The Journal of Supercomputing
https://doi.org/10.1007/s11227-024-06153-2

A federated learning approach to network intrusion detection using residual networks in industrial IoT networks

Nisha Chaurasia[1] · Munna Ram[1] · Priyanka Verma[3] · Nakul Mehta[1] · Nitesh Bharot[2]

Accepted: 17 April 2024

Abstract

This paper introduces a sophisticated approach to network security, with a primary emphasis on utilizing deep learning for intrusion detection. In real-world scenarios, the high dimensionality of training data poses challenges for simple deep learning models and can lead to vanishing gradient issues with complex neural networks. Additionally, uploading network traffic data to a central server for training raises privacy concerns. To tackle these issues, the paper introduces a Residual Network (ResNet)-based deep learning model trained using a federated learning approach. The ResNet effectively tackles the vanishing gradient problem, while federated learning enables multiple Internet Service Providers (ISPs) or clients to engage in joint training without sharing their data with third parties. This approach enhances accuracy through collaborative learning while maintaining privacy. Experimental results on the X-IIoTID dataset indicate that the proposed model outperforms conventional deep learning and machine learning methods in terms of accuracy and other metrics used for evaluation. Specifically, the proposed methodology achieved 99.43% accuracy in a centralized environment and 99.16% accuracy in a federated environment.

Keywords Intrusion detection (IDS) · Industrial IoT · Deep learning · Residual networks · ML · Industry 4.0

1 Introduction

In the continuously evolving digital landscape of today, the growth of the internet brings increasingly complex security challenges, including various network security threats like viruses, malware, and ransomware. Detecting and mitigating these attacks is crucial to protect a network's availability, confidentiality, and integrity [1]. For this, an Intrusion Detection System (IDS) serves as a tool that

Extended author information available on the last page of the article

Published online: 15 May 2024

analyses ongoing ingress (incoming traffic), alerts users of potential anomalies, and takes preventive measures [2]. Intrusion detection is essentially a classification problem that distinguishes different types of network attacks based on network traffic characteristics [3]. Therefore, enhancing the accuracy of these IDS becomes a vital task [4].

Deep Learning (DL) has recently gained prominence as a promising approach to intrusion detection research. DL involves the usage of multi-layered artificial neural networks to extract hierarchical data representations, drawing inspiration from the design and function of human brain. DL algorithms analyze large, complex datasets to obtain optimal features from input data, enabling systems to make accurate predictions and decisions regarding new data. DL has facilitated significant advancements in image and speech recognition, medical diagnosis, and autonomous systems. Its capability to learn intricate patterns in data renders it a valuable tool for addressing complex challenges across diverse fields.

However, the vanishing gradient problem is a crucial consideration in DL models. Adding more layers with certain activation functions makes training a neural network increasingly difficult, as the gradients of the loss function eventually approach zero. Activation functions like sigmoid and hyperbolic tangent yield gradients within the range (0, 1), and back-propagation calculates gradients using the chain rule. When employing an activation function like sigmoid across 'n' hidden layers, 'n' small derivatives are multiplied together. Consequently, as the gradient rapidly decreases when propagating toward lower layers, the initial layers' weights and biases are inadequately updated after each training session. Given that these initial layers often rely on the recognition of incoming data's fundamental components, this issue can result in overall network inaccuracy. Several potential solutions to the vanishing gradient problem include using the ReLU activation function, gradient clipping, Long Short-Term Memory (LSTM), and ResNet. Hence, we propose a modified version of the ResNet, capable of handling numerous neural network layers without encountering the vanishing gradient problem, thereby ensuring high accuracy.

Observations in Fig. 1 indicate that DL-based classification requires a substantial volume of data for effective performance. It also indicates that the classification model's accuracy increases with the growth of training data. However, obtaining such an extensive dataset might pose a challenge for individual Internet Service Providers (ISPs).

