A Novel Intrusion Detection System for Dew Computing Environments Based on an Enhanced Federated Deep Learning Model

Alireza Fadaei[1], Behrang Barekatain[1,2*]

[1] Department of Computer Engineering, Na.C., Islamic Azad University, Najafabad, Iran
[2] Big data Research Center, Na.C., Islamic Azad University, Najafabad, Iran
*Corresponding Author: Behrang_Barekatain@iau.ac.ir

Abstract

Recently, dew computing has drawn extreme attention in response to the ever-increasing demand for processing massive data rapidly in the Industrial Internet of Things (IIoT). However, the vulnerability of such infrastructures to cyberattacks has also risen drastically. Localized services and uninterrupted functionality even during disconnections represent key advantages of dew computing. Nevertheless, due to computational resource limitations and connectivity instabilities, conventional intrusion detection systems, which are mostly based on or require massive computational resources, may lack efficiency in such networks. In contrast to existing dew or edge intrusion detection approaches that commonly rely on centralized training with raw traffic transferred to a server, or adopt heavier deep architectures that are less suitable under dew constraints, this study proposes a single-layer 1D CNN that offers a practical balance between detection performance and deployment feasibility in resource-constrained and intermittently connected environments. The proposed model was implemented through federated learning on a fully dew-layer architecture, where the aggregation server and all participating nodes are deployed within the dew layer and proceed with the training process without dependence on the upper (fog or cloud) layers. This approach reduces bandwidth consumption, ensures data privacy, and significantly decreases latency, enabling effective training, even under unstable conditions. Edge-IIoTset data were preprocessed, and influential features were extracted through mutual information (MI) during the development process. The proposed model was evaluated in binary, six-class, and fifteen-class scenarios in a centralized setting and then simulated in a federated setting. It was found to show stable performance under both settings, yielding an average detection accuracy of 97.01% under the fifteen-class scenario in the centralized setting and 96.67% in the federated setting. The results of this study will be of great help to future researchers in advancing their goals in this research field.

Keywords: Industrial Internet of Things (IIoT), Dew computing, Intrusion Detection System (IDS), Federated Learning, 1D-CNN (Convolutional Neural Network), Edge-IIoTset

1. Introduction

Considering the ever-increasing growth of the Industrial Internet of Things (IIoT) and the continuous generation of sensitive data, centralized architectures such as cloud computing cannot individually meet the real-time requirements of such systems since the continuous transmission of massive data to cloud centers raises latency and bandwidth consumption, and the massive dependence on stable connectivity threatens uninterrupted functionality [1,2]. To address this challenge, dew computing was introduced as the closest layer to the data source. It fulfills processing at the sensor and actuator level and remains autonomous even in the event of Internet disconnections [3–5]. In light of its operational independence, cooperation with upper layers, and real-time responsiveness, dew computing has become a key setting for reliable systems in areas such as IIoT, health, smart agriculture, autonomous vehicles, and robotics [2,6]. Furthermore, security and privacy are protected by storing the data in the generation location and decreasing the demand for raw data transmission [7,8].

Apart from its advantages, the dew layer encounters a set of intrinsic challenges. Limited processing power, memory, and bandwidth prevent the fulfillment of heavy and complex tasks and increase energy consumption [4,9]. Hardware and software heterogeneity between nodes challenges the scalability and maintenance of systems [5]. Moreover, data protection and security are a key dew-layer challenge to be addressed [9]. Hence, this study particularly focuses on the security challenge in the dew layer. This challenge can be rooted in the processing, communication, and structural limitations of the dew layer as lightweight equipment with no strong protective infrastructure cannot implement expensive security mechanisms, and a decentralized structure raises the risk of rapid attack propagation [7,8,10]. Therefore, the dew layer has a higher vulnerability than other layers to unauthorized access, sensitive data leakage, real-time and injection attacks, malware contamination, identity spoofing, and man-in-the-middle attacks, which could lead to serious functionality interruptions [4,7,9]. If this challenge is not addressed effectively, the compromise of dew nodes can propagate to upper layers and lead to critical data corruption, disrupting high-level decision-making and inducing serious functionality interruptions [7,8]. Therefore, it is crucial to develop lightweight and rapid intrusion detection systems for the dew layer.

Security approaches proposed for IoT in the literature can be divided into three groups:

(I) Distributed intrusion detection systems operating through node collaboration and trust mechanisms based on machine learning (ML) or Bayesian inference [11,12]; however, they are vulnerable to insider attacks and would require a more intricate design for heterogeneous settings;

(II) Cloud-based systems perform security analysis in the central layer and utilize strong resources [13,14]. However, such systems are dependent on permanent Internet connections and have high latency;

(III) ML-based systems have been employed to detect complex threats in recent years [15–18]. Despite their efficiency in extracting hidden nonlinear patterns from raw data, ML-based systems would not be efficient for the dew layer due to their computational complexity and demand for massive data and high energy.

To address the communication limitations and security requirements of the dew layer, the present study implements a lightweight 1D-CNN intrusion detection system via federated learning using a fully dew-layer architecture. In the proposed model, both the aggregation server and all nodes are within the dew layer, and training is performed independently of cloud/fog layers. To cope with limited computational resources, the model was designed with a small set of parameters and a short inference path, compared to heavier deep architectures reported in the literature, such as CNN+LSTM+GRU and LSTM+CNN models [19,20], and input dimensionality would be decreased through MI-based feature selection. The exchange of

data is limited to model updating, and no raw data is discharged from the nodes in light of federated aggregation. As a result, the proposed model not only significantly reduces bandwidth consumption and ensures privacy protection but also enables stable and low-latency training, even under unstable connectivity.
A total of fifteen federated scenarios were designed and simulated to evaluate the proposed model comprehensively based on three parameters: (I) number of nodes (as a measure of network size and data distribution level), (II) participation rate (to address offline/absent nodes), and (III) number of training rounds (regarding limited computational resources). Finally, the performance of the proposed model was evaluated using the Edge-IIoTset [21] in binary, six-class, and fifteen-class classification scenarios.
Overall, the main contributions of this study include:

- Deploying a lightweight and efficient intrusion detection system for the dew layer, where lightweight is explicitly defined in terms of (i) parameter count, (ii) model size (memory footprint), and (iii) inference time. By reducing the number of trainable parameters, the proposed 1D-CNN lowers both the memory requirement and the computational burden, while its compact representation and short inference path enable fast on-device inference. Consequently, the model is practically deployable on dew servers (e.g., PC/workstation-class nodes) and supports continuous, cloud-independent operation under intermittent connectivity.
- Reducing communication overhead in federated learning by transmitting model updates rather than raw traffic traces. Specifically, each round exchanges only the model parameters/updates, so the communication cost scales with the model size and number of participating clients, instead of the volume of raw data required in centralized training; this reduces bandwidth demand and improves feasibility under dew connectivity constraints.
- Shortened training time in federated learning as the processing load is distributed between nodes, and the resulting parallelization shortens the model updating cycle. Therefore, the average total training time in federated learning is 58% shorter than in centralized learning.
- Sensitivity analysis can be performed in federated learning. Drawing on controlled alteration of parameters, e.g., number of nodes, participation rate, and number of training rounds, the effect of each parameter is independently measured, evaluating the stability and efficiency of the system under various scenarios.

The remainder of the study is organized as follows: Section 2 reviews related work; Section 3 elaborates on the problem statement and key challenges; Section 4 highlights the main contributions of the study; Section 5 describes the proposed method; Section 6 provides and discusses the results; and Section 7 concludes the work and suggests future directions.

2. Related Work

Several attempts have been reported on improving the performance of intrusion detection systems in various computing settings, e.g., cloud, fog, and, more recently, dew computing in recent years. Given the unique characteristics of dew settings, such as limited resources, unstable connectivity, and the need for real-time responsiveness, it is essential to design lightweight, distributed intrusion detection systems with privacy protection. This section provides a structured review of the literature in three key contexts: (1) intrusion detection systems in dew computing, (2) intrusion detection models in other computing settings, e.g., cloud, fog, and edge computing, and (3) reports on intrusion detection using Edge-IIoTset.

2.1. Intrusion detection systems in dew computing settings

Moussa and Alazzawi [10] proposed a system to detect cyberattacks in the cloud-dew architecture in the automotive IoT. Their model was based on a stacked autoencoder and would transmit raw data from end nodes to the dew server in a centralized structure. Although they sought to address the requirements of dew settings, their model remained dependent on raw data transmission to the dew servers and did not consider a processing distribution at the dew node level. As a result, their system was not aligned with resource limitations and real-time responsiveness in dew settings, and gaps remained to be filled in terms of resource consumption and data privacy.

Singh et al. [7] developed the dew-as-a-service (DaaS) framework to deploy smart intrusion detection systems in the hybrid edge-dew architecture. The framework included the practical implementation of an ML model using the UNSW-NB15 dataset that would be performed in the dew layer and showed comparable performance with other models. Although the model had a lightweight structure that could be deployed within dew settings, processing was performed in a centralized setting in the dew server, with raw data being transmitted from nodes to the central dew server. Moreover, no mechanism was provided to distribute training or narrow the bandwidth, and techniques such as federated learning were not utilized. They provided a major step toward the practical implementation of dew-based intrusion detection systems; however, they did not address the connectivity and resource limitations effectively and paved the way for future work in combining federated learning and lightweight architectures.

Singh et al. [22] proposed a dew–cloud hierarchical federated learning framework for intrusion detection in the Internet of Medical Things (IoMT). Their architecture places learning on distributed dew servers and performs hierarchical aggregation to build a global model without raw data exchange, aiming to improve privacy and availability under unreliable connectivity. The intrusion detection model was based on a hierarchical long short-term memory (LSTM) and was evaluated on TON-IoT and NSL-KDD, reporting strong classification performance. Although the framework aligns with the motivation of dew computing, it relies on recurrent deep models and hierarchical training that can be computationally demanding, while the paper does not clearly quantify dew-node resource requirements or communication overhead under constrained and unstable dew settings. Therefore, further validation is needed to assess feasibility in lightweight, real-time dew deployments.

2.2. Intrusion detection models in cloud, fog, and edge computing

Meng et al. [11] introduced a Bayesian inference-based distributed intrusion detection system to cope with insider attacks in medical smartphone networks. It calculated trust in nodes based on the past experience and direct interactions and reported local results in a semi-centralized setting. Despite satisfactory performance in two real-life hospitals, the framework was limited to medical settings, lacked scalability for industrial or limited-resource scenarios, and did not leverage lightweight deep learning techniques.

Liu et al. [23] developed a model to improve security and reliability in cloud computing services. Drawing on secure virtual machines, network traffic analysis, and policy-based access control, the model addressed security threats, including intrusion attacks. It did not have an ML or deep learning approach and required massive data transmission to the cloud and stable connectivity to a data center. Due to the lack of localized processing and distributed architecture, it is not efficient in settings where computational resources are limited and real-time responsiveness is essential.

Zhao et al. [16] proposed an integrated intrusion detection framework in which a deep belief network (DBN) was employed for feature extraction, whereas a probabilistic neural network (PNN) would be used for classification. Particle swarm optimization (PSO) was utilized to

optimize the architecture, and the results on the KDD Cup 1999 dataset revealed a remarkable improvement in accuracy and speed compared to the basic methods. The model, however, was designed for centralized settings of strong computational resources and was inefficient in applications with limited computational resources or a distributed architecture, e.g., dew computing.
Tian et al. [18] introduced an improved intrusion detection approach based on the DBN in which the optimized PSO would be exploited to tune hyperparameters. The architecture involved several restricted Boltzmann machine (RBM) layers and a support vector machine (SVM) layer for the ultimate classification and outperformed the basic models and similar hybrid frameworks to the KDD Cup 1999 dataset. However, it was based on centralized processing and had no distributed mechanism or compatibility with settings of limited and unstable resources, such as dew computing.

2.3. Reports on the Edge-IIoTset

2.3.1. Centralized learning

Tareq et al. [24] compared the ToN-IoT, UNSW-NB15, and Edge-IIoTset datasets for use in deep learning to detect cyberattacks. Recurrent neural network (RNN), LSTM, and gated recurrent unit (GRU) models were compared within centralized learning settings on Edge-IIoTset, with GRU showing higher performance in accuracy and complexity. All models were dependent on centralized processing and full raw data transmission to the central server and lacked compatibility with distributed settings with limited resources, e.g., dew computing. Ding et al. [25] proposed the DeepAK-IoT model for cyberattack detection in IoT as a combination of the LSTM and GRU algorithms developed in a centralized setting. It was tested on three datasets, including Edge-IIoTset, and showed higher performance than the basic models. However, due to its full dependence on centralized processing and raw data transmission to the central server, the framework would be ineffective in distributed settings with limited resources, such as dew computing.
Konatham et al. [19] introduced an integrated technique for anomaly detection in IIoT networks. They combined a convolutional neural network (CNN) and an LSTM architecture to simultaneously leverage spatial features and temporal features. It was trained with the entire data of Edge-IIoTset in a centralized setting and outperformed single-architecture models. However, dependence on full data transmission and high computational resource demand remained challenges for edge or dew settings with limited resources. Sadhwani et al. [26] proposed the SmartSentry framework to analyze cyber threats in the IIoT. It exploited centralized deep learning and Edge-IIoTset for advanced and accurate attack detection. In spite of its satisfactory performance, the framework was dependent on centralized processing and full raw data transmission to the central server and did not support distributed implementation or compatibility with limited-resource settings such as dew computing.
Kilichev et al. [20] developed an integrated intrusion detection framework in IoT-based electric vehicle charging systems (EVCSs). It combined 1D-CNN, LSTM, and GRU in order to simultaneously analyze spatiotemporal features. The model was trained using Edge-IIoTset in a centralized setting and showed superb accuracy in detecting a broad range of attacks. However, dependence on centralized processing and full raw data transmission, along with extremely high computational costs, due to the simultaneous utilization of three deep architectures remained barriers to its implementation in lightweight and limited-resource settings, e.g., dew computing. However, the model showed excellent performance in terms of accuracy.
Laiq et al. [27] developed a distributed denial-of-service (DDoS) attack detection system via classical ML algorithms with ensemble (stacking/voting/bagging/AdaBoost/XGBoost) designs in Edge-IIoTset and data balancing through the synthetic minority over-sampling technique

(SMOTE). It was evaluated on Edge-IIoTset. Training was centralized, and remarkably high accuracy rates were reported. However, data heterogeneity, model lightweighting, and communication cost calculation constraints in dew setting were not addressed, and federated learning and evaluation on limited-resource hardware were not incorporated. Kumar et al. [28] proposed the PETDA2C-EC framework for privacy attack detection in Edge-IoT. The model included hybrid feature selection – e.g., random forest (RF), MI, and chi-square – and several classical ML classifiers and was tested on UNSW-NB15, CIC-IDS2017, ToN-IoT, and Edge-IIoTset in a centralized setting, with an accuracy rate above 99% being reported. However, the scope of the study is limited to privacy attack detection in a centralized setup, and it does not consider federated learning, lightweight architectures, or communication cost analysis. Moreover, the model is not experimentally evaluated on fog or dew nodes.

2.3.2. Federated learning

Thamar et al. [29] proposed a federated learning–based system using the FedAvg aggregation algorithm for anomaly detection in industrial IoT edge networks. By avoiding raw data exchange among clients, the framework preserves privacy. The model was designed for binary classification, implemented with fully connected layers, and tested on the Edge-IIoTset dataset. However, it lacks a lightweight architecture or optimization for the dew layer, leaving its applicability to multi-class scenarios and resource-constrained environments limited. Rashid et al. [30] developed a federated learning-based approach using the FedAvg algorithm for intrusion detection in the IIoT. It used Edge-IIoTset with no raw data exchange and would protect data privacy. The model was a simple, fully connected architecture and lacked more advanced structures, e.g., CNN or LSTM. Moreover, aspects such as the communication overhead, localized resource consumption, and compatibility with lightweight and limited-resource settings, e.g., dew computing, were not addressed.
Popoola et al. [31] introduced a deep federated learning-based framework for intrusion detection in the edge settings of the IoT. It covered three different classification scenarios by utilizing Edge-IIoTset with no raw data exchange. The model was trained locally in edge nodes, and only the weights were transmitted to the central server to protect data privacy. This approach is considerable in terms of scenario diversity and relative compatibility with edge nodes; however, heterogeneous data management was not addressed, and no lightweight architecture or optimization module was included for limited-resource settings such as dew computing. As a result, despite its major contribution to leveraging federated learning for IoT security, further optimization is required in lightweight and distributed settings for practical application.

Table 1. Summary of centralized and distributed learning models for intrusion detection in IIoT

Ref.	Year	Model	Learning	Dataset	Advantages	Disadvantages (dew settings)
[11]	2017	Bayesian inference	Semi-centralized	Real-life data from two hospitals	Robust under insider attacks, tested in real-life medical settings	Not generalizable to industrial settings, lack of deep learning
[23]	2017	Cloud security with traffic analysis and policy control	No learning	Conceptual/ simulation	High reliability, advanced access control, security traffic analysis	Stable connectivity demand, full data transmission, lack of localized processing
[16]	2017	DBN+PSO-PNN	Centralized	KDD Cup 1999	Optimized combination of DBN with accurate	Dependence on strong resources,

Ref.	Year	Model	Learning	Dataset	Advantages	Disadvantages (dew settings)
					classification, reduced data dimensionality, high accuracy	full data transmission
[10]	2020	SAE (Autoencoder)	Centralized	NSL-KDD	High accuracy	High bandwidth consumption, distributed processing not supported, full data transmission to the dew server
[18]	2020	DBN + improved PSO + SVM	Centralized	KDD Cup 1999	High accuracy, hyperparameter optimization, accelerated convergence, reduced error rate	Distributed processing not supported, stable connectivity demand, full data transmission
[7]	2021	DBN	Centralized	UNSW-NB15	Practical implementation in dew settings, serviceable infrastructure design	Full data transmission to the dew server, distributed processing not supported, high bandwidth consumption
[24]	2022	LSTM + RNN + GRU	Centralized	Edge-IIoTset, ToN-IoT, UNSW-NB15	Comparison of deep models, comprehensive data analysis, practical evaluation	Distributed and lightweight processing not supported, full data transmission
[22]	2023	HLSTM	Federated	ToN-IoT, NSL-KDD	Hierarchical aggregation, strong classification performance	Dependence on strong resources, high bandwidth consumption
[25]	2023	GRU + LSTM	Centralized	Edge-IIoTset, ToN-IoT, UNSW-NB15	Particular deep model design, simultaneous leverage of LSTM and GRU, high accuracy	Distributed processing not supported, dependence on strong resources, high bandwidth consumption, full data transmission
[29]	2023	Unsupervised deep autoencoder	Distributed	Edge-IIoTset	Privacy, no raw data transmission, non-labeled design	Evaluated solely in binary applications, lack of lightweight or optimized architecture
[30]	2023	DNN	Distributed	Edge-IIoTset	Privacy, no raw data transmission, heterogeneous data supported	Lack of lightweight or optimized architecture, dependence on strong resources
[19]	2024	LSTM + CNN	Centralized	Edge-IIoTset	Combined advantages of CNNs and LSTM, enhanced accuracy in anomaly detection	Distributed processing not supported, dependence on strong resources, full data transmission

Ref.	Year	Model	Learning	Dataset	Advantages	Disadvantages (dew settings)
[26]	2024	DNN	Centralized	Edge-IIoTset	Comprehensive framework design, multidimensional threat analysis, satisfactory classification accuracy	Distributed processing not supported, stable connectivity demand, full data transmission
[20]	2024	CNN + LSTM + GRU	Centralized	Edge-IIoTset	High accuracy in complex attack detection, integrated spatiotemporal architecture	Lack of lightweight or optimized architecture, dependence on strong resources, full data transmission
[31]	2024	DNN	Distributed	Edge-IIoTset	Privacy, evaluation in three classification scenarios (2, 6, and 15 classes)	Lack of lightweight or optimized architecture, dependence on strong resources
[27]	2025	ML – classical ensemble	Centralized	Edge-IIoTset (DDoS scenarios)	Extremely high accuracy, regular ensemble comparison to basic models, balanced classes (SMOTE)	Distributed processing not supported, dependence on strong resources, full data transmission, DDoS scenarios only
[28]	2025	Classical ML + hybrid FS	Centralized	UNSW-NB15 ‘ CIC-IDS2017, ToN-IoT, Edge-IIoTset	High accuracy, novel method for feature selection	Distributed processing not supported, dependence on strong resources, full data transmission, privacy attack scenarios only

3. Problem Statement

Dew computing is the closest layer to the data generation sources (sensors and actuators) and plays a key role in real-time processing and decreasing computational dependence on the Internet [3,4]. However, the intrinsic characteristics of the dew layer, including limited computational resources, low bandwidth, unstable connectivity, and major geographical distribution, expose dew computing to growing security threats [5,7]. In this respect, an intrusion detection system with three simultaneous key characteristics, i.e., lightweight architecture, lightweight computations, and distributed processing, remains to be developed to detect threats locally in real-time and prevent their propagation in multi-layer architectures without raw data transmission to the upper layers. In this study, lightweight architecture is assessed in terms of (i) the number of trainable parameters, (ii) the final model size, and (iii) the inference time, which together indicate whether the model can be practically deployed on dew-layer machines with constrained resources. In addition, lightweight computations refer to limiting both the local processing cost and the communication overhead in federated learning, characterized by (i) a bounded runtime and memory footprint during local updates and aggregation per communication round, and (ii) reduced model-update traffic per communication round, compared with centralized raw traffic data transfer and heavier federated designs. These criteria are later quantified and reported in the experimental section. A review of the literature shows that earlier works mostly did not effectively address such challenges in dew computing settings since:

- They mainly focused on cloud-based systems, incompatible with limited resources and real-time requirements in dew and fog layers [32,33].
- They employed deep learning frameworks with heavy architectures requiring high computational power and incompatibility with dew nodes [34–36].
- In a number of federated approaches (e.g., Popoolar et al. [31]), the processing and communication overheads increased internal bandwidth consumption and latency, which is inconsistent with the functional philosophy of dew computing [37–39].
- Industrial data are imbalanced and non-numerical in practice and contain missing or redundant points, diminishing performance in conventional methods [40–43].

