocomputing 5 (4–5) (1993) 185–196.

[43] M.C. Mukkamala, M. Hein, Variants of rmsprop and adagrad with logarithmic regret bounds, in: International Conference on Machine Learning, PMLR, 2017, pp. 2545–2553.
[44] D. Javeed, T. Gao, M.S. Saeed, P. Kumar, R. Kumar, A. Jolfaei, A softwarized intrusion detection system for iot-enabled smart healthcare system, ACM Trans. Internet Technol. (2023).

Danish Javeed is currently pursuing a Ph.D. degree in Software Engineering, specializing in Informaton Security with the Software College, Northeastern University, China under the prestigious fellowship of Ministry of Education funded by the Government of China. He got his M.E degree in Computer Applied Technology from Changchun University of Science and Technology, China, under the same fellowship in 2020. He is also working on various research projects with researchers from the LUT School of Engineering Science, LUT University, Lappeenranta, Finland. He has many research contributions in the area of Deep Learning, Cybersecurity, Intrusion Detection and Prevention Systems, the Internet of Things, Software-defned Networking, and Edge Computing. He has authored or co-authored over 15+ publications in high-ranked journals and conferences. He is also an IEEE Student Member.

Muhammad Shahid Saeed is currently pursuing a Ph.D. degree in Software Engineering, with the Dalian University of Technology, PR China under the prestigious fellowship of Ministry of Education funded by the Government of China. He is also working on various projects in collaboration with researchers from Northeastern University, China. He has a fet research contributions in the area of Intrusion Detection and Prevention Systems, the Internet of Things, Industry 4.0, Software-defned Networking, and Edge Computing.

Muhammad Adil received Masters degree in computer science and technology from Dalian University of Technology, China in 2022. He is currently he is pursuing Phd in computer science and technology from Tianjin university, China. His main research includes privacy-preserved distributive machine learning, time sensitive networking, and 5G for industrial networks.

Prabhat Kumar received his Ph.D. degree in Information Technology, National Institute of Technology Raipur, Raipur, India, under the prestigious fellowship of Ministry of Human Resource and Development (MHRD) funded by the Government of India in 2022. Thereafter, he worked with Indian Institute of Technology Hyderabad, India as a Postdoctoral Researcher under project "Development of Indian Telecommunication Security Assurance Requirements for IoT devices". He is currently working as Post Doctoral Researcher with the Department of Software Engineering, LUT School of Engineering Science, LUT University, Lappeenranta, Finland. He has many research contributions in the area of Machine Learning, Deep Learning, Federated Learning, Big Data Analytics, Cybersecurity, Blockchain, Cloud Computing, Internet of Things and Software Defined Networking. He has authored or coauthored over 35+ publications in high-ranked journals and conferences, including 13+ IEEE TRANSACTIONS paper. One of his Ph.D. publication was recognized as a top cited article by WILEY in 2020–21. He is also an IEEE Member.

Alireza Jolfaei is an Associate Professor of Networking and CyberSecurity in the College of Science and Engineering at Flinders University, Adelaide, Australia. He is a Senior Member of the IEEE and a Distinguished Speaker of the ACM. His main research interest is in Cyber-Physical Systems Security. He has published over 100 papers, this appeared in peer-reviewed journals, conference proceedings, and books. Before Flinders University, he has been a faculty member with Macquarie University, Federation University, and Temple University in Philadelphia, PA, USA. He received the prestigious IEEE Australian council award for his research paper published in the IEEE Transactions on Information Forensics and Security. Dr. Jolfaei is the IEEE Consumer Technology Publication Board member and the Editor-in-Chief of the Consumer Technology Society World Newsletter. He has served as the Regional Chair of the IEEE Technology and Engineering Management Society's Membership Development and Activities for Australia. He has served as a program coChair, a track Chair, a session Chair, and a Technical Program Committee member, for major conferences, including IEEE TrustCom and IEEE ICCCN.