This issue can be resolved by having multiple ISPs upload their data to a central server (collaborative learning). However, this raises privacy concerns. As an alternative, we propose using Federated Learning (FL) [5], which offers the advantages of collaborative learning while also ensuring privacy. FL [6] allows training on local datasets without sharing data with the central entity and only sharing the training results weights with a central server.

In summary, this study introduces an innovative approach for IDS in Industrial Internet of Things (IIoT) networks by leveraging ResNet and FL. The key contributions of this work are as follows:

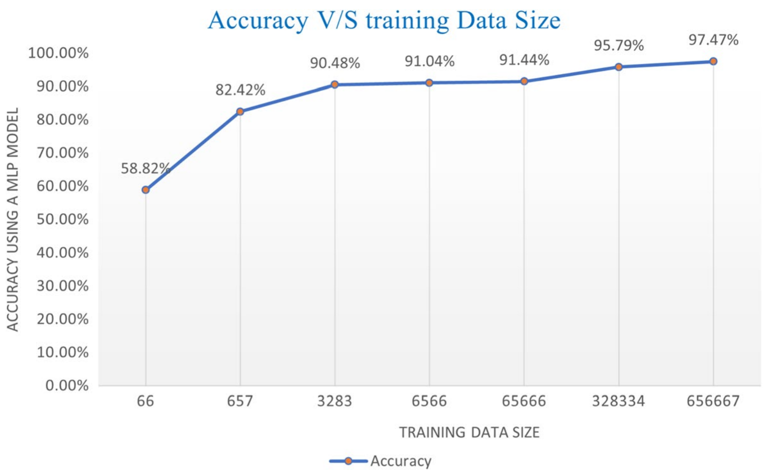

Fig. 1 Large amount of data are required for more accurate classification in DL

- It addresses the challenge of centralized data aggregation and analysis by integrating the FL algorithm. This facilitates collaborative model training among various industries while safeguarding the privacy and security of the clients.
- The novelty of this work involves adapting the ResNet-based DL model for intrusion detection. This modification specifically addresses the vanishing gradient problem, ensuring the model's effectiveness in handling highly dimensional datasets. The enhanced model exhibits a high accuracy rate in identifying network intrusions, showcasing its robust performance in complex scenarios.
- Leveraging ResNet in our DL model allows for the effective utilization of a substantial number of layers in a neural network, mitigating challenges posed by vanishing gradient problem. This ensures that the model performs optimally, maintaining stability and efficiency throughout its architecture.

The remaining paper is organized as: Section 2 provides a literature review and discusses relevant work. Section 3 elucidates the proposed approach in detail. Section 4 presents the experimental findings, demonstrating the effectiveness of our approach. Finally, Sect. 5 offers the conclusion and discussion of our findings.

2 Related work

Machine Learning (ML) techniques have proven to be valuable in mitigating attacks and assisting network administrators in preventing intrusions. Several methods, such as K-Nearest Neighbors (KNN) [7, 8], Random Forest (RF) [9, 10], Naïve Bayes (NB) [11, 12], and Support Vector Machine (SVM) [13, 14], have been proposed and employed for intrusion detection in earlier research. Traditional methods primarily focus on feature engineering and selection but are inefficient in detecting intrusions within large datasets in real-world application environments, as described in [15, 16]. These methods are not suitable for high-dimensional, large datasets.

Recently, DL-based intrusion detection has emerged as a new research area. Akashdeep et al. [17] utilized Artificial Neural Networks (ANN) for intrusion detection. Park et al. [18] employed a Recurrent Neural Network (RNN) with N-gram sliding window technique for anomaly detection. Torres et al. [19] also applied RNN for intrusion detection. Wang et al. [20] applied Convolutional Neural Networks (CNN) on traffic data for malware classification, and in [21], they combined the advantages of both RNN and CNN. Al-Qatf et al. [22] provided a Self-Taught Learning (STL) framework-based approach. L. Bontemps et al. [23] proposed a Long Short-Term Memory (LSTM) model for anomaly detection. Additionally, various IDS models using RNN, autoencoder, Deep Convolutional Neural Network (DCNN), and others have been proposed [3].