The negligence of such challenges may turn the dew layer into a critical disadvantage in the security chain of the IoT as the compromise of dew nodes may propagate to upper layers, corrupt the basic data, and disturb high-level decision-making [10,44]. Moreover, the lack of an effective defense mechanism can facilitate major attacks, such as DDoS, malware, and injection attacks. Three approaches can be adopted in order to cope with these challenges:

(1) Development of lightweight models that can be implemented in dew nodes in order to decrease computational resource consumption and maintain efficiency in limited-resource settings [45–47].
(2) Utilization of smart distributed architectures to balance loads and boost resilience, focused on collaboration between the nodes and computational layers to improve scalability and reduce points of failure [33,48–50].
(3) Federated learning is a novel approach providing a combination of the two above ideas: lightweighting computations at the local level and node collaboration without a need for raw data exchange [30,31,51]. This prevents direct data transmission and, therefore, minimizes the risk of information leakage. However, many methods in the federated learning realm encounter challenges such as high communication overhead, delayed convergence, and limited compatibility with weaker nodes in low-resource settings [37,52,53]. As a result, it is essential to develop optimized and lightweight versions of federated learning for fog and dew layers.

This study focuses on designing a lightweight federated intrusion detection system based on 1D-CNNs to detect threats at the dew layer without transmitting raw data and maintain stable learning under variable communication conditions. The goal is to develop a fully dew-layer federated IDS framework that ensures the minimum communication overhead, a short training time, and high accuracy in real-life industrial settings.

4. Contributions of the Proposed Model

This study proposes a novel and lightweight framework for intrusion detection in dew computing settings focused on leveraging the capabilities of 1D-CNNS in extracting temporal and statistical patterns and the advantages of federated learning in minimizing raw data exchange and protecting data privacy. The main novelty of this study lies in devising a shallow and localized model based on a single 1D convolutional layer that is particularly tailored for limited resources and unstable connectivity. In contrast to earlier methods that transmitted data to the central server with major communication loads and security risks, the proposed framework shows higher compatibility with limited hardware as it includes merely model parameters and has a lightweight design. In addition, compared to methods based on heavy networks, such as deep belief networks (DBNs) or classical architectures, such as a multilayer

perceptron (MLP), the proposed model leverages local filters and low-complexity convolutions, enabling the accurate extraction of attack patterns at low computational costs.
The evaluation of three independent classification scenarios (binary, six-class, and fifteen-class) based on Edge-IIoTset, which is structurally complex and homogeneous in terms of labeling, represents another advantage of the proposed framework [21]. The utilization of federated learning in these scenarios with the proposed model enables a comprehensive evaluation of the model under real-life conditions. Overall, the proposed method not only significantly decreases bandwidth consumption and eliminates raw data transmission but also meets the security requirements, resource limitations, and functional requirements in the dew computing layer. Furthermore, the proposed model maintains a lightweight design without compromising detection quality, and provides a robust balance between efficiency and effectiveness compared with prior heavier deep architectures on Edge-IIoTset, such as the CNN+LSTM model in [19] and the CNN+LSTM+GRU model in [20].

5. The Proposed Model: Technical Discussion

Conventional ML training techniques are either dependent on the upper layers (e.g., fog or cloud) or are incompatible with dew layer limitations due to communication overheads and high latency, representing a major challenge. Therefore, it is crucial to develop a model that enables distributed and lightweight training performed fully in the dew layer without a dependence on the upper infrastructures. Thus, this study developed a framework based on the fully dew-layer federated learning architecture where both the aggregation server and all nodes are deployed in the dew layer. This structure is inspired by the single-super-hybrid-peer (SSHP) model in peer-to-peer (P2P) networks [4]. In this setup, a central node with effective super-peer processing serves as the federated aggregation server, collects model parameters from the other nodes, and redistributes the updated version, while the other nodes (hybrid peers) implement localized data training (Fig. 1).
The main advantage of this approach is that the training process can be fully performed in the dew layer or even offline with no dependence on the upper layers (fog or cloud). This substantially decreases latency, bandwidth consumption, and connection dependence. In this architecture, the dew servers typically include personal computers (PCs), workstations, or local organizational servers that are deployed in user sites and are equipped with sufficient processing, storage, and software resources to execute programs [4]. This study assumes that such servers have no energy consumption constraints and can perform ML computations with no concerns regarding power consumption.

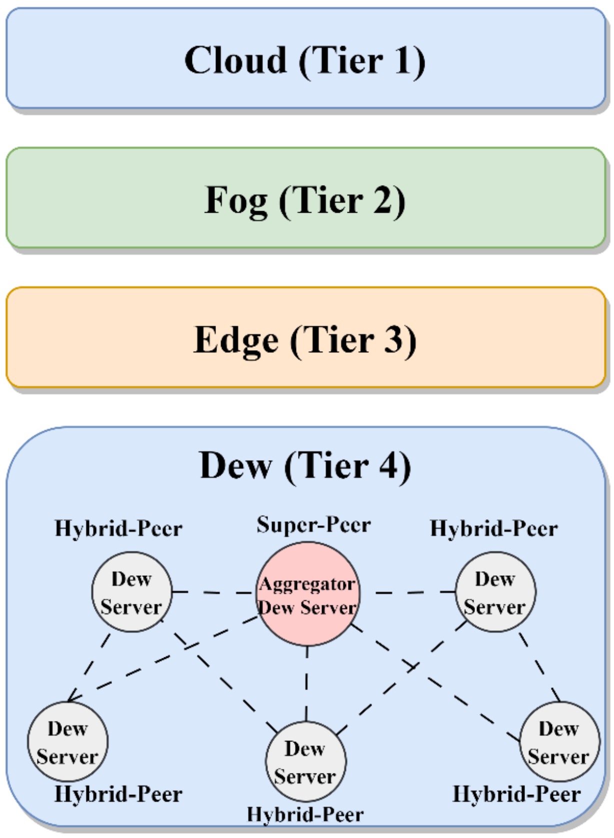


Fig. 1. Proposed fully dew-layer federated learning architecture

To adapt to the hardware limitations of dew servers, a lightweight model was designed and evaluated based on a single 1D convolutional layer (Conv1D). The design process included data preprocessing, 1D-CNN architecture development, and implementation in centralized and federated settings. First, the model was executed in a centralized setting to provide a benchmark for performance evaluation so that the results could be compared to federated learning results and earlier works. This study mainly focused on deploying the model in the federated learning setting, in which an effective aggregation algorithm would be chosen for each classification scenario based on multi-objective optimization, i.e., F1-score (macro), communication cost, and convergence rate. In addition, fifteen independent scenarios were designed and simulated to comprehensively evaluate model performance and measure the sensitivity of the learning process to key factors; the number of nodes, participation rate, and the number of training rounds were altered to systematically evaluate the contribution of each parameter to the ultimate performance of the model.

5.1. Centralized learning

The training process was performed in a centralized setting. Fig. 2 depicts a flowchart of the phases, from data preparation to ultimate model training.

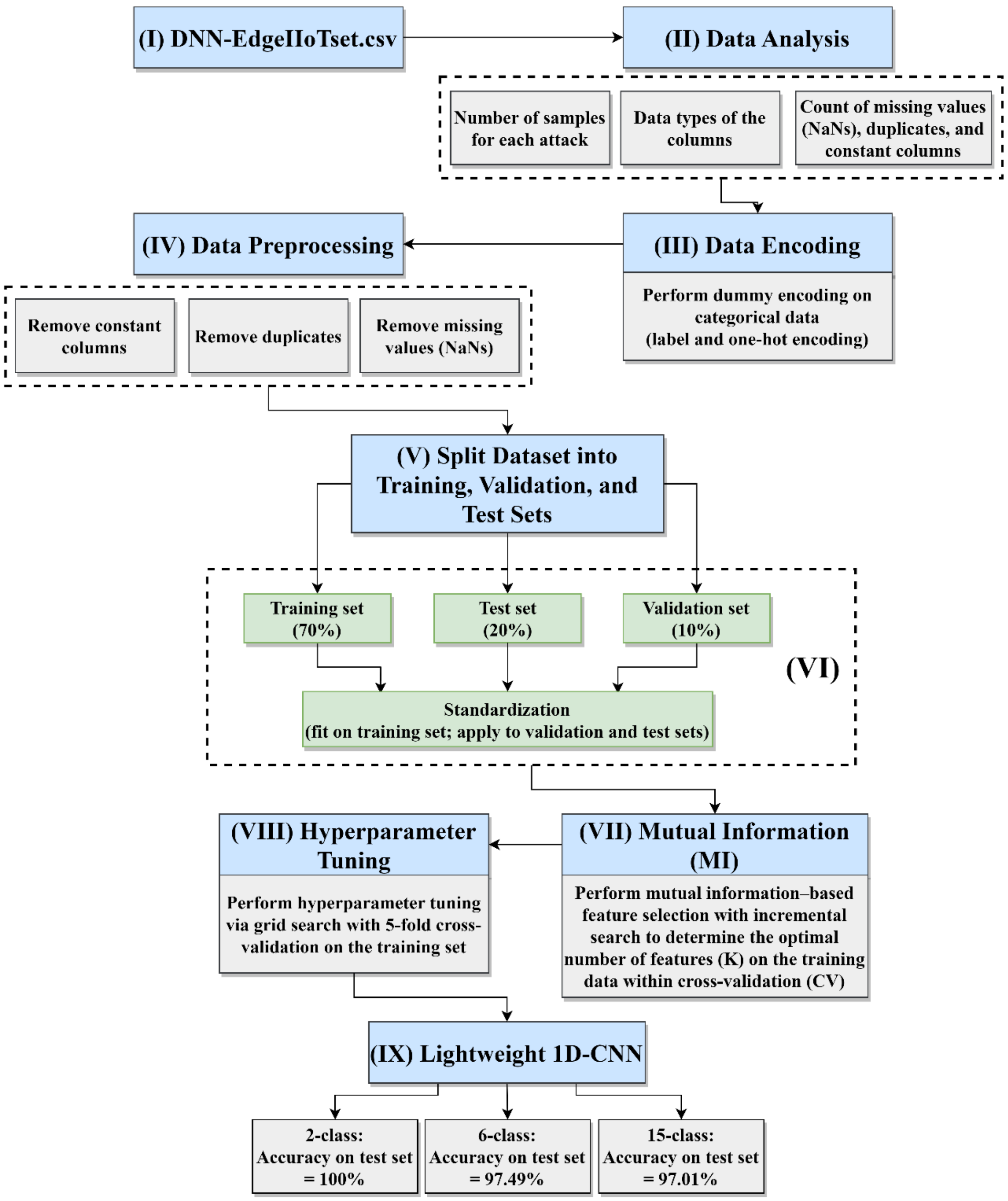

(I) DNN-EdgeIIoTset.csv
(II) Data Analysis
Number of samples for each attack
Data types of the columns
Count of missing values (NaNs), duplicates, and constant columns
(III) Data Encoding
Perform dummy encoding on categorical data (label and one-hot encoding)
(IV) Data Preprocessing
Remove constant columns
Remove duplicates
Remove missing values (NaNs)
(V) Split Dataset into Training, Validation, and Test Sets
Training set (70%)
Test set (20%)
Validation set (10%)
Standardization (fit on training set; apply to validation and test sets)
(VI)
(VII) Mutual Information (MI)
Perform mutual information–based feature selection with incremental search to determine the optimal number of features (K) on the training data within cross-validation (CV)
(VIII) Hyperparameter Tuning
Perform hyperparameter tuning via grid search with 5-fold cross-validation on the training set
(IX) Lightweight 1D-CNN
2-class: Accuracy on test set = 100%
6-class: Accuracy on test set = 97.49%
15-class: Accuracy on test set = 97.01%

Fig. 2. Flowchart of centralized learning with feature selection and hyperparameter tuning in 1D-CNN

I. Loading the DNN-Edge-IIoTset

The DNN-Edge-IIoTset, including raw data of normal traffic and various attacks in the IIoT, was loaded [21]. It contains 1,909,671 records with 61 features and was selected in light of its comprehensive features, diverse attacks, and compatibility with federated learning scenarios and dew computing settings. Table 2 lists the main features in the DNN-Edge-IIoTset.

Table 2. Main feature categories in the Edge-IIoTset

Category	Example Features	Description
Flow-based	Flow Duration, Total Packets	Basic traffic volume over each flow
Time-based	Inter-arrival Time Mean, Flow Start Time	Temporal characteristics of packet streams
Statistical	Avg. Packet Size, Std. Dev. of Length	Distribution and variability of packet sizes
Protocol/Flags	SYN Flag, ACK Flag, Window Size	TCP/IP protocol behavior
System/Behavioral	Alert Count, Header Length	System-level or behavioral indicators

II. Data analysis

The data quality and structure of the DNN-Edge-IIoTset were analyzed. The distribution of samples in various classes was evaluated to identify the balance between normal data and attacks. Then, the type of data in each column was investigated to separate the numerical columns from textual and class columns and determine whether they needed to be coded. The missing points, duplicate records, and constant columns were assessed to identify low-quality data. This directly copes with a challenge highlighted in the problem statement, i.e., imbalanced and noisy industrial data. Identifying and addressing such challenges before the processing stage helps avoid training the model based on incomplete or biased data and enhances the quality of input features in the next stages.

III. Data encoding

Once the data of the Edge-IIoTset had been analyzed, textual and non-categorical features were converted into a numerical format so that they could be processed by the model. Two common techniques, i.e., label encoding and one-hot encoding, were used for this purpose [54,55]. The former maps each textual value onto a unique numerical label in the range of 0 to x (x=number of features – 1). This is an effective approach for compositional data or data with limited values, even though a false order may be induced in nominal data. The latter, on the other hand, changes values into a binary vector with elements of 0 or 1, which is efficient for nominal protocols and fields since it does not induce false order dependence.

This phase is crucial as industrial data often contains heterogeneous values, e.g., protocols, addresses, and textual identifiers, whose direct utilization would lead to errors or reduced model accuracy. The combined utilization of label and one-hot encoding techniques enables a uniform and significant data representation and handles the challenge of non-numerical and heterogeneous data in industrial settings [56].

IV. Data preprocessing

The data were preprocessed to ensure input quality. Duplicate records were eliminated by detecting entirely identical rows while excluding missing points (NaNs). To detect constant columns or columns of similar patterns, the content of the columns was evaluated through MD5 hash comparisons, excluding non-distinct columns. Furthermore, non-useful columns, e.g., IP addresses and pack schedules, were excluded since they had no analytical value for model training. This eliminated noise and incompatibility from the data and enabled more accurate model training in the next stages.

V. Dataset split into training, validation, and testing subsets

The dataset was split into a training subset (70%), a validation subset (10%), and a testing subset (20%) to ensure a proper balance between training efficiency, hyperparameter tuning, and final evaluation. This ratio follows the recommendation of Géron [56], which provides a well-established guideline for maintaining representativeness while preventing data leakage. The training data were used to learn model parameters, the validation data were employed to tune hyperparameters and prevent overfitting, and the testing data were utilized to evaluate the model's generalization capability to unseen data.

VI. Standardization

The features were standardized using Standard Scaler to ensure a mean of 0 and a standard deviation of 1. Standardization was applied to merely the testing data, and the same scaling was implemented on the validation and testing data to prevent information leakage. The standardization phase is crucial to address the scale heterogeneity of industrial data since such data are obtained from various sources and sensors with various numerical ranges. Géron [56] held that feature scaling prevents biases toward variables of larger scales and maintains model accuracy and stability under operational conditions.

VII. MI

This study used the MI method for feature selection. MI measures the relative importance of each feature by evaluating its dependence on the output label [57]. In this study, MI was estimated using a nonparametric k-nearest neighbor entropy estimator based on neighbor distances, which is suitable for continuous variables and mixed discrete–continuous features [58,59]. The optimal number of features K was set through incremental search [60]. This process excludes insignificant features and reduces data dimensionality, tackling two key challenges in IIoT and dew computing: (1) limited computational and storage resources and (2) complexity of multidimensional data, which may lead to overfitting. As a result, the final model would be more efficient, faster, and more stable, and could be more effectively implemented in the dew layer.

VIII: Hyperparameter tuning

The hyperparameters were tuned using a grid search and five-fold cross-validation on the testing data. This would ensure the selection of optimal values for the model parameters and address the performance optimization challenge in the dew setting with computational limitations. The optimal tuning of hyperparameters boosts model accuracy and speed and decreases resource consumption, while cross-validation evaluates performance stability in various data scenarios and minimizes the risk of overfitting.

IX. Lightweight 1D-CNN

The model proposed in this study is a simplified and optimized one-dimensional convolutional neural network (1D-CNN) designed to achieve maximum efficiency under the resource constraints of dew computing environments. As shown in Fig. 3, the proposed architecture comprises a linear and efficient processing path, where the feature vector of each network flow (variables such as the pack exchange rate, flow length, pack size, number of ports, and protocol flags) with a length of L (number of features selected) and input channel $\mathrm{C}_{in} = 1$ is fed as a 1D sequence to the Conv1D layer. The input channel determines that each record is represented as solely a single-channel vector of features (unlike multidimensional data of multiple channels). The convolution operation is applied to the feature axis to identify local patterns and short-range relationships between them. The network security data in the Edge-IIoTset, which have a vector and sequential format, are used in a numerical and structured form for learning.

In contrast to earlier models in the literature with multiple convolutional layers with heavy combinations (such as a CNN+LSTM+GRU model [20]), the proposed architecture comprises a single Conv1D layer. The number of parameters in the Conv1D layer is calculated as:

$$\#Params_{Conv1D} = F(K \times C_{in} + 1) \tag{1}$$

where F is the number of filters in the convolutional layer, K is the kernel size to cover the sequence of network flow features and identify local patterns, and C_{in} is the number of input channels (i.e., the single-channel feature vector of each network flow in the Edge-IIoTset). Furthermore, the parameters of the fully connected (dense) layer are written as:

$$\#Params_{Dense} = H_{in} \times H_{out} + H_{out} \tag{2}$$

where H_{in} is the number of input neurons in the dense layer (the flattened output of the convolution layer), and H_{out} is the number of output neurons (i.e., the number of classes).

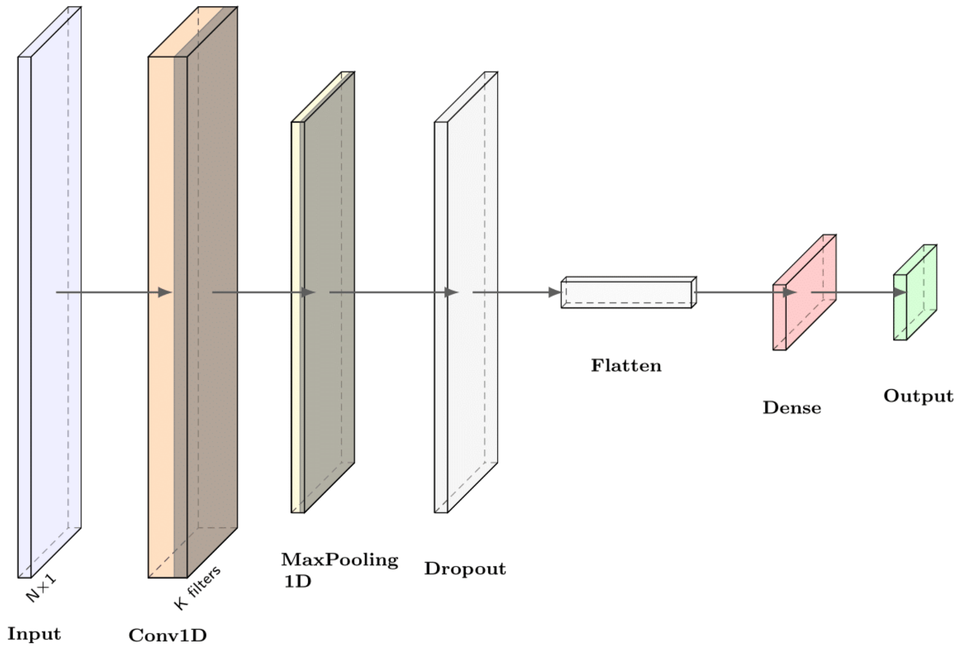


Fig. 3. The Proposed 1D-CNN Architecture

As a result, the total number of parameters in the proposed model is obtained as:

$$\#Params_{Total} = \#Params_{Conv1D} + \#Params_{Dense} \tag{3}$$

This variable increases exponentially in multilayer models, requiring further memory and computational power. In contrast, a single Conv1D layer, a small hidden dense layer, and an output dense layer ensure that the total number of parameters remains in a limited range. This allows for implementing the model in low-resource dew servers and, simultaneously, helps use the memory more efficiently through sparse interactions and parameter sharing in the convolution layer [61].

Regarding time complexity, the main operation in Conv1D is of order $O(F \times K \times C_{in} \times L)$, where L denotes the length of the input feature vector (the number of features selected for each network flow). As the proposed architecture has a single-channel input ($C_{in} = 1$) and a short input feature vector (10-40, depending on the scenario), the computational cost of the Conv1D layer remains remarkably low. The complexity of the fully connected (dense) layer is of order $O(H_{in} \times H_{out})$, leading to a small computational overhead due to the small number of classes and small flattened output. Therefore, the total time complexity of the model remains at the linear level, unlike deeper or combined models (CNN+LSTM/GRU) with exponentially growing time complexity.

From a modeling capacity perspective, the Edge-IIoTset data mainly comprise short-term and repetitive local patterns, as with sudden increases in the packet rates in DDoS attacks or abnormal sequences of TCP flags in scanning attacks. However, some attack types such as MitM do not always manifest through strong short-term statistical deviations. MitM attacks typically involve traffic interception, modification, or replay, which may span longer interaction sequences and exhibit weaker local anomalies at the flow level. In the Edge-IIoTset, MitM samples are relatively limited and are mainly represented through aggregated flow features, which restricts the diversity of observable MitM behaviors.

The Conv1D layer with small kernels can directly detect such local motifs without deeper layers to learn complex hierarchies. In addition, recent theoretical research has shown that CNNs can approximate a broad range of functions through an optimized width (number of filters) and kernel size, even at low depth [62]. Hence, despite its simplicity, the proposed architecture can effectively model complex attacks. The incorporation of MaxPooling1D and Dropout into the proposed architecture also plays a key role in performance improvement. MaxPooling1D enables more stable and robust representation by reducing the size of intermediate data and creating invariance to small shifts [63,64]. In this setup, the convolution axis corresponds to feature sequencing or packet sub-sequencing, and MaxPooling suppresses the noise of highly variable features and produces more robust representations by reducing local resolution and selecting dominant responses.

Moreover, Dropout minimizes the overfitting risk and enhances generalizability by addressing excessive inter-neuron dependencies and decreasing the effective network capacity [63,65]. This is, in particular, essential for the imbalanced Edge-IIoTset, in which frequent classes, e.g., DDoS and Scanning, account for a major portion of the data, whereas classes such as Injection and Malware have a small number of samples. Deep architectures often tend to focus solely on dominant patterns and neglect minority classes.

The proposed model, however, tackles this challenge by integrating the shallow Conv1D with MaxPooling and Dropout. MaxPooling highlights distinct features by compressing the intermediate data and mitigates the dependence on the number of samples, while Dropout drives the model to utilize a diverse combination of neurons and prevents an excessive focus on particular patterns of the most frequent classes. Thus, learning is not limited to the majority classes, and the model can focus on more general and discriminative features, leading to improved detection rates of rare attacks without a drop in the overall accuracy.

Ultimately, the lightweight and low-depth architecture of the proposed model shortens the inference path and enables real-time detection. This is crucial in attack scenarios such as DDoS attacks since the model can detect attacks before the full saturation of the bandwidth or processor resources. In light of its flexible design, the proposed framework can also be extended from a binary setting (normal/attack) to multiclass settings (six or fifteen) only by altering the number of neurons within the output layer and activation function, without major alterations in the architecture.

This study used the binary cross-entropy (BCE) cost function for the binary scenario and the sparse categorical cross-entropy (SCCE) cost function for the multiclass scenario. These cost functions are completely aligned with the nature of problems and the type of labels in the Edge-IIoTset and improve model performance in several terms. In the binary scenario, the detection is aimed at realizing whether each sample belongs to the “attack” or “normal traffic” classes. The output of the final layer is a scalar in the range of [0, 1], which would be obtained from the sigmoid function and represent the membership likelihood of a given sample in the “attack” class. In such a case, the BCE cost function is the best alternative for optimization since it directly compares the predicted likelihood to the real value:

$$BCE(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \tag{4}$$

where N is the total number of samples in the Edge-IIoTset before preprocessing, y_i is the real label of sample i, which is 1 for the attack class and 0 for the normal traffic class, and $\hat{y}_i$ is the predicted membership likelihood of sample i for the attack class.

The logarithmic character of the BCE cost function is a key advantage that ensures a greater penalty for false predictions in the minority classes (e.g., rare attacks). This is particularly essential in the imbalanced Edge-IIoTset since the ratio of normal traffic samples to attack

samples is significantly large, and the BCE cost function effectively prevents the negligence of minority classes in the learning process.
In the multiclass scenarios (six and fifteen classes), the labels are stored as integers rather than one-hot vectors. Therefore, the SCCE cost function was employed to train the model:

$$SCCE(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} \log(\hat{y}_{i,y_i}) \tag{5}$$

where N is the total number of samples in the Edge-IIoTset before preprocessing, y_i is the integer label of sample i, and $\hat{y}_{i,y_i}$ is the predicted membership likelihood of integer class y_i, which is obtained from the output of the softmax function.
Reduced memory consumption and accelerated training represent the first advantage of the SCCE cost function since it is no longer essential to convert labels into one-hot vectors. This is a key advantage in the fifteen-class scenario in the limited-source dew setting. The alignment of SCCE with the output of the softmax function and the generation of significant gradients for the minority classes represent the second advantage. This, along with the Dropout/Pooling, was helpful in improving the detection of rare classes in this study. The stability of SCCE in handling imbalanced data is the third advantage since false predictions for minority classes (e.g., Injection or Malware) are subject to a larger penalty, boosting the detection rate of rare attacks without a drop in the overall accuracy of the model in frequent classes.
This study purposefully employed the adaptive moment estimation (ADAM) optimizer to train the 1D-CNN model. The ADAM algorithm provides a combination of the advantages of the Momentum and RMSProp models and calculates the learning rate of a parameter based on two exponential average estimations. The first- and second-order exponential averages are updated as:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{6}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{7}$$

where g_t denotes the error in detecting network attacks with respect to the weights of the Conv1D layer and fully connected layer, β_1 is the reduction factor of first-order exponential averaging, which is used to smoothen sharp gradient shifts in highly dynamic networks (e.g., DDoS attack-induced bursts), β_2 is the second-order averaging factor, which ensures the stability and accuracy of weight updates by controlling the variation ranges of features with heterogeneous scales (e.g., high packet rates versus binary protocol flags), θ represents the weights of the Conv1D filters and dense neurons, which, in practice, represents attacks and normal traffic versus the network flow feature vector, m_t is the first-order exponential gradient average, which serves as a stabilizer to update weights such that error variations in stable features (e.g., packet rate or repetitive protocol flags) are enhanced and the learning path of the model is facilitated, and v_t is the second-order exponential gradient average and ensures that the learning rate remains higher for more stable features.
These values are corrected through Eqs. (8) and (9) to eliminate the initial bias, and the weights are updated based on Eq. (10):

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{8}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{9}$$

$$\theta_{t+1} = \theta_t - \alpha \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{10}$$

where α is the learning rate and plays a crucial role in setting the convergence rate of the model in dew settings, while ϵ is a small quantity to avoid division by zero and particularly prevents computational errors when gradients become extremely small in minority data (e.g., rare Injection and Malware attacks).

The Momentum component prevents extreme variations in the learning path based on m_t, while RMSProp regulates the learning rate of each parameter independently based on v_t. This optimizer is essential for the heterogeneous and multidimensional Edge-IIoTset data. The Edge-IIoTset comprises a variety of features, e.g., packet rates and protocol flags, and the ADAM algorithm takes more optimal steps for weight updating by automatically regulating the learning rate. Since the Edge-IIoTset is highly imbalanced, ADAM regulates the weights of minority classes (e.g., Injection and Malware attacks) more effectively in a shorter time without a drop in model accuracy for frequent classes. Thus, the model enjoys a balanced capability of learning for all classes.

Furthermore, in dew computing settings with limited computational resources and training time, ADAM decreases energy consumption and shortens the computational time due to its high convergence rate and lower demand for training iterations (epochs). Exponential gradient averaging is another characteristic of the ADAM algorithm, which provides robustness to the noise and variability of attack patterns and prevents trapping in local minima. Most importantly, ADAM's dynamic and automatic regulation of the learning rate eliminates the need for an extensive manual search to find the optimal value, which, in turn, accelerates and facilitates the training process.

5.2. Federated learning

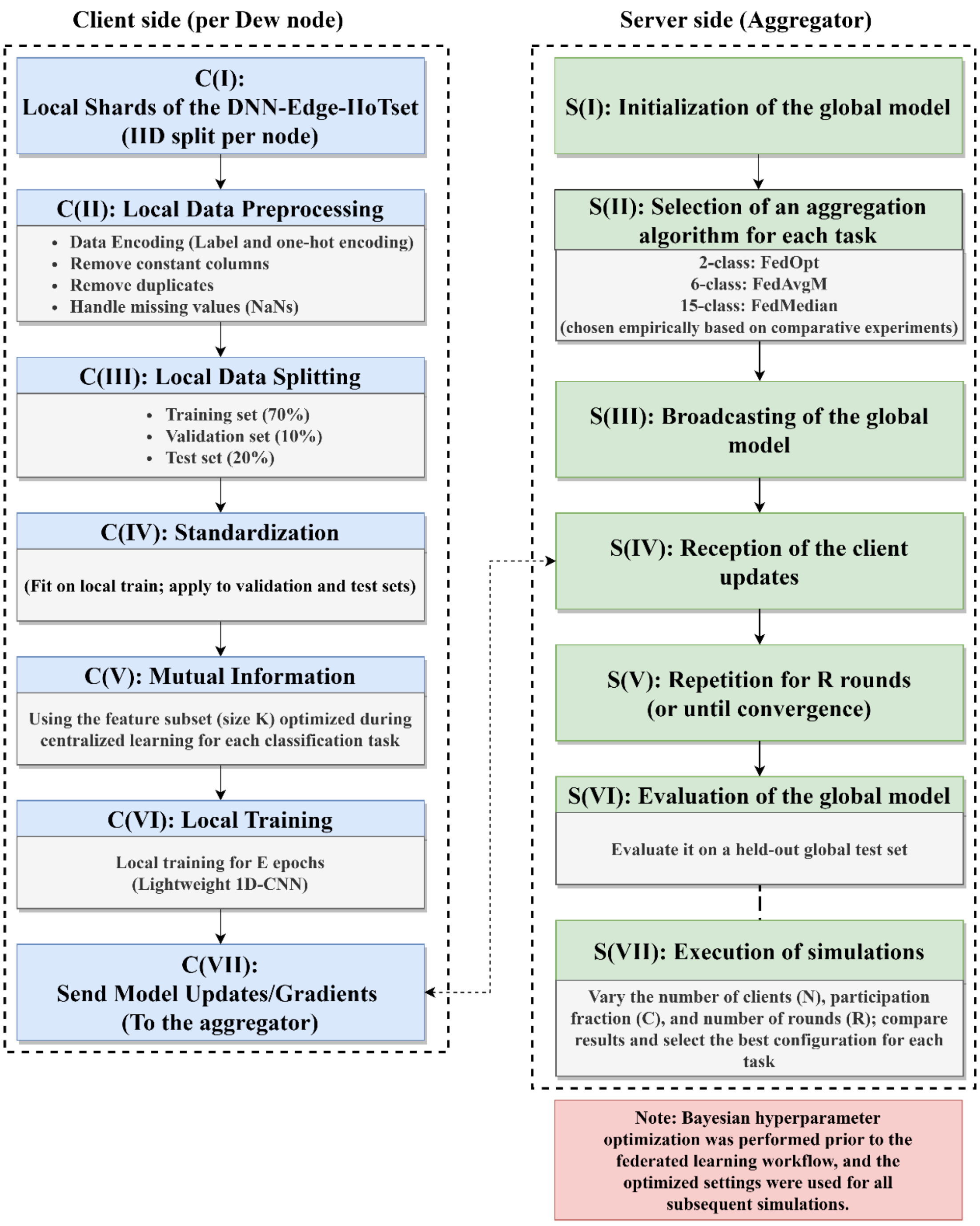
Fig. 4 illustrates the flowchart of federated learning in the dew layer, organized into the client side (C) and server side (S). The former implements local processing and training on the data of each dew node, while the latter is responsible for coordination, aggregating updates, and managing learning cycles between nodes.

The client-side federated learning process begins with Stage C(I), in which the Edge-IIoTset is split as independent and identically distributed (IID) local shards between dew nodes so that each node contains data of the same statistical distribution. Stage C(II) performs local preprocessing, including manual data coding through label encoding and one-hot encoding, eliminating constant columns, excluding duplicate records, and managing missing values (NaNs). Stage C(III) splits the local data of each node into a training subset (70%), a testing subset (20%), and a validation subset (10%) to ensure that the model can be generalized to new data [56]. Stage (IV) standardizes the data through Standard-Scaler by calculating the scaling parameters merely based on the training data before they are applied to the testing and validation data. Stage C(V) performs feature selection using the MI approach.

Drawing on gradual searches, it applies the best number of features K, which is previously optimized for each problem in centralized learning, to each node. Stage C(VI) trains the lightweight 1D-CNN model for a given number of epochs (determined through Bayesian optimization based on each classification problem) on local data. Finally, in Stage C(VII), each node sends only updated gradients or weights to the aggregation server rather than transmitting raw data.

The server side in federated learning is responsible for coordinating and aggregating the knowledge of local models and plays a key role in the effective and efficient convergence of the final model. The hyper-parameters of the local models are optimized through Bayesian optimization before beginning distributed training in order to ensure that each node starts training with the most effective configuration.

This is also theoretically justifiable since negligence to optimize hyper-parameters at the local level in distributed architectures may lead to decelerated convergence, increased variance between local models, and reduced global model quality. The server-side stages of federated learning are described below.



Client side (per Dew node)
Server side (Aggregator)
C(I): Local Shards of the DNN-Edge-IIoTset (IID split per node)
C(II): Local Data Preprocessing
Data Encoding (Label and one-hot encoding)
Remove constant columns
Remove duplicates
Handle missing values (NaNs)
C(III): Local Data Splitting
Training set (70%)
Validation set (10%)
Test set (20%)
C(IV): Standardization
(Fit on local train; apply to validation and test sets)
C(V): Mutual Information
Using the feature subset (size K) optimized during centralized learning for each classification task
C(VI): Local Training
Local training for E epochs (Lightweight 1D-CNN)
C(VII): Send Model Updates/Gradients (To the aggregator)
S(I): Initialization of the global model
S(II): Selection of an aggregation algorithm for each task
2-class: FedOpt
6-class: FedAvgM
15-class: FedMedian
(chosen empirically based on comparative experiments)
S(III): Broadcasting of the global model
S(IV): Reception of the client updates
S(V): Repetition for R rounds (or until convergence)
S(VI): Evaluation of the global model
Evaluate it on a held-out global test set
S(VII): Execution of simulations
Vary the number of clients (N), participation fraction (C), and number of rounds (R); compare results and select the best configuration for each task
Note: Bayesian hyperparameter optimization was performed prior to the federated learning workflow, and the optimized settings were used for all subsequent simulations.

Fig. 4. Flowchart of federated learning in the dew layer, including local processing and training on the client side and aggregation and evaluation on the server side

S(I) Initialization of the global model

The global model (i.e., lightweight 1D-CNN) is generated and initialized in Stage S(I). The initial parameters of the model, including the weights and biases, are randomly initialized so that the learning process can begin without a bias toward specific data. The initialized global model is transferred to the nodes to begin local training and distributed aggregation.

S(II): Selection of an aggregation algorithm for each task

Stage S(II) is intended to select the optimal aggregation algorithm for each classification problem. Seven well-known aggregation algorithms in federated learning, i.e., FedAvg [66], FedAvgM [67], FedProx, FedAdam, FedOpt [68], FedYogi [69], and FedMedian [70], were executed under the same test conditions using the same local model (lightweight 1D-CNN). The configuration parameters, including the number of nodes (N), participation rate (C), the number of training rounds (R), and data settings, remained unchanged to perform evaluation based on differences in the aggregation. These seven aggregation algorithms were selected since they cover a wide range of approaches in federated learning, including FedAvg as a basic algorithm, FedAvgM as a momentum-based algorithm, FedMedian with robustness to outlier data, and FedYogi, FedOpt, and FedAdM as adaptive optimization approaches. Thus, the evaluation represents the most common aggregation algorithms in the literature.