Nonetheless, most researchers have utilized centralized DL methods to accumulate large-scale datasets, raising privacy concerns. FL addresses this issue. Zhao et al. [24] applied FL for intrusion detection using an LSTM model, demonstrating its potential to overcome privacy challenges while maintaining effective detection capabilities. Priyanka et al. [25] utilized a dual autoencoder-based model in a federated environment for handling Zero-Day Attacks in 5G-enabled IIoT. Table 1 shows the brief comparative state-of-the-art analysis.

However, despite their successes, these approaches are not without their limitations. One significant limitation is the need for large amounts of labeled data for training. ML algorithms rely heavily on data to learn patterns and make predictions, and the quality and quantity of the data can significantly impact the model's performance. Acquiring labeled data can be expensive and time-consuming, particularly for specialized domains where expertise is required to annotate the data accurately. Additionally, centralized methods pose privacy issues, as they often require pooling sensitive data from various sources into a central repository, raising concerns about data security and confidentiality. These centralized approaches can undermine privacy regulations and may deter individuals or organizations from sharing their data for fear of breaches or misuse.

Another issue the models face is the vanishing gradient problem, particularly in deep neural networks with many layers. It occurs during the training process when gradients become exceedingly small as they propagate backward through the network, leading to stagnant or slow learning. As a result, earlier layers in the network receive negligible updates, hindering their ability to learn meaningful representations from the data effectively.

Therefore, we introduce a ResNet-based DL approach that successfully addresses these issues. It effectively handles the vanishing gradient issues, ensuring stable and efficient training. By overcoming this obstacle, it demonstrates significant advancements in the field of DL, paving the way for more robust and scalable models. Additionally, our approach stands out in its novelty through the incorporation of FL, a technique that enables training on decentralized data sources while preserving data privacy. It allows us to leverage insights from diverse datasets without compromising individual data privacy, making it particularly advantageous in sensitive domains such as healthcare and finance.

Table 1 Comparison of related work

Model	Year	Approach	Vanishing gradient problem	Specification
Akashdeep et al. [17]	2017	Artificial neural network	Not evaluated	Reduced number of features, high accuracy
Park et al. [18]	2020	Recurrent neural network	Not evaluated	Distinguish abnormal network packet using cosine distance measure
Torres et al. [19]	2016	Recurrent neural network	Not evaluated	Effective in detecting botnets
ElKashlan et al. [26]	2023	Naïve Bayes and J48 classifier	Not evaluated	Intrusion detection system for IoT Electric Vehicle Charging Stations (EVCSs)
Wang et al. [20]	2017	CNN	Not evaluated	Effective in classifying malware traffic
Wang et al. [21]	2018	DL (HAST-IDS)	Not evaluated	Improved performance using hierarchical spatial–temporal features
M. Al-Qatf et al. [22]	2018	DL (autoencoders)	Not evaluated	Using sparse autoencoder with SVM, high accuracy
Awajan et al. [27]	2023	DL (4-layer fully connected network)	Not evaluated	Plug-and-play intrusion detection system that can detect various types of attacks
L. Bontemps et al. [23]	2016	Recurrent neural network	Not evaluated	Effective in detecting collective anomalies
Zhao et al. [24]	2020	FL (LSTM)	Not evaluated	Preserving privacy of data, effective in detecting unknown attacks
Verma et al. [25]	2023	FL (autoencoder)	Not evaluated	Cyberthreat detection in 5G networks, leveraging FL
Meryem et al. [28]	2023	FL (FedProx and autoencoder)	Not evaluated	High detection accuracy and fewer false alarms
Proposed approach	-	FL (ResNet)	Handled	Provides Collaborative Learning, and also handles vanishing gradient problem smoothly, More Deeper Network

3 Proposed method

3.1 Residual networks

ResNet, proposed by Microsoft Research [29], emerged as the winner of the 2015 ImageNet Large Scale Visual Recognition Competition (ILSVRC) in both classification and object recognition categories. Outperforming Inception v3, the third version of Google's GoogLeNet [30], ResNet is a CNN based on residual blocks. It is 20 times larger compared to AlexNet [31] and VGG-16 [32].