The final aggregation algorithm for each classification problem would be selected based on a multi-objective analysis that simultaneously met three major criteria: F1-score (macro), communication cost, and the number of rounds before convergence. As no algorithm was superior for all three criteria, decision-making was performed based on Pareto optimality; the algorithm with the most effective trade-off between F1-Macro, connectivity efficiency, and convergence rate, would be selected in each classification problem. Therefore, FedOpt was selected for the binary scenario, FedAvgM was utilized for the six-class scenario, and FedMedian was employed for the fifteen-class scenario. Overall, aggregation in federated learning is formulated as:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_t^k \tag{11}$$

where K is the number of dew nodes participating in training round *t*, which is set based on the predefined participation rate in the simulations, w_t^k denotes the weights of local model *k* trained on the specific split of the Edge-IIoTset in the same node, n_k is the number of the training samples of node *k*, *n* is the total number of samples in all nodes in the same round, and w_{t+1} represents the weights of the global model after aggregation (lightweight 1D-CNN model designed for intrusion detection in this study).

The aggregation algorithms differ from each other in calculating and updating w_{t+1} and/or implementing constraints and specific optimizations in the process. The FedOpt algorithm (selected for the binary scenario) utilizes an adaptive optimizer on the server side that performs updating as:

$$w_{t+1} = w_t - \eta_s \cdot \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{12}$$

where η_s is the server-side learning rate and controls the updating rate of the global model, $\widehat{m}_t$ is the first-order exponential gradient average that captures the dominant direction of weight variations during training rounds ($\widehat{m}_t$ is calculated using the local weight variations of the dew nodes with respect to the global model and represents the overall tendency of all nodes in learning the attack patterns in this study), $\hat{v}_t$ is the second-order exponential gradient average that controls the magnitude and variation of gradients (it ensures that the learning rate for noisy or variable features, such as ports or communication delays, reduces and remains larger for more stable features, such as the flow length or protocol flags), and ε is an extremely small constant added to the denominator to stabilize the training process (it particularly prevents an excessive learning rate increment and model divergence when some features have nearly-zero gradients).

The FedOpt aggregation algorithm is advantageous as it ensures stable training and raises the convergence rate by combining these two components. This is important in the binary scenario of the Edge-IIoTset since the model should balance the highly imbalanced data (in which normal traffic dominates attack samples). In such a case, $\widehat{m}_t$ prevents extreme bias variations in the updates, while $\hat{v}_t$ moderates the learning rate under noisy features and enables the proposed lightweight 1D-CNN model to converge more robustly in a shorter time [68].

The FedAvgM algorithm (selected for the six-class scenario) is a variant of FedAvg that uses the server momentum. It updates the model weights as:

$$\begin{aligned}\Delta_t &= \beta\Delta_{t-1} + (w_t - w_{t-1}) \\ w_{t+1} &= w_t + \Delta_t\end{aligned} \tag{13}$$

where β is a coefficient determining the dependence of the current variations on the previous variations (it determines the extent to which the super-peer in the dew layer uses the shift direction of model weights in the previous weight; this is particularly vital in the Edge-IIoTset with a multiclass, imbalanced distribution since the momentum decreases heterogeneous data-induced variations between dew nodes), Δ_t is a linear combination of the previous and current weight variations (it reflects the averaged shift direction of 1D-CNN weights obtained from the training of dew nodes on different subsets of the Edge-IIoTset; this vector enables the global model to have a more robust and smoother path over time rather than following merely instantaneous node variations), Δ_{t-1} is the accumulated weight variation in the previous round, which reflects the learning trend in round t-1 and serves as a short-term memory in weight updating, w_t denotes the global model weights at the beginning of round t (the weights of the lightweight 1D-CNN model stored in the dew super-peer and can be readily aggregated with the node variations, and w_{t-1} represents the global model weights at the beginning of round t-1, which represents the previous version of 1D-CNN weights before aggregation in round t-1.The weight updating path is not merely dependent on the instantaneous variations in this mechanism as the overall gradient orientation in the previous rounds is also incorporated. This approach is particularly essential for Edge-IIoTset multiclass scenarios since the data distribution between dew nodes is heterogeneous, and the global model weights may have sharp variations without the momentum. The FedAvgM algorithm mitigates such instability and enables more rapid and robust convergence for the 1D-CNN model in the dew setting [67].
The FedMedian algorithm (selected for the fifteen-class scenario) applies the coordinate-wise median rather than weight averaging for aggregation:

$$w_j^{t+1} = median(w_{1,j}^t, w_{2,j}^t, \dots, w_{K,j}^t) \tag{14}$$

where j is the index of each 1D-CNN parameter (e.g., Conv1D filter weights or dense output neurons), $w_{K,j}^t$ is value j of the local model weight of node k in round t (obtained from local component training via the Edge-IIoTset in each dew node), and w_j^{t+1} is value j of the global model weight after aggregation in the dew super-peer (which forms the updated version of the model for the next training round).
While FedAvg considers the weighted averages, FedMedian selects the coordinate-wise median and is more robust to inconsistent nodes and/or outlier data. This is, in particular, important in the fifteen-class scenario since the Edge-IIoTset data have a heterogeneous distribution, and some classes comprise many fewer samples. Thus, FedMedian further stabilizes the learning path of the global model and prevents the excessive effects of nodes with non-representative or noisy data [70].

S(III): Broadcasting of the global model

The initialized global model coupled with the optimal aggregation algorithm for the given scenario is sent to a set of nodes at the beginning of each training round. The number of these nodes is determined based on the participation rate C, with only a fraction of the total nodes being selected and activated in each round. This mechanism reduces communication costs and computational resource consumption in the dew layer and ensures the stability of the system under conditions where not all nodes are simultaneously available. Random node selection in each round maintains randomization during distributed data sampling and minimizes the overfitting risk in a fixed set of nodes, enabling the global model to have a more effective representation of the total data distribution. This approach also provides excellent flexibility in

IIoT and dew computing settings, where a number of nodes may not be available due to resource limitations or a lack of connectivity.

S(IV): Reception of the client updates

Once each local training round in the selected nodes has been completed, model updates (including weights or gradients) are transmitted to the server. The server integrates these updates based on the aggregation algorithm (FedOpt, FedAvgM, or FedMedian) to generate a new version of the global model. This ensures that the knowledge extracted from the local data of each node is incorporated into the global model, and the effects of inconsistent or noisy data are minimized. This phase is essential in the IIoT and dew computing settings as data are highly heterogeneous and decentralized, and the quality of updates transmitted by nodes may vary due to limited resources or unstable connectivity.

S(V): Repetition for R rounds

The federated learning cycle, including the transmission of the global model to the nodes, local training, and receiving and aggregating updates, is iteratively executed for a certain number of rounds until the convergence criterion is met. This study would discontinue the training process if no significant improvement in model performance was observed over a number of consecutive rounds based on error metrics. This prevents ineffective training rounds and saves computational and communication resources within limited-resource settings, such as dew computing.

S(VI): Evaluation of the global model

Once aggregation and convergence have been completed, the trained global model is finally evaluated on the global testing dataset, which has not been used in the training process. The evaluation data are stored independently in order to ensure that the results reflect the generalizability of the global model to new and unseen data.

S(VII): Execution of simulations

Several simulation configurations are implemented to systematically evaluate the effects of each parameter on the federated learning process to identify optimal configurations and enhance model performance. Three key parameters are systematically altered during the federated learning process [66,71,72]:

(I) Number of nodes (N): It determines the number of local nodes that participate in the training process simultaneously. This parameter reflects the network scale and data heterogeneity level [66,72].

(II) Participation rate (C): It is the ratio of nodes selected in each round to the total nodes and simulates the unavailability or temporal disconnection of some nodes. This parameter affects data diversity and convergence rate [66,71].

(III) Number of training rounds (R): It determines the number of model uplinks and downlinks between the nodes and central server and directly relates to resource limitations in the dew layer [71,72].

A total of fifteen simulation configurations with different conditions were devised to assess the relative effects of each parameter both independently and in interaction with the other parameters. The results of these simulations would be compared based on the evaluation criteria (i.e., F1-macro, communication cost, and convergence rate), extracting optimization policies to select the optimal configuration for each scenario (**Section 6).**

This study would select two different approaches to hyperparameter optimization within centralized learning and federated learning settings based on both theoretical and practical aspects. In centralized learning settings, data are stored in an integrated form, and model execution solely includes local training. Hence, the computational time and cost of testing various combinations of hyperparameters are lower, and grid search can be performed. This approach completely evaluates a discrete space of possible values and ensures that the global optimum is identified in this space. The time complexity of this grid search is written as:

$$O(\prod_{i=1}^{k} n_i) \tag{15}$$

where n_i is the number of tested values for hyperparameter i, whereas k is the number of hyperparameters. This cost is acceptable when n_i is small and the data are centralized. In federated learning, on the other hand, data are distributed between several nodes, and the evaluation of a configuration includes multiple stages of model transmission and reception between nodes and the server. The evaluation cost in such settings is written as:

$$Cost_{\text{evaluation}}(f) = Cost_{communication} + Cost_{computational} \tag{16}$$

The total cost of an evaluation comprises a computational cost and a communication cost. In federated learning, the communication cost of repetitive transmissions of model weights between nodes and the server is significantly dominant since it grows proportional to the number of model parameters × the number of participating nodes × the number of training rounds, while the computational cost of each node is limited to local processing on a subset of data. This difference between the two cost components is even greater in the dew layer, in which the bandwidth and network stability are limited.

Research has shown that the communication overhead is the primary bottleneck in federated learning [53]. To address this challenge, this study employed Bayesian optimization in the federated learning process by implementing the structured Parzen estimator (TPE) in the Optuna framework. While Gaussian process-based methods utilize acquisition functions, e.g., expected improvement (EI) or upper confidence bound (UCB), the TPE algorithm guides the search by directly modeling the conditional probability distribution of the objective function. In this method, two independent conditional distributions are defined:

$$l(x) = p(x|y < y^*) \tag{17}$$

$$g(x) = p(x|y \geq y^*) \tag{18}$$

where x is the hyperparameter vector (e.g., learning rate, number of filters, or kernel size in the proposed 1D-CNN), y is the objective function value, which is assumed to correspond to model performance indices in federated learning (validation dataset accuracy), and y^* is a performance threshold (set as a quantile of the previous results).

The objective is to maximize the $l(x)/\text{g(x)}$ ratio in order to select new values of x with a higher likelihood of corresponding to the zone or higher performance. The significant drop in the number of model executions and the focus on promising zones in the search space are major advantages of this approach in federated learning. This remarkably saves time, decreases the communication cost, and improves efficiency in hyperparameter selection, ensuring model convergence under limited-resource conditions.

6. Simulation Results and Technical Discussion

This section discusses the results of the proposed model in the centralized learning and federated learning settings. The hardware and software of the simulation system, statistical data distribution, performance evaluation criteria (accuracy, precision, recall, and F1-score), inference time, communication cost, and the number of convergence rounds are discussed. Finally, the hyperparameter configuration is described, followed by a discussion of the qualitative and quantitative results of each scenario.

6.1. Hardware and software configuration of the simulation system

The simulations were performed under a fixed set of controlled settings using an Asus ROG Zephyrus G14 system with 32 GB of RAM, an NVIDIA GeForce RTX 2060 Max-Q GPU (6 GB), and an estimated compute throughput of 7.5 TFLOPS. These hardware and software details are reported to support reproducibility and facilitate independent verification. Table 3 summarizes the software configuration.

Table 3. Software configuration in the simulations

Library, Tool, or Programming Language	Python	TensorFlow	Flower (Flwr)	Matplotlib	Scikit-Learn	Numpy	Pandas	Optuna
Version	3.10	2.19.0	1.18.0	3.9.4	1.6.1	2.1.3	2.2.3	4.4.0

6.2. Statistical data distribution

Three classification scenarios, including binary, six-class, and fifteen-class, were analyzed. The data were split using a stratified split technique to maintain the same ratios of samples in the classes within the training, validation, and test subsets. Table 4 shows the statistical distribution of the data.

Table 4. Distribution of classes and sub-classes in the Edge-IIoTset after data preprocessing

Category	Subcategory	Class	Records	Percentages (%)
Normal	Normal	Normal	1399624	72.65%
Attack	DDoS	DDoS_UDP	121567	6.31%
		DDoS_ICMP	67939	3.53%
		DDoS_TCP	50062	2.60%
		DDoS_HTTP	48544	2.52%
	Injection	SQL_Injection	50826	2.64%
		Uploading	36957	1.92%
		XSS	15068	0.78%
	Scanning	Vulnerability_Scanner	50026	2.60%
		Port_Scanning	19977	1.04%
		Fingerprinting	853	0.04%
	Malware	Password	49933	2.59%
		Backdoor	24026	1.25%
		Ransomware	9689	0.50%
	MITM	MITM	358	0.02%
Total			1945449	100%

6.3. Evaluation criteria

Macro and weighted accuracy, precision, recall, and F1-score were employed to evaluate model performance in both the centralized and federated learning settings. The number of convergence rounds, communication cost, and inference time were also incorporated as additional criteria for federated learning.

6.3.1. Accuracy

Accuracy is defined as the ratio of true predictions to total samples and serves as a measure of model efficiency in classification:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{19}$$

where *TP*, *TN*, *FP*, and *FN* denote true positives, true negatives, false positives, and false negatives, respectively. In multiclass systems, accuracy is formulated as the average true predictions in all classes:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} 1\,\{\hat{y}_i = y_i\} \tag{20}$$

where *N* is the total number of samples, y_i is the real label, $\hat{y}_i$ is the predicted label, and the characteristic function $1\,\{\hat{y}_i = y_i\}$ is 1 for correct predictions and 0 for incorrect ones.

6.3.2. Precision

Precision is the ratio of true positives to the total positives:

$$Precision = \frac{TP}{TP + FP} \tag{21}$$

In multiclass scenarios, precision is calculated using two approaches:

- Macro-precision: It is the non-weighted average precision of all classes given by:

  $$Precision_{Macro} = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FP_c} \tag{22}$$

  where *C* is the total number of classes, *TPc* is the number of true positives in class *c*, and *FPc* is the number of false positives in class *c*.
- Weighted precision: It is the weighted average of the precision scores of the classes based on the relative frequency of each class:

  $$Precision_{Weighted} = \sum_{c=1}^{C}\frac{n_c}{N}\cdot\frac{TP_c}{(TP_c + FP_c)} \tag{23}$$

  where n_c is the number of real samples in class *c*, while *N* denotes the total number of samples.

6.3.3. Recall

Recall measures the efficiency of a model in detecting true positives and is written as:

$$Recall = \frac{TP}{TP + FN} \tag{24}$$

In multiclass scenarios, recall is calculated through two approaches:

- Macro-recall: It is the non-weighted average recall score of all classes:

  $$Recall_{Macro} = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FN_c} \tag{25}$$

  where *FNc* denotes the number of false negatives in class *c*.
- Weighted recall: It is the weighted average recall score of the classes based on the relative frequency of each class:

  $$Recall_{weighted} = \sum_{c=1}^{C}\frac{n_c}{N}\cdot\frac{TP_c}{TP_c + FN_c} \tag{26}$$

6.3.4. F1-score

The F1-score is a combined criterion defined as the coordinated average of precision and recall and evaluates the balanced performance of a model in producing true positives and avoiding false positives:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{27}$$

In multiclass scenarios, the F1-score is calculated through two approaches:

- Macro-F1: It is the non-weighted F1-score of all classes:

  $$F1_{Macro} = \frac{1}{C}\sum_{c=1}^{C}\frac{2\cdot P_c\cdot R_c}{P_c + R_c} \tag{28}$$

  where P_c is the precision of class *c*, while *Rc* is the recall of class *c*.
- Weighted F1: It is the weighted average F1-score based on the relative frequency of each class:

$$F1_{weighted} = \sum_{c=1}^{C} \frac{n_c}{N} \cdot \frac{2 \cdot P_c \cdot R_c}{P_c + R_c} \tag{29}$$

6.3.5. Specific evaluation criteria in federated learning

Three additional evaluation criteria were incorporated for federated learning, including:

(1) Number of convergence rounds: It denotes the number of rounds required to enable the central model to converge in terms of the target criterion (i.e., loss). This study defined loss function reduction as the stopping criterion for model training; the training process would be discontinued once no improvement occurred in the loss function over a sequence as large as half the total number of predefined rounds (for example, nine rounds in eighteen-round training), with the corresponding round being recognized as the convergence round. It should be noted that a stricter stopping criterion was assumed in the binary scenario due to the relative simplicity of the system and higher convergence rate; i.e., training would be discontinued once the loss function did not improve over five consecutive rounds.

(2) Communication cost: It indicates the total size of data exchanged between the nodes and the central server during the entire federated learning process and is measured in MB. In each training round, each participating node transmits its local model to the server and receives the aggregated model from it. Thus, the communication cost is a function of the model size, the number of participating nodes in each round, and the total number of training rounds:

$$Communication\ Cost\ (MB) = R \times (N \times C) \times 2\ \times Model\ Size\ (MB) \tag{30}$$

where N is the total number of nodes, C is the participation rate of nodes in each round, and R is the number of training rounds. A factor of 2 represents uplink+downlink in each round.

(3) Inference time: It is the average time required to predict each sample in the testing dataset and is calculated based on well-known standards, e.g., MLPerf [73]:

$$Inference\ Time = \frac{Total\ Testing\ Time}{Number\ of\ Test\ Samples} \tag{31}$$

6.4. Hyperparameter configuration in centralized and federated learning

This study tuned hyperparameters through different approaches in the centralized and federated learning settings. The grid search was employed in centralized learning to find an optimal combination of hyperparameters (Table 5). Due to the distributed character of data and limited resources in federated learning, Bayesian optimization and the TPE algorithm were implemented (Table 6).

In both centralized and federated learning settings, feature selection was performed using the MI technique based on the optimal number of features K through the incremental search. Table 7 lists the optimal parameters for each classification scenario.

Table 5. Hyperparameter configuration in classification scenarios for centralized learning

Classification Task	Filters	Kernel Size	Dense Units	Dropout Rate	Learning Rate	Batch Size	Epoch
Binary	16	3	32	0.2	0.001	16	6
6-class	128	3	32	0.2	0.0005	64	50
15-class	128	7	128	0.2	0.0001	32	50

Table 6. Hyperparameter configuration in classification scenarios for federated learning

Classification Task	Filters	Kernel Size	Dense Units	Dropout Rate	Learning Rate	Batch Size	Local Epoch
Binary	16	3	32	0.2	0.001	16	6
6-class	128	2	32	0.3	0.0005	64	10
15-class	64	2	32	0.2	0.0005	64	10

Table 7. Number of selected features in classification scenarios based on MI

Classification Task	**2-class**	**6-class**	**15-class**
Number of Selected Features (K)	10	40	30

6.5. Centralized learning

The performance of the proposed model was evaluated under the framework of centralized learning to enable intricate, reliable comparisons. The first goal was to offer a quantitative baseline for comparison to federated settings and measure the potential drop in accuracy under distributed conditions so that model efficiency could be assessed, along with the security and data privacy advantages of federated learning. The second goal was to compare the performance of the proposed model to well-known methods in the literature in the centralized setting to demonstrate the superior or comparative performance of the proposed model in terms of accuracy, computational efficiency, and learning stability. As a result, three classification scenarios, i.e., binary, six-class, and fifteen-class, were analyzed, reporting the evaluation criteria for each scenario. Table 8 summarizes the performance of the proposed model in the three scenarios, suggesting high accuracy and optimal training and testing times with limited epochs.

Table 8. Model performance in three classification scenarios in centralized learning

Performance Metrics	**2-class**	**6-class**	**15-class**
Test Loss	0.0000	0.0532	0.0625
Test Accuracy	100	97.49	97.01
Epoch	5	50	50
Training Time (s)	793.35	4663.87	8935.29
Testing Time (s)	29.36	10.38	21.85

6.5.1. Binary scenario

The proposed model was found to effectively detect all samples in the binary scenario, yielding an accuracy, precision, recall, and F1-score of 100% for both the “attack” and “non-traffic” classes (Table 9).

Table 9. Model performance under the binary scenario in centralized learning

	Precision (%)	**Recall (%)**	**F1-Score (%)**	**Support**
No Intrusion	100	100	100	280115
Intrusion	100	100	100	108975
Accuracy			100	389090
Macro Avg	100	100	100	389090
Weighted Avg	100	100	100	389090

6.5.2. Six-class scenario

The proposed model had an accuracy score of 97.49% in the six-class scenario. According to Table 10, the precision, recall, and F1-score were maximized to 100% in the normal and MITM classes. The model also showed excellent performance in the DDoS and Scanning classes, yielding evaluation criteria above 93%. Despite the relatively low precision score (71.90%), the model effectively detected the Injection class, with a recall score of 96.04%. The Malware class was the largest challenge; despite a high precision score of 98.64%, the model showed a low recall score (60.12%) and a decreased F1-score.

Table 10. Model performance under the six-class scenario in centralized learning

	Precision (%)	**Recall (%)**	**F1-Score (%)**	**Support**
Normal	100	100	100	280115
DDoS	98.03	97.60	97.81	57299
Scanning	94.74	93.73	94.23	14141
Injection	71.90	96.04	82.23	20702
MITM	100	100	100	69
Malware	98.64	60.12	74.70	16764
Accuracy			97.49	389090
Macro Avg	93.88	91.25	91.50	389090
Weighted Avg	97.96	97.49	97.43	389090

6.5.3. Fifteen-class scenario

The proposed model had an overall accuracy score of 97.01 in the fifteen-class scenario. As shown in Table 11, the model was effective for the Normal, MITM, DDoS-UDP, and DDoS-ICMP classes, with a precision, recall, and F1-score of 100%. The evaluation criteria were higher than 92% for many other classes, including Ransomware, DDoS-TCP, Vulnerability-scanner, and Backdoor. On the other hand, the precision score was relatively low for the Password, SQL Injection, and Uploading classes, even though the model showed acceptable recall scores, suggesting that the proposed model can detect the real-life instances of such attacks. Password and SQL Injection were the most challenging classes, as the F1-score was calculated to be 62.61% and 69.49%, respectively. While the proposed model demonstrates stable performance across all attack classes, it is important to note the inherent complexity of certain intrusion types. Specifically, MitM attacks in the Edge-IIoTset are predominantly represented by short-term flow-level features. This data limitation constrains the model's ability to learn the diverse, long-term sequential dependencies typically associated with real-world MitM behaviors, thus directing the lightweight 1D-CNN to capture local statistical anomalies rather than deep temporal patterns.

Table 11. Model performance under the fifteen-class scenario in centralized learning

	Precision (%)	**Recall (%)**	**F1-Score (%)**	**Support**
Normal	100	100	100	280115
MITM	100	100	100	69
Uploading	89.92	68.66	77.87	7588
Ransomware	99.95	91.55	95.56	2012
SQL_injection	69.41	69.56	69.49	10041
DDoS_HTTP	94.11	89.79	91.90	9620
DDoS_TCP	98.16	94.06	96.07	10052
Password	57.85	68.23	62.61	9869
Port_Scanning	86.84	99.40	92.70	3985
Vulnerability_Scanner	98.70	92.66	95.58	9985
Backdoor	97.62	98.30	97.96	4883
XSS	64.95	87.18	74.44	3073
Fingerprinting	77.06	76.61	76.83	171
DDoS_UDP	99.90	100	99.95	24316
DDoS_ICMP	99.88	99.54	99.71	13311
Accuracy			97.01	389090
Macro Avg	88.96	89.04	88.71	389090
Weighted Avg	97.26	97.01	97.07	389090

It should be noted that high F1-scores may coexist with a non-zero error rate, particularly in imbalanced multiclass datasets such as the Edge-IIoTset. While the error rate is a sample-level metric directly derived from overall accuracy, precision, recall, and F1-score are class-level metrics that are computed independently for each class and then aggregated using macro or weighted averaging. In such datasets, misclassifications in minority classes may increase the overall error rate, even when dominant classes achieve near-perfect precision and recall. Therefore, F1-scores close to 100% do not necessarily imply a zero-error rate, and the reported evaluation metrics remain consistent and complementary. Accordingly, both macro-averaged and weighted-average metrics are reported in this study to provide a balanced evaluation across majority and minority classes.

6.5.4. Proposed model versus earlier methods in centralized learning

Table 12 compares the proposed model to a set of recent studies that utilized various deep learning architectures for intrusion detection on the Edge-IIoTset based on the accuracy, precision, recall, F1-score criteria, along with the training time and testing time in binary, six-class, and fifteen-class scenarios. The architectural and methodological explanations for the superiority of the proposed model over earlier methods are discussed below.

The major improvements in the accuracy, precision, recall, and F1-score of the proposed framework compared to fully connected architectures (DNN) [21,26] stem from a technical chain of architectural mechanisms. First, the Conv1D layer implements convolution on the feature vector, helping detect local inter-feature dependencies. This leads to the extraction of significant spatial patterns, while fully connected architectures transfer all features only as a flat vector to the next layer, and, therefore, local correlations disappear. Hence, DNNs cannot represent such dependencies.

Second, the MaxPooling1D layer leads to decreased dimensionality and helps focus on major patterns, attenuating data noise and moderating the computational load. Third, Dropout randomly excludes a fraction of neurons during the training phase and, therefore, avoids overfitting to the training data. As a result, the model not only shows improved accuracy on the seen data but can also be generalized to unseen data. In light of this chain, the proposed model yielded reproducible results, even under the six- and fifteen-class scenarios in which the data were more dispersed and imbalanced. On the other hand, the DNNs showed substantially lower accuracy and F1-score as they lacked such mechanisms.

In the RNN models, e.g., GRU and LSTM [19], each computational step is dependent on the output of the previous step due to their sequential processing character. This sequential dependence prevents parallel training and, therefore, substantially increases the training time (over 100% longer training times relative to the proposed model). Furthermore, the data of Edge-IIoTset do not contain long-term temporal dependencies, and, consequently, the major advantage of RNN models in capturing long-range dependencies is not triggered. As a result, an increased training time would not improve the performance criteria.

On the other hand, the proposed model performs parallel training on the entire feature sequence by leveraging Conv1D and simultaneously captures local patterns and key inter-feature dependencies. This chain remarkably decreases the computational cost and shortens the training and inference times. In addition, complex integrated models, e.g., CNN+LSTM+GRU [20], have training times over 200% longer than the proposed model, raise the overfitting risk, and require stronger hardware resources due to their large depth and multiplicity of parameters. In contrast, the proposed lightweight architecture, with tailored layers and effectively optimized hyperparameters, maintained competitive accuracy and F1-score, with substantially shorter training and testing times.

The use of complex feature extraction blocks has a generic design in the InceptionTime [24] and DeepAK-IoT [25] architectures, without compatibility with the imbalanced and noisy data of the Edge-IIoTset. Despite their high learning capacities, such blocks cannot generate detailed decision boundaries effectively in multiclass scenarios, particularly in the presence of feature overlaps, and these architectures showed inferior performance to the proposed model. In contrast, through the optimized number of filters and kernel size based on MI maximization, the proposed model focused on extracting the most discriminative features. In addition, the use of an optimized number of layers prevented unnecessary complexity and maintained the efficiency of the model in separating the classes. This chain of design enabled a more accurate generation of decision boundaries within classes with overlapping features, leading to significant improvements in the evaluation criteria compared to earlier architectures under multiclass scenarios.

Table 12. Proposed model versus earlier works on the Edge-IIoTset in centralized learning

Model	Accuracy (%)			Precision (%) (Macro/Weighted)			Recall (%) (Macro/Weighted)			F1-Score (%) (Macro/Weighted)			Train Time (s)			Test Time (s)		
	2-class	6-class	15-class	2-class	6-class	15-class	2-class	6-class	15-class	2-class	6-class	15-class	2-class	6-class	15-class	2-class	6-class	15-class
DNN [21]	99.99	96.01	94.67	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
Inception Time [24]	-	-	94.94	-	-	NR/98.3	-	-	NR/92.4	-	-	NR/95.3	-	-	NR	-	-	NR
DeepAK-IoT [25]	-	-	94.96	-	-	NR/95.40	-	-	NR	-	-	NR/94.72	-	-	NR	-	-	NR
Unsupervised Deep Auto-Encoder [29]	99.7	-	-	99.9	-	-	100	-	-	99.9	-	-	NR	-	-	NR	-	-
CNN [19]	-	-	94.83	-	-	NA/98.38	-	-	NR/92.30	-	-	NR/95.19	-	-	1080	-	-	NR
GRU [19]	-	-	93.98	-	-	NR/97.02	-	-	NR/91.76	-	-	NR/94.28	-	-	2100	-	-	NR
GRU + CNN [19]	-	-	94.94	-	-	NR/98.49	-	-	NR/92.28	-	-	NR/95.23	-	-	5820	-	-	NR
LSTM [19]	-	-	93.93	-	-	NR/97.45	-	-	NR/91.28	-	-	NR/94.21	-	-	2160	-	-	NR
DNN [26]	100	96.78	94.90	100	NR/96.82	NR/95.33	100	NR/96.16	NR/94.90	100	NR/95.97	NR/94.62	715.36	373.88	367.58	20.05	18.11	27.99
CNN + LSTM + GRU [20]	100	97.44	96.90	100	93/98	90/97	100	91/97	85/97	100	91/97	86/97	1885.46	14803.63	14719.47	42.53	42.20	40.65
DNN [31]	100	98.83	99.39	100	91.58/NR	81.46/NR	100	85.27/NR	75.90/NR	100	86.68/NR	77.67/NR	384.82	363.55	388.24	8.10	6.68	6.74
The Proposed Model	100	97.49	97.01	100	93.88/97.96	88.96/97.26	100	91.25/97.49	89.04/97.01	100	91.50/97.43	88.71/97.07	793.35	4663.87	8935.29	29.36	10.38	21.85

This is a key operational advantage in dew computing settings with limited computational resources and demand for real-time responsiveness, unlike a major fraction of heavy models in the literature.

6.6. Federated learning

The performance of the proposed model in federated learning under binary, six-class, and fifteen-class scenarios is reported in this section. The main goal is to evaluate the efficiency of the model in distributed settings without raw data transmission to the central server, and not only accuracy but also the communication cost and simulation time are important. For each scenario, a multi-objective analysis is performed based on the Pareto technique under three key criteria: the maximization of F1-score (macro) and the minimization of the communication cost and simulation time. Then, the Pearson correlation analysis ($p<0.05$) is performed on a set of criteria, including accuracy, precision (macro), recall (macro), F1-score (macro), simulation time, communication cost, and number of convergence rounds, to evaluate the inter-index correlations.

It should be mentioned that the IID data were used for the nodes in the simulations so that the effects of system parameters (i.e., number of nodes, participation rate, and number of rounds) could be clearly studied. However, the Edge-IIoTset is intrinsically heterogeneous and imbalanced and contains protocols, various network features, and attack classes with highly different frequencies. Therefore, the results reported for the three scenarios can demonstrate the efficiency of the proposed model in coping with computational resource limitations in the dew layer and the complexity of real-life data simultaneously.

6.6.1. Simulation configurations and federated learning scenarios

Three key parameters, i.e., the number of nodes, participation rate in each round, and the number of training rounds, were determined to manage the resource budget and evaluate the effects of design factors on the performance of federated learning. The combination of these three parameters controls the total computational and communicational loads of the system. Therefore, the normalized computational load was designed for a fair comparison of scenarios:

$$Load_{norm} = \frac{N \times C \times R}{6} \tag{32}$$

A normalization factor of 6 is set. The product of N×C×R is 6 in the reference scenario (i.e., Configuration 1). Dividing all values by this factor yields a computational load of 1 for the reference scenario, and the values of other scenarios are reported in an undimensionalized and comparable format. Table 13 lists the configurations of fifteen simulations devised for federated learning, where the three main parameters (N, C, and R) are provided, along with the corresponding normalized computational load.

Table 13. Configurations of fifteen federated learning simulations

Simulation ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Number of Clients (N)	5	5	5	10	10	10	20	20	20	20	50	50	50	50	50
Client Participation Rate (C)	0.1	0.3	1	0.1	0.3	1	0.05	0.2	0.5	1	0.05	0.1	0.3	0.5	1
Number of Rounds (R)	12	15	18	12	15	18	10	12	15	18	8	10	12	15	18
Normalized Computational Load Prediction	1	3.75	15	2	7.5	30	1.66	8	25	60	3.33	8.33	30	62.5	150

The search space of the three main parameters is:

- $N \in \{5,10,20,50\}$;
- $C \in \{0.05,0.1,0.2,0.3,0.5,1\}$;
- $R \in \{8,10,12,15,18\}$.

These ranges produce 120 possible combinations. However, not all of these combinations are unique since combinations that have the same product of N×C×R generate the same normalized computational load (Eq. (32)) and produce duplicate results. To eliminate such redundancy and enable optimal resource utilization, merely fifteen configurations yielding distinct normalized computational loads were selected (Table 13). Two configurations with a normalized computational load of 30 (i.e., Configurations 6 and 13) were not excluded so that the effects of N and C at a given normalized computational load on stability, latency, and communication cost could be evaluated.

6.6.2. Binary classification (FedOpt aggregation algorithm)

As shown in Table 14, the lightweight 1D-CNN model produced an accuracy and F1-score of 1.0 in all federated learning simulations. This is explained by the optimized architecture with a single Conv1D layer that effectively captured the local patterns of network flows and addressed computational resource limitations in the dew layer. The selection of the FedOpt aggregation algorithm was also a determinant; the adaptive server optimizer enabled the global model to converge in only five training rounds, minimizing the communication cost. This combination indicates that the lightweight design and adaptive aggregation ensured full accuracy and communicational efficiency at the same time.

This finding is further supported by comparison to centralized learning. In the centralized setting, the model obtained a test accuracy of 100% and a test loss of nearly 0.0000 within five epochs and then displayed the same performance in the federated setting.

The main difference lies in the training time and communication cost; the training time was 793 s in the centralized setting, while it was found to vary from 753 to 4629 s in the federated setting, depending on the configuration. In the optimal configuration (i.e., Configuration 11), the training time was almost the same as that in the centralized setting, with the advantage that no raw data was transmitted, and merely model parameters were shifted.

Therefore, although there was a communication overhead, it was negligible compared to full data transmission, and data privacy was protected. It should also be noted that, despite the IID data in the nodes, the Edge-IIoTset is intrinsically heterogeneous and imbalanced. Thus, the high convergence rate with such excellent accuracy demonstrates the efficiency of the model in handling the complexity of real-life data.

As shown in Fig. 5 and Table 15, Configuration 11 established the best trade-off between the three key criteria: an F1-score of 100, a communication cost of 0.8 MB, and a simulation time of 753 s. Despite their identical classification performance, the other configurations were excluded from the Pareto frontier due to their extremely high communication costs (up to 19.89 MB) and simulation times (up to 4629 s).

This highlights the necessity of choosing optimal values for N, C, and R to control the computational and communicational loads. Moreover, the Pearson correlation analysis supported this finding (Fig. 6). The evaluation criteria, i.e., accuracy, precision, recall, and F1-score, were the same across all configurations; however, a significant, positive correlation existed between the communication cost and simulation time. In other words, an increase in the participation rate or the number of rounds merely increases the system overhead and decreases overall efficiency without improving model quality.

Table 14. Results of fifteen simulation configurations under the binary scenario in federated learning

Simulation ID	Accuracy (%)	Loss	Rounds to Convergence	Precision (%)	Recall (%)	F1-Score (%)	Comm. Cost (MB)	Simulation Time (s)	Inference Time (s)
1	100	1.00E-15	5	100	100	100	0.80	2147.69	0.1497
2	100	2.95E-15	5	100	100	100	0.80	2111.07	0.1489
3	100	1.37E-17	5	100	100	100	1.99	3744.79	0.1464
4	100	3.30E-17	5	100	100	100	0.80	1334.31	0.1492
5	100	4.98E-18	5	100	100	100	1.19	1391.73	0.1504
6	100	8.40E-17	5	100	100	100	3.98	3820.14	0.151
7	100	1.52E-14	5	100	100	100	0.80	968.22	0.1501
8	100	8.01E-15	5	100	100	100	1.59	1371.13	0.1504
9	100	8.74E-15	5	100	100	100	3.98	2295.39	0.151
10	100	1.21E-15	5	100	100	100	7.96	3701	0.1457
11	100	1.11E-13	5	100	100	100	0.80	753.66	0.1529
12	100	2.60E-14	5	100	100	100	1.99	971.8	0.1501
13	100	5.59E-15	5	100	100	100	5.97	1641.61	0.1482
14	100	2.88E-15	5	100	100	100	9.94	2524.47	0.1616
15	100	5.20E-15	5	100	100	100	19.89	4628.74	0.1725

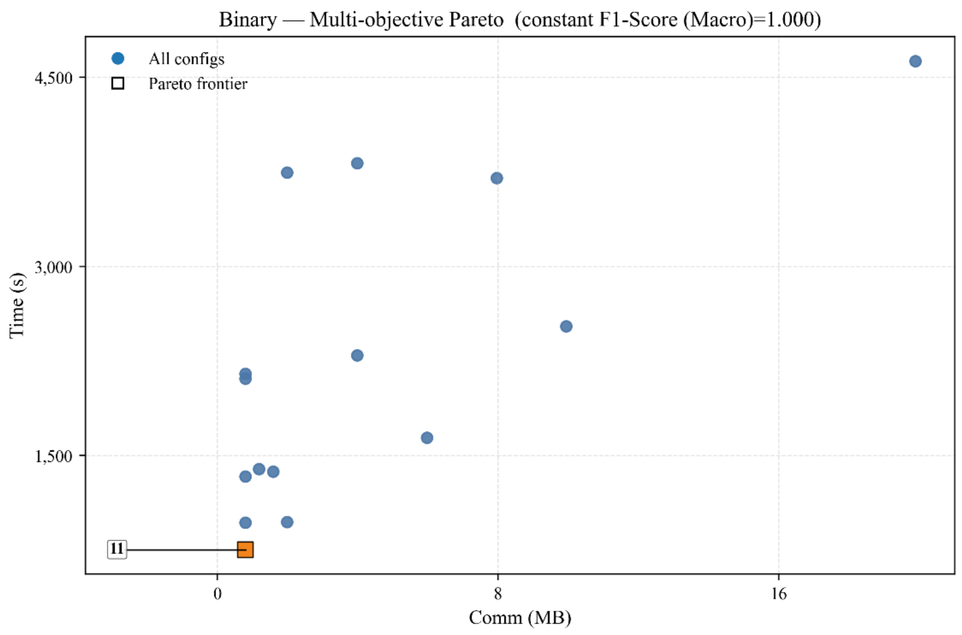


Fig. 5. Pareto frontier under the binary scenario (trade-off between the communication cost and simulation time) in federated learning

Table 15. Pareto-optimal configuration corresponding to Fig. 5 under the binary federated learning scenario

ID	F1-Score (Macro) (%)	Comm (MB)	Time (s)	Conv. Rounds
11	100	0.80	753	5

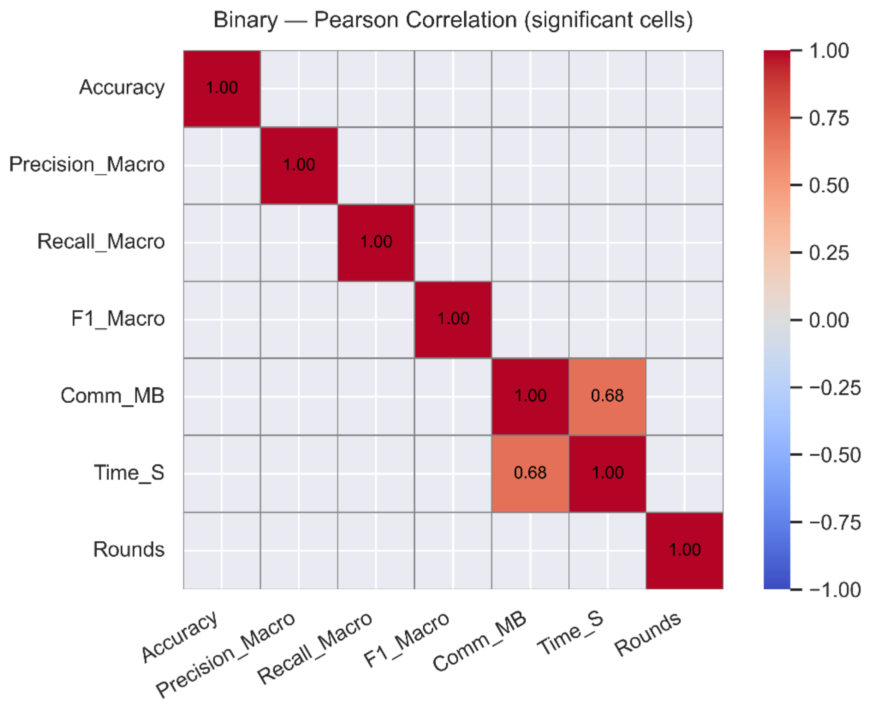


Fig. 6. Pearson correlation matrix for evaluation criteria under the binary scenario in federated learning

6.6.3. Six-class scenario (FedAvgM aggregation algorithm)

As shown in Table 16, the lightweight 1D-CNN model with the FedAvgM algorithm showed satisfactory stability in the six-class scenario, despite its higher complexity than the binary scenario. In most configurations, accuracy was higher than 96%, and the F1-score (macro) varied from 0.88 to 0.91. FedAvgM played a significant role in this performance since the incorporation of the momentum into the aggregation process decreased variations induced by the intrinsic data heterogeneity of the Edge-IIoTset and shortened and stabilized the convergence path. The combination of the optimal 1D-CNN model and FedAvgM algorithm indicates that an efficient and lightweight architecture can be established for dew settings even under imbalanced and heterogeneous data.