ResNet's unique residual effects from skip connections enable it to have a deeper network compared to other neural networks while preventing the vanishing gradient problem. Unlike typical networks, ResNet's performance does not deteriorate with increased layers; instead, it improves. The design of residual block is depicted by Fig. 2.

x represents the input of a residual block; $F(x)$ represents the output of the ResNet before the second activation function.

In ResNet, skip connections are employed that bypass certain model layers. Due to these skip connections, the output deviates from the standard layer-by-layer processing. Without skip connections, the input x would be multiplied by the layer's weights and a bias term would be joined. However, with the presence of skip connections, even if the given block in Fig. 2 is not learning anything, it will still retain the input information. This is because, at the next block, it utilizes the input x when moving forward and then applies the activation function to sum of input and output from the previous layers.

Here,

$$F(x) = W_2\alpha(W_1x) \tag{1}$$

where W_1 and W_2 stand for the weights of the first and second layers, respectively, and α stands for the rectified linear unit (ReLU) [33]. $\alpha(F(x) + x)$ is result of the residual block. Including this type of skip connection offers the advantage of allowing regularization to bypass any layer that might negatively impact the performance of the architecture.

The skip connections [34] applied in the ResNet model, Identity Block and Bottleneck/Convolutional Block, are shown in Fig. 3. The identity block adds

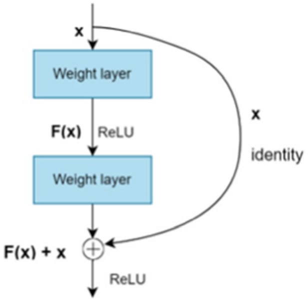

Fig. 2 Design of residual block

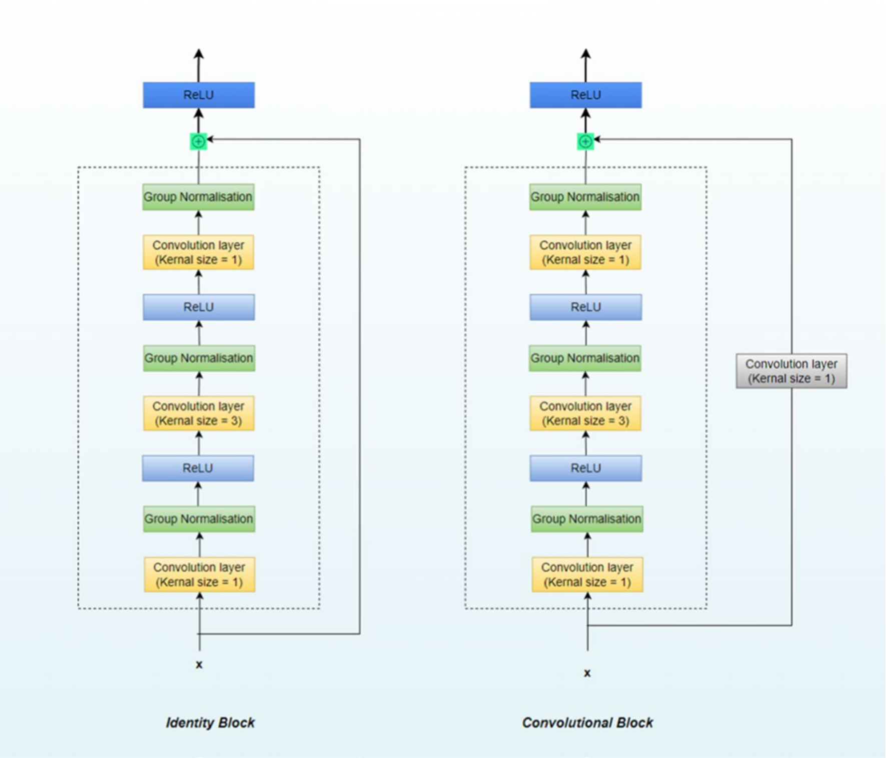
ReLU
Group Normalisation
Convolution layer (Kernal size = 1)
ReLU
Group Normalisation
Convolution layer (Kernal size = 3)
ReLU
Group Normalisation
Convolution layer (Kernal size = 1)
x
Identity Block
ReLU
Group Normalisation
Convolution layer (Kernal size = 1)
ReLU
Group Normalisation
Convolution layer (Kernal size = 3)
ReLU
Group Normalisation
Convolution layer (Kernal size = 1)
Convolution layer (Kernal size = 1)
x
Convolutional Block