Fig. 7 and Table 17 represent the multi-objective analysis to evaluate the trade-off between three key criteria: F1-macro, communication cost, and simulation time. Five configurations (including Configurations 1, 5, 7, 8, and 11) were found to form the Pareto frontier. They produced an F1-macro of approximately 0.91, which is comparable to centralized learning (F1-score (macro)=91.50 in 4664 s). However, the training times of the configurations in the Pareto frontier were significantly shorter, varying from 972 to 3388 s. For example, Configuration 1 was calculated to have an F1-score of 91.25, a communication cost of 11.09 MB, and a time of 2301, representing the best trade-off between the key criteria. This suggests that federated learning enabled comparable accuracy to centralized learning without raw data transmission and even displayed superior performance in terms of training time in many configurations.

The Pearson correlation analysis provides deeper insights into these patterns (Fig. 8). The negative correlation between the F1-score (macro), the number of rounds, and simulation time ($r \approx -0.7$) suggests that an increase in the training length in heterogeneous data not only does not improve model quality but also reduces efficiency. Furthermore, the positive correlation between the communication cost and simulation time ($r \approx 0.61$) demonstrates that an increase in the participation rate of nodes directly raises the system load without significantly improving accuracy. These findings, along with Pareto analysis, highlight the importance of setting a

limited participation rate and using momentum-based aggregation algorithms since this may protect data privacy, control the network overhead, and provide a comparable level of accuracy to centralized learning.
The average inference time for each sample remained in the range of 0.156-0.163 s across all federated learning simulations (Table 16). This stability indicates that the alteration of N, C, R, and, thus, computational and communicational loads had no effect on the ultimate prediction speed of the model. This is explained by the lightweight character of the 1D-CNN architecture; once training has been completed, the inference process is dependent only on the final model structure and operates independently of the simulation conditions.

Table 16. Results of fifteen simulation configurations under the six-class scenario in federated learning

Simulation ID	Accuracy (%)	Loss	Rounds to Convergence	Precision (%) (Macro/Weighted)	Recall (%) (Macro/Weighted)	F1-Score (%) (Macro/Weighted)	Comm. Cost (MB)	Simulation Time (s)	Inference Time (s)
1	97.41	0.0558	7	93.62/97.89	91.03/97.41	91.25/97.35	11.09	2301.05	0.1632
2	97.39	0.0566	9	93.35/97.82	91.01/97.39	91.17/97.33	14.26	2946.65	0.1598
3	93.54	0.1333	18	82.55/95.51	81.71/93.54	78.17/93.39	71.29	10016.86	0.1615
4	97.26	0.0628	12	90.51/97.70	90.65/97.26	89.62/97.23	19.01	2643.86	0.1565
5	97.45	0.0566	15	93.64/97.94	91.35/97.45	91.44/97.40	35.64	3388.27	0.1606
6	95.68	0.0843	18	75.65/95.99	87.58/95.68	78.38/95.59	142.57	10650.41	0.1606
7	97.23	0.0613	6	92.54/97.62	90.42/97.23	90.59/97.18	9.50	1132.51	0.1591
8	97.26	0.0601	7	92.91/97.70	90.35/97.26	90.61/97.20	22.18	1688.9	0.1581
9	96.81	0.0694	9	92.77/97.23	87.32/96.81	88.88/96.71	71.29	3481.99	0.1622
10	96.84	0.0723	9	92.43/97.48	88.19/96.84	88.87/96.77	142.57	5643.37	0.1608
11	96.93	0.0699	6	91.50/97.27	88.57/96.93	89.23/96.88	9.50	972	0.1572
12	96.91	0.0663	6	91.52/97.24	87.83/96.91	88.86/96.86	23.76	1165.55	0.1566
13	96.77	0.0725	8	91.22/97.09	87.21/96.77	88.32/96.69	95.05	2571.88	0.1603
14	96.53	0.0751	8	81.33/96.73	86.07/96.53	81.47/96.42	158.41	3945.11	0.1572
15	96.7	0.0751	10	90.94/97.09	87.76/96.70	88.04/96.55	396.04	8118.9	0.1608

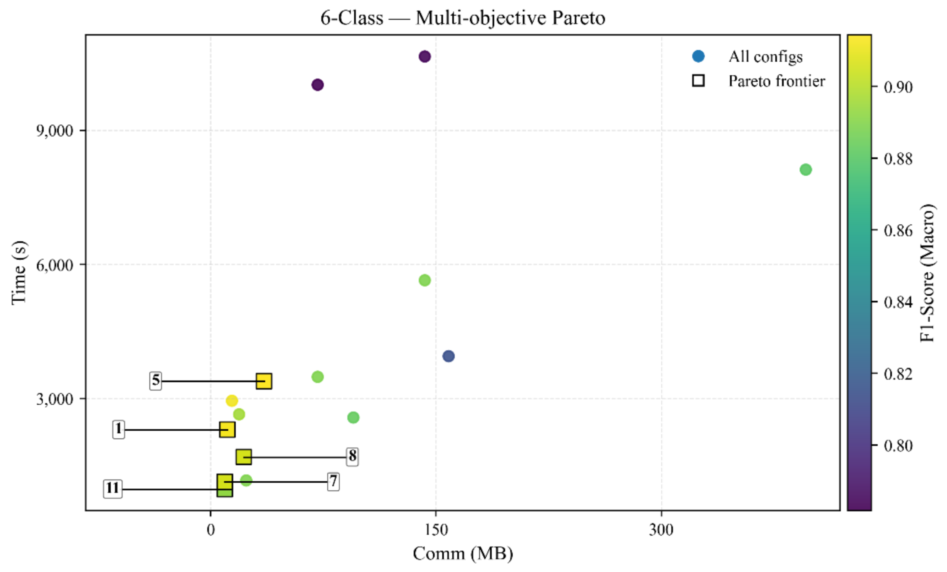


Fig. 7. Pareto frontier under the six-class scenario in federated learning (trade-off between communication cost and simulation time)

Table 17. Pareto-optimal configurations corresponding to Fig. 7 under the six-class federated learning scenario

ID	F1-Score (Macro) (%)	Comm (MB)	Time (s)	Conv. Rounds
11	89.23	9.50	972	6
7	90.59	9.50	1132	6
1	91.25	11.09	2301	7
8	90.61	22.18	1688	7
5	91.44	35.64	3388	15

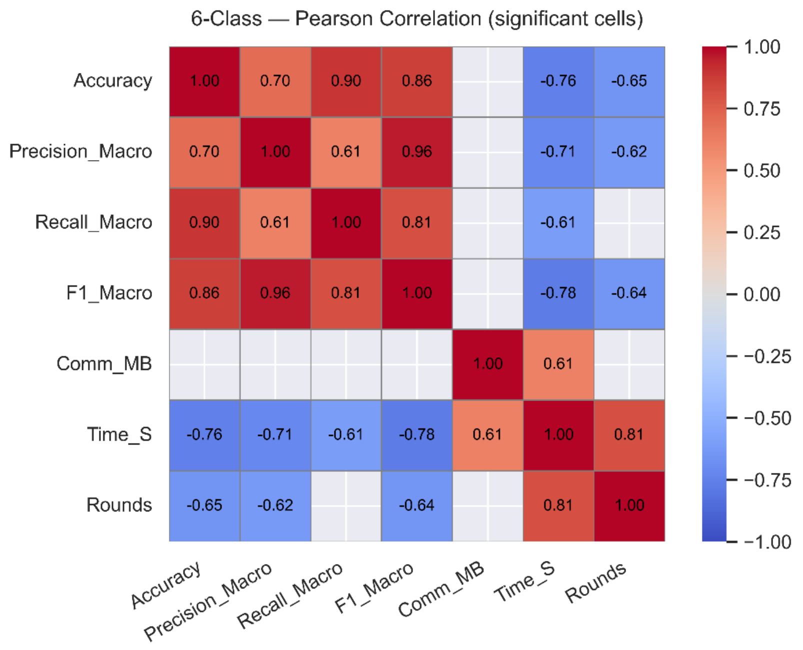


Fig. 8. Pearson correlation matrix for evaluation criteria under the six-class scenario in federated learning

6.6.4. Fifteen-class scenario (FedMedian aggregation algorithm)

As shown in Table 18, the 1D-CNN model maintained a high level of stability and accuracy in the fifteen-class scenario, despite a significant increase in the complexity and diversity of classes. The shallow structure of 1D-CNN minimized the computational load in extracting local features, with no major drop in the key criteria, such as F1-score and precision. The FedMedian algorithm played a key role since it is robust to inconsistent and imbalanced data and prevents the deviation of the global model due to the extreme heterogeneity of the Edge-IIoTset, stabilizing convergence in a limited number of rounds.

Fig. 9 and Table 19 show the multi-objective optimization results. The first Pareto frontier comprised Configurations 2, 4, 7, 8, 11, and 12, with a trade-off between the three key criteria: F1-score (macro), simulation time, and communication cost. These configurations yielded an F1-score (macro) of 0.80-0.85; although the F1-score was smaller than centralized learning (F1=0.8871), the configurations were found to have significantly shorter training times. For example, Configuration 11 obtained an F1-score of 0.8012 with a communication cost of 2.83 MB in only 933 s, while an F1-score of 0.881 was achieved within nearly 8935 s in centralized learning. This indicates that federated learning can significantly decrease the training time and communication overhead with a slight accuracy drop in limited-resource operational settings, representing a more practical solution.

These findings are supported by the Pearson correlation coefficients (Fig. 10). It was found that the F1-score (macro) had strong correlations with precision (macro) (r≈0.92) and recall (macro) (r≈0.99), suggesting model robustness in various performance aspects. In addition, the positive correlations between the communication cost and simulation time (r≈0.58) and between the number of rounds and simulation time (r≈0.90) suggest that an increase in the participation of nodes or training time directly increases the computational and communicational loads without a significant improvement in model quality. Therefore, optimizing N, C, and R and restraining the participation rate would be an effective strategy to achieve communication and time efficiency, along with an acceptable level of accuracy. The average inference time under the fifteen-class scenario varied from 0.154 to 0.196 s for each sample. These relatively stable inference times indicate that a rise in the number of classes had no significant effect on the online prediction cost, and the 1D-CNN model provided a short response time for real-time applications, even in more complex scenarios.

Table 18. Results of fifteen simulation configurations under the fifteen-class scenario in federated learning

Simulation ID	Accuracy (%)	Loss	Rounds to Convergence	Precision (%) (Macro/Weighted)	Recall (%) (Macro/Weighted)	F1-Score (%) (Macro/Weighted)	Comm. Cost (MB)	Simulation Time (s)	Inference Time (s)
1	96.62	0.0713	12	89.01/96.97	83.59/96.62	84.14/96.67	5.66	3992.92	0.1646
2	96.67	0.0722	9	89.05/96.93	84.10/96.67	84.54/96.72	4.25	2991.84	0.1543
3	96.55	0.0768	13	89.18/96.92	83.42/96.55	83.62/96.46	15.33	7796.55	0.1578
4	96.62	0.074	10	89.62/97.07	83.08/96.62	84.06/96.69	4.72	2368.98	0.1606
5	96.52	0.0726	15	89.97/97.09	82.67/96.52	83.71/96.56	10.61	4270.93	0.1964
6	95.67	0.1052	18	82.35/96.26	73.87/95.67	74.76/95.41	42.45	12638.28	0.185
7	96.25	0.082	9	87.91/96.76	82.16/96.25	82.19/96.13	4.25	1926	0.1934
8	96.13	0.0838	7	88.22/96.84	81.86/96.13	81.99/95.98	6.60	1701.01	0.1603
9	96.08	0.0828	9	85.99/96.18	80.38/96.08	81.32/96.09	21.23	3475.98	0.1619
10	96.01	0.0914	14	88.29/96.58	80.66/96.01	81.28/95.89	66.04	8715.55	0.1611
11	95.7	0.0957	6	84.78/95.83	79.13/95.70	80.12/95.71	2.83	933.05	0.1551
12	95.78	0.0958	8	85.52/95.91	79.32/95.78	80.52/95.78	9.43	1515.73	0.161
13	95.8	0.0958	8	83.61/95.91	79.00/95.80	79.05/95.81	28.30	2582.91	0.1601
14	96.1	0.0888	14	86.06/96.17	81.20/96.10	81.76/96.11	82.55	7115.5	0.1627
15	95.87	0.0954	18	85.59/96.04	79.28/95.87	80.21/95.84	212.27	8873.4	0.1558

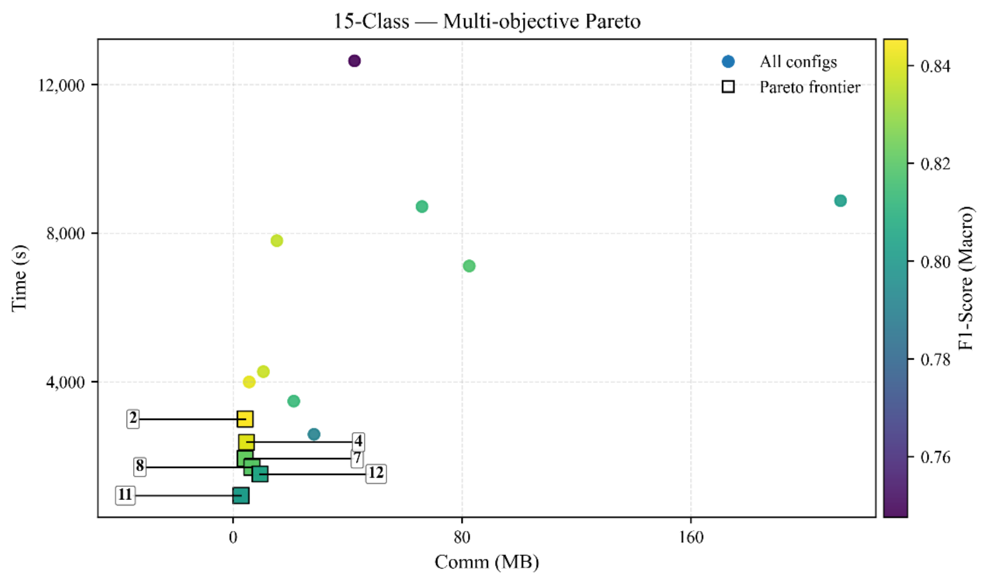


Fig. 9. Pareto frontier under the fifteen-class scenario in federated learning (trade-off between communication cost and simulation time)

Table 19. Pareto-optimal configurations corresponding to Fig. 9 under the fifteen-class federated learning scenario

ID	F1-Score (Macro) (%)	Comm (MB)	Time (s)	Conv. Rounds
11	80.12	2.83	933	6
7	82.19	4.25	1926	9
2	84.54	4.25	2991	9
4	84.06	4.72	2368	10
8	81.99	6.60	1701	7
12	80.52	9.43	1515	8

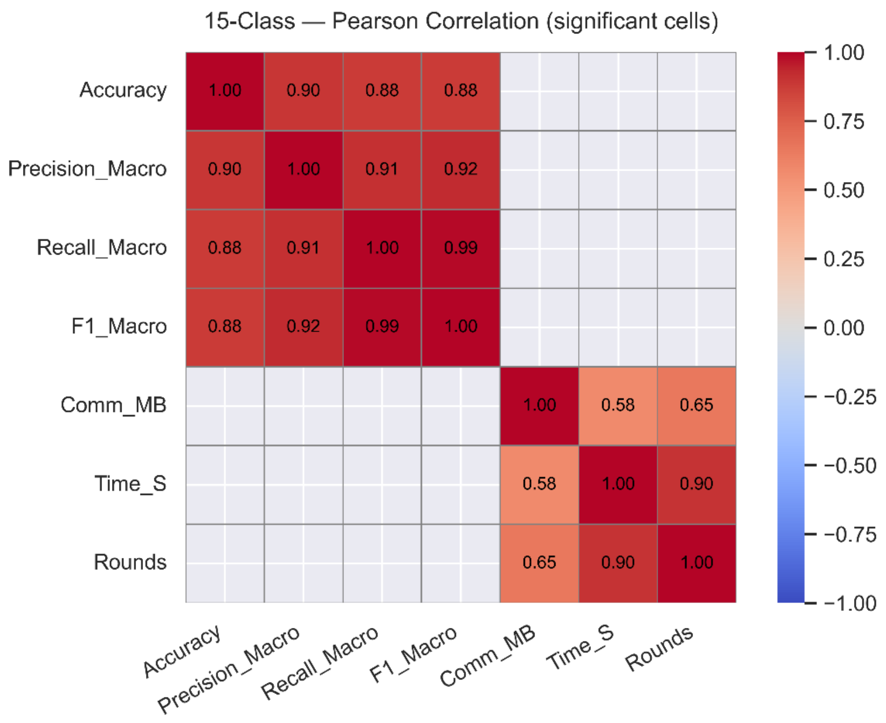


Fig. 10. Pearson correlation matrix for evaluation criteria under the fifteen-class scenario in federated learning

6.6.5. Analysis of sensitivity to R, N, and C

To further evaluate the performance of the proposed model under various scenarios, the sensitivity of its efficiency and effectiveness to three key parameters in federated learning, i.e., N, C, and R, was measured. Sensitivity analysis allows for measuring the contribution of a given factor to performance enhancement/degradation and highlights its optimal range or saturation points. This approach is even more important in real-life settings with limited resources as it helps select an efficient configuration and avoid resource waste. To derive clear and concise results, the sensitivity analysis was conducted based on the F1-macro only under the six-class scenario since it provides an intermediate level of complexity between simple (binary) and sparse (fifteen-class) scenarios and offers a balanced representation of system behavior. Furthermore, F1-macro would enable a fairer evaluation on imbalanced data since it assigns equal weights to classes.

A single-variable analysis was performed to assess the contribution of the number of nodes (i.e., C and R remained unchanged to capture the independent effect of N). Configurations with C=1.0 and R=18 were selected since they covered the maximum participation rate and maximum number of training rounds, and the alteration of N would produce the maximum outcome. In addition, to obtain a more reliable sensitivity profile, this analysis was conducted by re-running the federated learning experiments under multiple N configurations rather than relying on a single limited set of predefined settings. As shown in Fig. 11, increasing the number of nodes from 5 to around 20 substantially improved F1-macro (from approximately 0.82 to about 0.88). After reaching this range, further increases in N did not yield a consistent performance gain, and F1-macro remained largely stable with only minor fluctuations across larger N values up to 50. This indicates that raising the number of nodes up to an effective range can enhance detection quality by leveraging broader data diversity, whereas beyond an optimal level the marginal benefits saturate and small oscillations may occur due to increased dispersion across clients and the stochasticity of distributed training, without providing a clear improvement in efficiency.

To assess the independent contribution of the participation rate, the number of nodes, and the number of rounds remained unchanged. Since such a combination of parameters did not exist in the configurations, the partial dependence plot (PDP) was employed based on regression analysis. It would estimate the average effect of C on F1-macro by keeping other variables unchanged. According to Fig. 12, the correlation between the participation rate and performance was not completely linear or uniform. At smaller participation rates (≈0.1-0.2), model performance was relatively high since the gradients were mostly stable, and the communication cost was low. As the participation rate rose to a medium level (0.3-0.6), the lack of a sample distribution balance and increased gradient noise decreased F1-macro to a significant extent, despite increased data diversity. At highly large participation rates (>0.7), almost all nodes participated in each round, and the data heterogeneity effect declined. As a result, performance began to improve again. This represents an inverted U-curve; highly small or large participation rates yielded higher performance, while medium participation rates led to the lowest performance. This finding is essential from a practical perspective. A low participation rate (≈0.1-0.2) would be optimal when the communication resources are limited since optimal performance can be achieved. On the other hand, high participation rates (≈1.0) ensure the maximum performance when accuracy maximization is the primary goal, and there is no communication limitation. Thus, it is essential to avoid medium participation rates to prevent poor accuracy and communication resource waste.

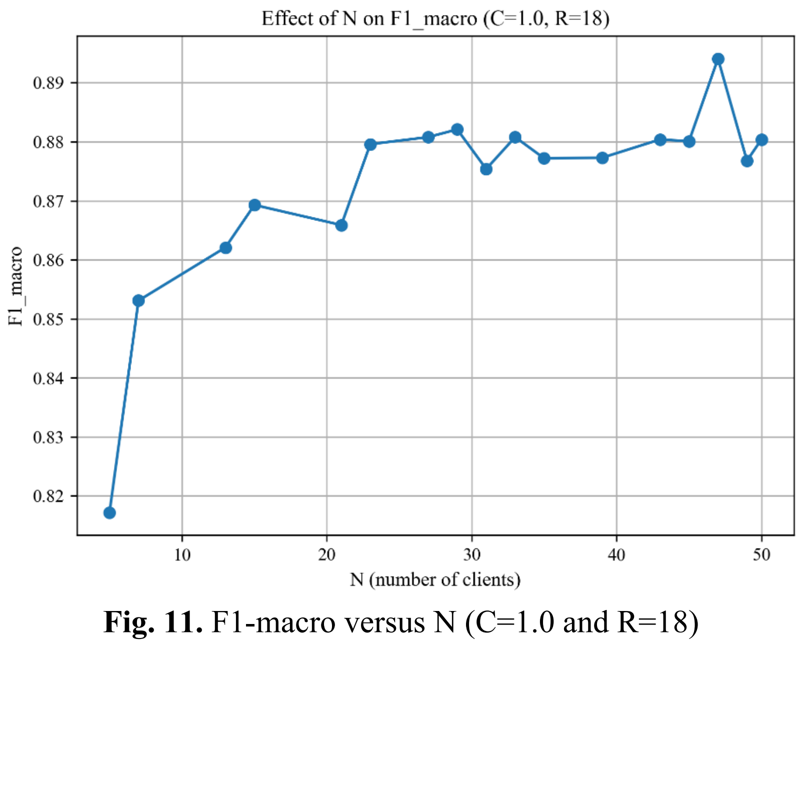


Fig. 11. F1-macro versus N (C=1.0 and R=18)

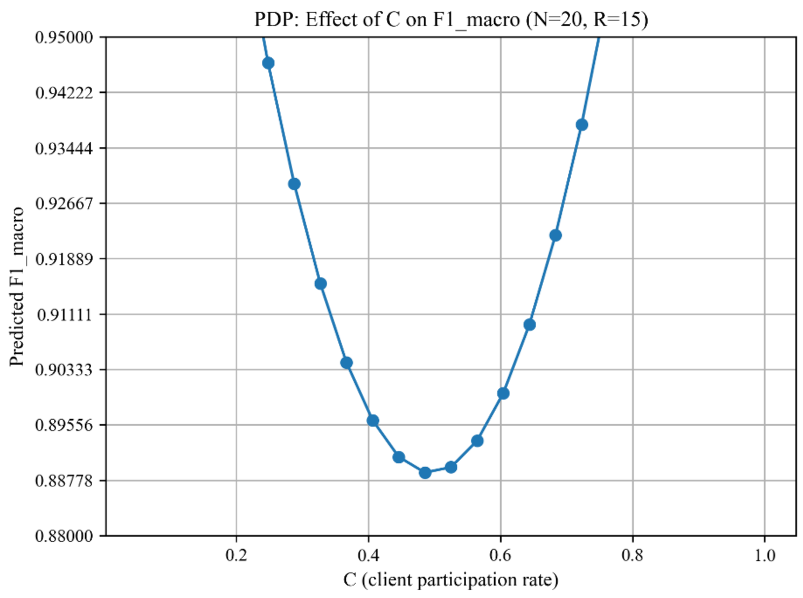


Fig. 12. F1-macro versus C (N=20 and R=15)

Likewise, the independent contribution of the number of training rounds to performance was evaluated at a given number of nodes and participation rate. However, there was no such confirmation in the simulations, and the PDP approach was adopted. Fig. 13 plots F1-macro versus the number of training rounds for 20 nodes and a participation rate of 0.3. An estimated R^* value of nearly 8.4 was found as an approximate threshold to handle the initial instability. However, the observed data suggest that performance continued to drop at R values of up to 13-14, and stable convergence began at R values above 13-14 in the curve. This difference is explained by the approximate character of the PDP method, which represents the minimum smoothed curve slightly earlier than the real minimum. From a practical perspective, the results suggest that (1) training rounds fewer than 10 leads to a major performance decrease, and (2) a rise in the number of training rounds above the real threshold (13-14) improves performance; however, the improvement rate reduces, and a trade-off is to be established between performance improvement, communication cost, and training time.

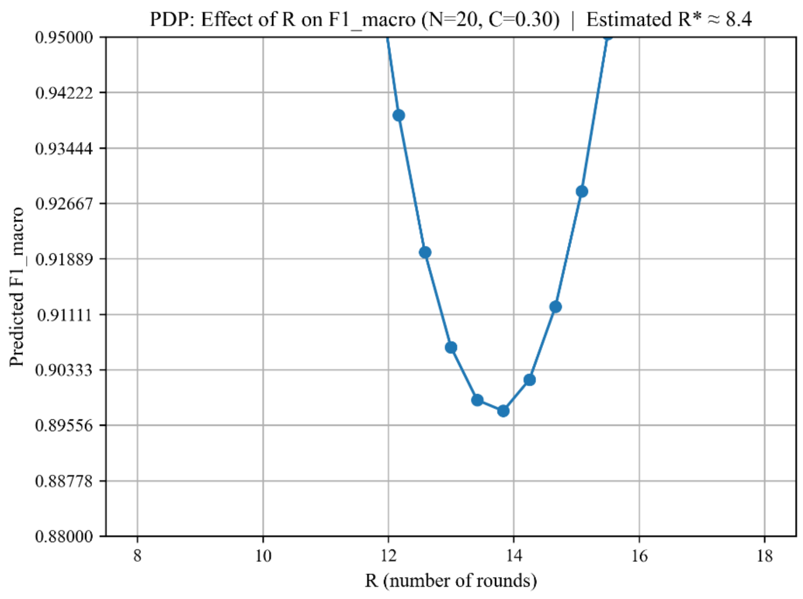


Fig. 13. F1-macro versus R (N=20 and C=0.30)

The interaction of the number of nodes and participation rate was also evaluated. In the design of real-life federated learning systems, these parameters are tuned simultaneously, and altering one parameter without taking the other into account would lead to incomplete interpretations. Therefore, the number of training rounds was set to an intermediate level (R=12), and F1-macro was plotted versus N and C in a contour map to capture the overlapping effect of these two parameters (Fig. 14). An inverted U-curve was derived, even though its intensity would be dependent on the network scale; in small to medium networks (N<20), an intermediate participation rate (C≈0.3-0.6) maximizes the F1-score reduction. In larger networks, on the other hand, the F1-score reduction is moderated, with lower sensitivity to the participation rate. In other words, a larger network would enjoy higher robustness to participation rate variations. However, achieving higher accuracy requires the simultaneous enlargement of N and C as increasing either parameter is not sufficient. The iso-contours suggest that a moderate performance level (F1≈0.88) can be achieved even at a small participation rate (C≈0.1) and a medium network scale (N=10-25); however, higher performance levels (F1=0.91-0.92) would require a minimum of 20 nodes and a minimum participation rate of 0.2-0.3. The maximum performance (R≈0.95) can be implemented only by combining large networks (N≈30-50) and relatively large participation rates (C≈0.35-0.50). Overall, based on the interaction of the number of nodes and participation rate, it can be concluded that (1) moderate participation rates are to be avoided in small or medium networks to prevent maximum performance reduction, (2) maximum performance in larger networks requires a simultaneous enlargement of N and C, even though the negative effect of medium C levels is lower, and (3) these results complement the single-variable sensitivity analysis and show that the participation strategy should be optimized based on the network scale to prevent apparent contradictions in the interpretation of the outcomes.

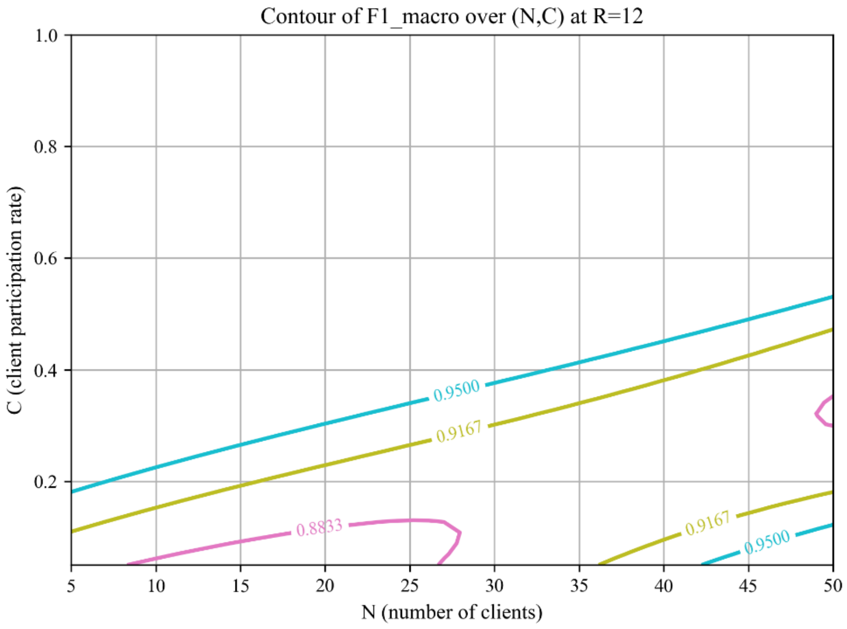


Fig. 14. F1-macro versus N and C (R=12)

As mentioned, Configurations 6 and 13 were deliberately incorporated to evaluate the effects of the internal structure of parameters under the same normalized load. They had a normalized load of 30 and applied different combinations of parameters (Configuration 6 with R=18, C=1, and N=10 and Configuration 13 with R=12, C=0.3, and N=50). According to Table 20, significant differences between the configurations were observed, despite their identical normalized load. Configuration 13 yielded a greater F1-macro (0.8832>0.7838), converged in a smaller number of rounds (8<18), and had a lower communication cost (95.05 MB<142.57 MB). As a result, although it can serve as a measure of the overall computational and communicational pressure, the normalized load is not sufficient for explaining convergence quality and communication efficiency. The parameter combination (N, C, and R) can independently affect gradient stability, data diversity, and communication cost. In particular, the combination of more nodes with a moderate participation rate and fewer training rounds (Configuration 13) was more effective than the combination of fewer nodes with full node participation and many training rounds in establishing a robust trade-off between accuracy, convergence rate, and communication efficiency. Therefore, it can be concluded that an optimized combination of parameters is as important as overall load management in the design of federated learning systems.

Table 20. Comparison of Configuration 6 versus Configuration 13 at a given normalized load

Simulation ID	N	C	R	Normalized Computational Load Prediction	F1-Macro	Rounds to Convergence	Comm. Cost (MB)
6	10	1	18	30	0.7838	18	142.57
13	50	0.3	12	30	0.8832	8	95.05

6.6.6. Real-Time Feasibility and Resource-Efficiency Analysis

To evaluate the practical deployability of the proposed lightweight 1D-CNN in dew-computing environments, its runtime, memory footprint, and compatibility with resource-constrained hardware are examined. The analysis focuses on three key aspects: inference latency, memory consumption, and integration with lightweight inference engines.

The model employs a single Conv1D layer followed by a small dense layer, avoiding recurrent structures (LSTM/GRU) that introduce sequential dependencies and longer inference times. The forward-pass complexity is $O(F \times K \times L)$ for the convolution and $O(D_{in} \times D_{out})$ for the dense layer, where F is the number of filters, K the kernel size, L the length of the input feature vector (10–40 after MI-based selection), and D_{in}, D_{out} the input/output dimensions of the dense layer. With typical values ($F = 32, K = 3, L \approx 30, D_{in} = 32$, and $D_{out} = 2,6, or\ 15$), the total floating-point operations per sample remain below 5 kFLOPs, enabling inference within a few milliseconds on modest CPUs (e.g., Intel Atom or ARM Cortex-A series).

The total number of trainable parameters ranges from approximately $1\,k\text{–}\,2\,k$ for the binary scenario to $3\,k\text{–}\,5\,k$ for the fifteen-class scenario. Storing the model as a 32-bit float TensorFlow Lite flat-buffer yields a size of $\approx\ 12\text{–}\ 20\ KB$, comfortably fitting in the SRAM of typical dew-layer microcontrollers (e.g., ESP32, Raspberry Pi Pico). During inference, the peak RAM usage (activations plus intermediate buffers) is below 50 KB, leaving ample room for other dew-layer tasks.

The purely feed-forward structure and small parameter count make the model directly convertible to TF-Lite, ONNX Runtime Mobile, or similar edge-optimized frameworks. Post-training quantization (e.g., INT8) can reduce the model size by 4 times and accelerate inference by 2–3 times with negligible accuracy loss. Pruning (removing redundant filters/weights) can additionally shrink the model to under 10 KB without sacrificing detection performance.

In the federated setting, each local update involves a single epoch over the node's data subset, which, thanks to the shallow architecture, completes in seconds even on low-power dew hardware. The aggregated model delta transmitted per round is merely the difference of the small parameter set (a few kilobytes), drastically cutting communication costs compared to raw-data transmission. This design ensures that the overall training loop respects the limited bandwidth and intermittent connectivity of dew networks.

6.6.7. Proposed model versus earlier frameworks in federated learning

The performance of the proposed model in federal learning is compared to earlier works in this section. However, this comparison encounters structural limitations since each work had different definitions of factors, such as the number of clients, client participation rate in each round, the number of local training rounds, and data split strategy (IID and non-IID). These setting differences do not allow for a direct comparison under completely identical conditions. Therefore, Table 21 provides only an overall representation of models in similar scenarios rather than a solid comparison.

Previous studies [29–31] focused on specific aspects of model design and/or experimental conditions and reported considerable results. However, they incorporated a limited range of scenarios or lacked sufficient mechanisms to control communication costs and stability under real-life IIoT conditions. In contrast, the proposed model yielded better results based on three key factors. First, the intrinsic characteristics of the model were designed such that not only computational simplicity would be retained but also effective patterns could be extracted from massive, diverse data. Second, the selection of aggregation algorithms based on the complexity level of the system enhanced convergence and stability. Third, the adoption of Bayesian optimization to tune hyperparameters enabled efficient exploration in a large search space and provided combinations that established an optimal trade-off between accuracy, training time, and communication cost.

Table 21. Proposed model versus earlier works under three classification scenarios in federated learning

Classifier	FL Settings	Best Accuracy (%)			Best F1-Weighted (%)		
		2-class	6-class	15-class	2-class	6-class	15-class
Unsupervised Deep Auto-Encoder [29]	N=2 R=5	99.8	-	-	NR	-	-
CNN [30]	$N \in \{3,9,15\}$ R=50	-	-	90.19	-	-	NR
RNN [30]	$N \in \{3,9,15\}$ R=50	-	-	91.26	-	-	NR
DNN [31]	N=10 R=25	100	98.80	98.80	100	86.23	86.23
Proposed Solution	$N \in \{5,10,20,50\}$ $C \in \{0.05,0.1,0.2,0.3,0.5,1\}$ $R \in \{8,10,12,15,18\}$	100	97.45	96.67	100	97.40	96.72

7. Conclusion

The findings demonstrated that the proposed 1D-CNN architecture serves as a lightweight and effective intrusion detection framework in IIoT systems and dew computing in both centralized and federated learning settings. This study developed an optimal technique for intrusion detection in dew settings, which established a trade-off between high accuracy, a short inference time, and an optimal computational cost. The proposed model was found to outperform earlier works in terms of quantitative indices and can be deployed in real-life industrial settings from an operational perspective. The proposed model consistently yielded high accuracy, precision, recall, and F1-score on the Edge-IIoTset under binary, six-class, and fifteen-class scenarios, while it had a remarkably short inference time. This is, in particular, important for implementation in limited-source systems where real-time responsiveness is required. Two major factors had the greatest contributions to model quality: (1) the use of the Conv1D layer allowed for extracting local patterns from the time-series data of networks and effectively detected the discriminative structures of attacks, and (2) the Dropout and optimal hyperparameter tuning prevented overfitting and enabled stable model performance, even under imbalanced data conditions. As a result, the proposed model provided a more robust and effective trade-off between accuracy, time, and computational cost compared to basic architectures, e.g., DNN, and even heavier models, e.g., CNN+LSTM+GRU. In the federated learning setting, the proposed model enjoyed key advantages, i.e., the elimination of raw data exchange and the reduction of the communication cost, while maintaining prediction accuracy. Sensitivity analysis on key parameters, including the number of nodes N, participation rate C, and the number of training rounds R, revealed that the system achieved stable convergence in a wide range of configurations, even though its growing efficiency decreased above an optimal level. This finding is essential for system designers since the selection of optimal configurations helps ensure quality and control the communication cost and training time. The Pareto frontier analysis further demonstrated that points can be identified to establish an effective trade-off between performance and efficiency. While the proposed lightweight 1D-CNN is designed as a general-purpose intrusion detection model for dew environments, future work may explore the integration of specialized or hybrid models for attacks with strong sequential characteristics, such as MitM, Password, SQL Injection, and Malware. For instance, protocol-aware or temporal sequence models could be combined with the proposed framework to enhance the detection of specific intrusion behaviors, provided that sufficient representative data are available. Moreover, the adoption of techniques such as model pruning, quantization, or knowledge distillation may enable the deployment of the model in even more limited devices. Ultimately, non-IID data conditions and explainability mechanism development can be addressed to enhance trustworthiness and transparency in system decision-making.