Fig. 3 Identity and Convolutional Block in ResNet

the residue directly to the output, whereas the Bottleneck Block incorporates a convolutional layer with dimensions (1 × 1) on the residue before combining it with the output. We will add layers according to Table 2.

3.1.1 Identity Block

In the Identity Block, the first layer consists of a convolutional layer with kernel size 3, stride 2, padding 1, and channels as specified in Table 2. Following this, there is group normalization and a ReLU activation function. The subsequent layer is another convolutional layer, resembling the preceding layer but with a kernel size 3. It is also succeeded by a group normalization layer and ReLU. The final layer is a convolutional layer with the same parameters as the first layer but with an increased number of channels. This is followed by group normalization, and the output ($F(x)$) is then added to original input (x). The combined result is provided as input to the next block after applying the ReLU activation function to it.

Table 2 Architecture of the proposed residual network

Layer name	Layer details			
Conv1	Conv1D, kernel size = 7, filters = 8, and strides = 2			
Conv2_x	Max pool, pool size = 3 and strides = 2			
	Layers in each Block			**Block details**
	Kernel size	**Filters**	**Layer name**	• First Block is the Convolutional Block with these three layers in the main path and Conv1D layer skip connection having kernel size = 1, filters = 32, and strides = 2
	1	8	Conv1D	• Now add two Identity Blocks with the same three layers in the main path
	3	8	Conv1D	
	1	32	Conv1D	
Conv3_x	**Layers in each Block**			**Block details**
	Kernel size	**Filters**	**Layer name**	• First Block is the Convolutional Block with these three layers in the main path and Conv1D layer skip connection having kernel size = 1, filters = 64, and strides = 2
	1	16	Conv1D	• Now add three Identity Blocks with the same three layers in the main path
	3	16	Conv1D	
	1	64	Conv1D	
Conv4_x	**Layers in each Block**			**Block details**
	Kernel size	**Filters**	**Layer name**	• First Block is the Convolutional Block with these three layers in the main path and Conv1D layer skip connection having kernel size = 1, filters = 128, and strides = 2
	1	32	Conv1D	• Now add five Identity Blocks with the same three layers in the main path
	3	32	Conv1D	
	1	128	Conv1D	
Conv5_x	**Layers in each Block**			**Block details**
	Kernel size	**Filters**	**Layer name**	• First Block is the Convolutional Block with these three layers in the main path and Conv1D layer skip connection having kernel size = 1, filters = 256, and strides = 2
	1	64	Conv1D	• Now add three Identity Blocks with the same three layers in the main path
	3	64	Conv1D	

Table 2 (continued)

Layer name	Layer details		
	1	256	Conv1D
Average pooling	Average pooling layer, pool size = 2		
Flatten layer	Flatten layer		
Output layer	Dense layer, units = number of labels in class and activation = Softmax		

3.1.2 Convolutional Block

In the Convolutional Block, the architecture of the main path is the same as that of the Identity Block, as illustrated in Table 2 and Fig. 3. However, there is a modification in the skip connection. In the Identity Block, the input and output share the same shape, allowing for the direct addition of the input to the output. In contrast, the Convolutional Block requires a transformation to accommodate the change in shape between the input and output.