Alireza Fadaei: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Behrang Barekatain:** Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Supervision, Validation, Writing - review & editing.

Funding: There was no Funding.

Data and Code Availability:
The source code implemented for the experiments and results presented in this manuscript has been made publicly available for reproducibility purposes:

- Centralized version
- Federated version

References

1. Mukherjee A, De D, Dey N, Crespo RG, Song HH. Dew as a Service for Intermittently Connected Internet of Drone Things. In: Internet of Things. Springer Science and Business Media Deutschland GmbH; 2024. p. 241–60.
2. Khatua S, Manerba D, Maity S, De D. Dew Computing-Based Sustainable Internet of Vehicular Things. In: Internet of Things [Internet]. Springer Science and Business Media Deutschland GmbH; 2024. p. 181–205. Available from: https://link.springer.com/10.1007/978-981-99-4590-0_9
3. Wang Y. Definition and Categorization of Dew Computing. Open Journal of Cloud Computing (OJCC) [Internet]. 2016;3(1):1–7. Available from: www.ronpub.com/ojcc

4. Ray PP. An Introduction to Dew Computing: Definition, Concept and Implications. IEEE Access [Internet]. 2018 Nov 16;6:723–37. Available from: http://ieeexplore.ieee.org/document/8114187/
5. Roy S, Panda D, Kim BG, Bairagi P, Mondal T, Arosh S, et al. DewMetrics: Demystification of the Dew Computing in Sustainable Internet of Things. In: Internet of Things. Springer Science and Business Media Deutschland GmbH; 2024. p. 3–39.
6. Bhattacharyya A, Sau A, Ganguly M. Dew-Computing in Future Telerobotic Applications: An Exploration. In: Internet of Things. Springer Science and Business Media Deutschland GmbH; 2024. p. 207–23.
7. Singh P, Kaur A, Aujla GS, Batth RS, Kanhere S. DaaS: Dew Computing as a Service for Intelligent Intrusion Detection in Edge-of-Things Ecosystem. IEEE Internet Things J [Internet]. 2021 Aug 15;8(16):12569–77. Available from: https://ieeexplore.ieee.org/document/9216004/
8. Singh M, Mishra D. Security and Privacy Aspects of Authorized and Secure Communications in Dew-Assisted IoT Systems. In: De D, Roy S, editors. Dew Computing: The Sustainable IoT Perspectives [Internet]. Singapore: Springer Nature Singapore; 2024. p. 79–101. Available from: https://doi.org/10.1007/978-981-99-4590-0_4
9. Das S, Naskar A, Majumder R, De D, Ahmadpour SS. DewIDS: Dew Computing for Intrusion Detection System in Edge of Things. In: De D, Roy S, editors. Dew Computing: The Sustainable IoT Perspectives [Internet]. Singapore: Springer Nature Singapore; 2024. p. 133–48. Available from: https://link.springer.com/10.1007/978-981-99-4590-0_7
10. Moussa MM, Alazzawi L. Cyber Attacks Detection based on Deep Learning for Cloud-Dew Computing in Automotive IoT Applications. In: 2020 IEEE International Conference on Smart Cloud (SmartCloud) [Internet]. IEEE; 2020. p. 55–61. Available from: https://ieeexplore.ieee.org/document/9265942/
11. Meng W, Li W, Xiang Y, Choo KKR. A bayesian inference-based detection mechanism to defend medical smartphone networks against insider attacks. Journal of Network and Computer Applications [Internet]. 2017 Jan 15;78:162–9. Available from: https://linkinghub.elsevier.com/retrieve/pii/S1084804516302818
12. Li J, Sun L, Yan Q, Li Z, Srisa-an W, Ye H. Significant Permission Identification for Machine-Learning-Based Android Malware Detection. IEEE Trans Industr Inform [Internet]. 2018 Jul 1;14(7):3216–25. Available from: https://ieeexplore.ieee.org/document/8255798/
13. Yassin W, Udzir NI, Muda Z, Abdullah A, Abdullah MT. A Cloud-based Intrusion Detection Service framework. In: Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec) [Internet]. IEEE; 2012. p. 213–8. Available from: http://ieeexplore.ieee.org/document/6246098/
14. Alharkan T, Martin P. IDSaaS: Intrusion Detection System as a Service in Public Clouds. In: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012) [Internet]. IEEE; 2012. p. 686–7. Available from: http://ieeexplore.ieee.org/document/6217486/
15. Ding Y, Chen S, Xu J. Application of Deep Belief Networks for opcode based malware detection. In: 2016 International Joint Conference on Neural Networks (IJCNN) [Internet]. IEEE; 2016. p. 3901–8. Available from: http://ieeexplore.ieee.org/document/7727705/
16. Zhao G, Zhang C, Zheng L. Intrusion Detection Using Deep Belief Network and Probabilistic Neural Network. In: 22017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on

Embedded and Ubiquitous Computing (EUC) [Internet]. IEEE; 2017. p. 639–42. Available from: http://ieeexplore.ieee.org/document/8005871/

17. Kaiser J, Zimmerer D, Tieck JCV, Ulbrich S, Roennau A, Dillmann R. Spiking Convolutional Deep Belief Networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) [Internet]. Springer Verlag; 2017. p. 3–11. Available from: http://link.springer.com/10.1007/978-3-319-68612-7_1
18. Tian Q, Han D, Li KC, Liu X, Duan L, Castiglione A. An intrusion detection approach based on improved deep belief network. Applied Intelligence [Internet]. 2020 Oct 6;50(10):3162–78. Available from: https://link.springer.com/10.1007/s10489-020-01694-4
19. Konatham B, Simra T, Amsaad F, Ibrahem MI, Jhanjhi NZ. A Secure Hybrid Deep Learning Technique for Anomaly Detection in IIoT Edge Computing [Internet]. 2024. Available from: https://www.techrxiv.org/users/662346/articles/706122-a-secure-hybrid-deep-learning-technique-for-anomaly-detection-in-iiot-edge-computing?commit=e4d90929f9887c6d4c70d2e54cde6348880c8865
20. Kilichev D, Turimov D, Kim W. Next–Generation Intrusion Detection for IoT EVCS: Integrating CNN, LSTM, and GRU Models. Mathematics [Internet]. 2024 Feb 14;12(4):571. Available from: https://www.mdpi.com/2227-7390/12/4/571
21. Ferrag MA, Friha O, Hamouda D, Maglaras L, Janicke H. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. IEEE Access [Internet]. 2022;10:40281–306. Available from: https://ieeexplore.ieee.org/document/9751703/
22. Singh P, Gaba GS, Kaur A, Hedabou M, Gurtov A. Dew-Cloud-Based Hierarchical Federated Learning for Intrusion Detection in IoMT. IEEE J Biomed Health Inform [Internet]. 2023 Feb 1;27(2):722–31. Available from: https://ieeexplore.ieee.org/document/9826463/
23. Liu Q, Wang G, Liu X, Peng T, Wu J. Achieving reliable and secure services in cloud computing environments. Computers & Electrical Engineering [Internet]. 2017 Apr 1;59:153–64. Available from: https://linkinghub.elsevier.com/retrieve/pii/S004579061630502X
24. Tareq I, Elbagoury BM, El-Regaily S, El-Horbaty ESM. Analysis of ToN-IoT, UNW-NB15, and Edge-IIoT Datasets Using DL in Cybersecurity for IoT. Applied Sciences [Internet]. 2022 Sep 23;12(19):9572. Available from: https://www.mdpi.com/2076-3417/12/19/9572
25. Ding W, Abdel-Basset M, Mohamed R. DeepAK-IoT: An effective deep learning model for cyberattack detection in IoT networks. Inf Sci (N Y) [Internet]. 2023 Jul 1;634:157–71. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0020025523003511
26. Sadhwani S, Modi UK, Muthalagu R, Pawar PM. SmartSentry: Cyber Threat Intelligence in Industrial IoT. IEEE Access [Internet]. 2024;12:34720–40. Available from: https://ieeexplore.ieee.org/document/10456888/
27. Laiq F, Al-Obeidat F, Amin A, Moreira F. Securing edge-IIoT networks: a comprehensive ensemble-based DDoS detection system. Journal of Physics: Complexity [Internet]. 2025 Jun 1;6(2):025005. Available from: https://iopscience.iop.org/article/10.1088/2632-072X/ad506b
28. Kumar V, Kumar V. PETDA2C-EC: a privacy-enhancing technique to detect attacks against confidentiality in edge computing. J Reliab Intell Environ [Internet]. 2025 Jun 9;11(2):8. Available from: https://link.springer.com/10.1007/s40860-025-00248-0

29. Thamar N, Sharara HSE. Unsupervised anomalies detection in IIoT edge devices networks using federated learning. 2023 Aug 23; Available from: http://arxiv.org/abs/2308.12175
30. Rashid MM, Khan SU, Eusufzai F, Redwan MdA, Sabuj SR, Elsharief M. A Federated Learning-Based Approach for Improving Intrusion Detection in Industrial Internet of Things Networks. Network [Internet]. 2023 Jan 30;3(1):158–79. Available from: https://www.mdpi.com/2673-8732/3/1/8
31. Popoola SI, Imoize AL, Hammoudeh M, Adebisi B, Jogunola O, Aibinu AM. Federated Deep Learning for Intrusion Detection in Consumer-Centric Internet of Things. IEEE Transactions on Consumer Electronics [Internet]. 2024 Feb 1;70(1):1610–22. Available from: https://ieeexplore.ieee.org/document/10373897/
32. Mallidi SKR, Ramisetty RR. A multi-level intrusion detection system for industrial IoT using bowerbird courtship-inspired feature selection and hybrid data balancing. Discover Computing [Internet]. 2025 Jun 7;28(1):109. Available from: https://link.springer.com/10.1007/s10791-025-09632-z
33. Tawfik M. Optimized intrusion detection in IoT and fog computing using ensemble learning and advanced feature selection. Singh R, editor. PLoS One [Internet]. 2024 Aug 1;19(8):e0304082. Available from: https://dx.plos.org/10.1371/journal.pone.0304082
34. Roy S, Sankaran S, Zeng M. Green Intrusion Detection Systems: A Comprehensive Review and Directions. Sensors [Internet]. 2024 Aug 26;24(17):5516. Available from: https://www.mdpi.com/1424-8220/24/17/5516
35. Yang K, Wang J, Li M. An improved intrusion detection method for IIoT using attention mechanisms, BiGRU, and Inception-CNN. Sci Rep [Internet]. 2024 Aug 20;14(1):19339. Available from: https://www.nature.com/articles/s41598-024-70094-2
36. Rahman MM, Shakil S Al, Mustakim MR. A survey on intrusion detection system in IoT networks. Cyber Security and Applications [Internet]. 2025 Dec;3:100082. Available from: https://linkinghub.elsevier.com/retrieve/pii/S2772918424000481
37. Le K, Luong-Ha N, Nguyen-Duc M, Le-Phuoc D, Do C, Wong KS. Exploring the Practicality of Federated Learning: A Survey Towards the Communication Perspective. 2024 May 30; Available from: http://arxiv.org/abs/2405.20431
38. Makris I, Karampasi A, Radoglou-Grammatikis P, Episkopos N, Iturbe E, Rios E, et al. A comprehensive survey of Federated Intrusion Detection Systems: Techniques, challenges and solutions. Comput Sci Rev [Internet]. 2025 May;56:100717. Available from: https://linkinghub.elsevier.com/retrieve/pii/S157401372400100X
39. Buyuktanir B, Altinkaya Ş, Karatas Baydogmus G, Yildiz K. Federated learning in intrusion detection: advancements, applications, and future directions. Cluster Comput [Internet]. 2025 Sep 4;28(7):473. Available from: https://link.springer.com/10.1007/s10586-025-05325-w
40. Haque MdE, Polash MdSH, Simla MAIS, Hossain MA, Jahan S. Enhancing IoT Cyber Attack Detection in the Presence of Highly Imbalanced Data. 2025 May 15; Available from: http://arxiv.org/abs/2505.10600
41. Shanmugam V, Razavi-Far R, Hallaji E. Addressing Class Imbalance in Intrusion Detection: A Comprehensive Evaluation of Machine Learning Approaches. Electronics (Basel) [Internet]. 2024 Dec 27;14(1):69. Available from: https://www.mdpi.com/2079-9292/14/1/69
42. Tahir M, Abdullah A, Udzir NI, Kasmiran KA. A novel approach for handling missing data to enhance network intrusion detection system. Cyber Security and Applications [Internet]. 2025 Dec;3:100063. Available from: https://linkinghub.elsevier.com/retrieve/pii/S2772918424000298

43. Chen X, Gong Z, Huang D, Jiang N, Zhang Y. Overcoming Class Imbalance in Network Intrusion Detection: A Gaussian Mixture Model and ADASYN Augmented Deep Learning Framework. In: Proceedings of the 2024 4th International Conference on Internet of Things and Machine Learning [Internet]. New York, NY, USA: ACM; 2024. p. 48–53. Available from: https://dl.acm.org/doi/10.1145/3697467.3697595
44. Skala KA, Davidovi D, Afgan EA, Sovi I. Scalable Distributed Computing Hierarchy: Cloud, Fog and Dew Computing *. Open Journal of Cloud Computing (OJCC) [Internet]. 2015;2(1):16–24. Available from: http://creativecommons.org/licenses/by/3.0/
45. Pandey VK, Sahu D, Prakash S, Rathore RS, Dixit P, Hunko I. A lightweight framework to secure IoT devices with limited resources in cloud environments. Sci Rep [Internet]. 2025 Jul 17;15(1):26009. Available from: https://www.nature.com/articles/s41598-025-09885-0
46. Alzahrani H, Sheltami T, Barnawi A, Imam M, Yaser A. A Lightweight Intrusion Detection System Using Convolutional Neural Network and Long Short-Term Memory in Fog Computing. Computers, Materials & Continua. 2024;80(3):4703–28.
47. Sharifian Z, Barekatain B, Ariza Quintana A, Beheshti Z, Safi-Esfahani F. LOADng-AT: a novel practical implementation of hybrid AHP-TOPSIS algorithm in reactive routing protocol for intelligent IoT-based networks. J Supercomput [Internet]. 2022 May 17;78(7):9521–69. Available from: https://link.springer.com/10.1007/s11227-021-04256-8
48. Tariq N, Alsirhani A, Humayun M, Alserhani F, Shaheen M. A fog-edge-enabled intrusion detection system for smart grids. Journal of Cloud Computing. 2024 Dec 1;13(1).
49. Ayad AG, Sakr NA, Hikal NA. Fog-empowered anomaly detection in IoT networks using one-class asymmetric stacked autoencoder. Cluster Comput [Internet]. 2025 Sep 23;28(8):550. Available from: https://link.springer.com/10.1007/s10586-025-05234-y
50. Alves J, Sousa P, Cruz T, Mendes J. A review of architecture features for distributed and resilient industrial cyber–physical systems. J Manuf Syst [Internet]. 2025 Oct;82:1069–90. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0278612525001876
51. Majidi F, Khayyambashi MR, Barekatain B. HFDRL: An Intelligent Dynamic Cooperate Cashing Method Based on Hierarchical Federated Deep Reinforcement Learning in Edge-Enabled IoT. IEEE Internet Things J [Internet]. 2022 Jan 15;9(2):1402–13. Available from: https://ieeexplore.ieee.org/document/9447004/
52. Pfeiffer K, Rapp M, Khalili R, Henkel J. Federated Learning for Computationally Constrained Heterogeneous Devices: A Survey. ACM Comput Surv [Internet]. 2023 Dec 31;55(14s):1–27. Available from: https://dl.acm.org/doi/10.1145/3596907
53. Teixeira R, Almeida L, Antunes M, Gomes D, Aguiar RL. Efficient training: Federated learning cost analysis. Big Data Research. 2025 May;40:100510.
54. Hancock JT, Khoshgoftaar TM. Survey on categorical data for neural networks. J Big Data [Internet]. 2020 Dec 10;7(1):28. Available from: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00305-w
55. Poslavskaya E, Korolev A. Encoding categorical data: Is there yet anything “hotter” than one-hot encoding? 2023 Dec 28; Available from: http://arxiv.org/abs/2312.16930
56. Aurélien G. Chapter 2: End-to-End Machine Learning Project. In: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. 2nd ed. Sebastopol, CA: O’Reilly Media; 2019. p. 35–84.
57. Cover TM., Thomas JA. Elements of information theory. Wiley-Interscience; 2006. 748 p.

58. Kraskov A, Stögbauer H, Grassberger P. Estimating mutual information. Phys Rev E. 2004 Jun 23;69(6):066138.
59. Ross BC. Mutual Information between Discrete and Continuous Data Sets. PLoS One. 2014 Feb 19;9(2):e87357.
60. Hanchuan Peng, Fuhui Long, Ding C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans Pattern Anal Mach Intell. 2005 Aug;27(8):1226–38.
61. Goodfellow I, Bengio Y, Courville A. Convolutional Networks. In: Deep Learning [Internet]. 2016 [cited 2025 Aug 18]. p. 330–71. Available from: https://moodle2.units.it/pluginfile.php/389100/mod_folder/content/0/DL_Goodfellow_Bengio_Courville%20%281%29.pdf?forcedownload=1
62. Zhou DX. Universality of deep convolutional neural networks. Appl Comput Harmon Anal. 2020 Mar;48(2):787–94.
63. Qazi EUH, Almorjan A, Zia T. A One-Dimensional Convolutional Neural Network (1D-CNN) Based Deep Learning System for Network Intrusion Detection. Applied Sciences [Internet]. 2022 Aug 10;12(16):7986. Available from: https://www.mdpi.com/2076-3417/12/16/7986
64. Wang Z, Chen H, Yang S, Luo X, Li D, Wang J. A lightweight intrusion detection method for IoT based on deep learning and dynamic quantization. PeerJ Comput Sci [Internet]. 2023 Sep 22;9:e1569. Available from: https://peerj.com/articles/cs-1569
65. Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding deep learning requires rethinking generalization. 2017 Feb 26; Available from: http://arxiv.org/abs/1611.03530
66. McMahan HB, Moore E, Ramage D, Hampson S, Arcas BA y. Communication-Efficient Learning of Deep Networks from Decentralized Data. 2023 Jan 26; Available from: http://arxiv.org/abs/1602.05629
67. Liu W, Chen L, Chen Y, Zhang W. Accelerating Federated Learning via Momentum Gradient Descent. 2019 Oct 9; Available from: http://arxiv.org/abs/1910.03197
68. Reddi S, Charles Z, Zaheer M, Garrett Z, Rush K, Konečný J, et al. Adaptive Federated Optimization. 2021 Sep 8; Available from: http://arxiv.org/abs/2003.00295
69. Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. Federated Optimization in Heterogeneous Networks. 2020 Apr 21; Available from: http://arxiv.org/abs/1812.06127
70. Pillutla K, Kakade SM, Harchaoui Z. Robust Aggregation for Federated Learning. IEEE Transactions on Signal Processing [Internet]. 2022 Jan 17;70:1142–54. Available from: https://ieeexplore.ieee.org/document/9721118/
71. Li T, Sahu AK, Talwalkar A, Smith V. Federated Learning: Challenges, Methods, and Future Directions. IEEE Signal Process Mag [Internet]. 2019 Aug 21;37(3):50–60. Available from: http://arxiv.org/abs/1908.07873
72. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, et al. Advances and Open Problems in Federated Learning. 2021 Mar 9; Available from: http://arxiv.org/abs/1912.04977
73. Tan M, Le Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning [Internet]. PMLR; 2019. p. 6105–14. (Proceedings of Machine Learning Research; vol. 97). Available from: https://proceedings.mlr.press/v97/tan19a.html