To illustrate this with an example, consider the first Convolutional Block of model (i.e., Conv2_x Convolutional Block), as detailed in Table 2. The input shape is $(N \times M)$, where N represents the number of features in our input dataset, and M denotes the filters of the first layer (Conv1), i.e., 8. Therefore, the input shape is $(N \times 8)$, while the output shape is $(N \times 32)$, as there are 32 channels in the last convolutional layer of the block. To accommodate this change in shape, we introduce a convolutional layer with a kernel size 1, 2 strides, and 32 channels in the skip connection. This results in an output shape of $(N \times 32)$, which can then be added to the main path output.

The output generated by the Convolutional Block, with a shape of $(N \times 32)$, will now serve as the input for the Identity Block. Since the output of the Identity Block's main path also has the same shape, we can directly add a skip connection between them.

3.2 Network architecture

The proposed ResNet architecture is devised with several layers incorporated following the specifications in Table 2. The network initiates with a convolutional layer succeeding the input layer, and a max pooling layer follows. Subsequently, a Conv2_x-type Convolutional Block is introduced, trailed by two Identity Blocks. In a similar manner, a single Convolutional Block precedes three, five, and three Identity Blocks for Conv3_x, Conv4_x, and Conv5_x, respectively. Finally, architectural design is completed with an average pooling layer followed by an output layer.

The intended architecture seeks to improve the model's accuracy and efficiency through the integration of skip connections. These connections facilitate the smooth flow of gradients across the network, allowing for the training of deeper networks without facing issues related to vanishing gradients. The convolutional blocks are crafted to extract intricate features from input data, while identity blocks play a crucial role in maintaining the identity mapping between the input and output of residual blocks.

The ResNet architecture discussed by Microsoft researchers [29] is primarily used for image classification and features a large number of neurons, which increases training costs. Since our work does not involve image datasets, our proposed model is based on the ResNet50 architecture [29]. ResNet50 employs 2D convolutional layers; however, we utilize 1D convolutional layers with smaller kernel sizes and fewer filters to optimize training. Furthermore, we employ group normalization in place of

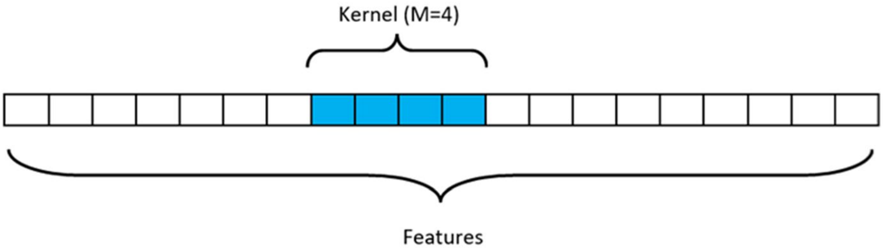

Fig. 4 Convolutional layer (1D)

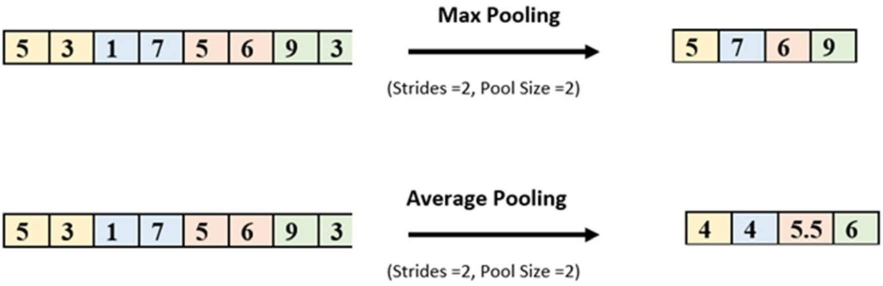

Fig. 5 Pooling layers (1D)

batch normalization as it offers a superior alternative for small-sized batches. Group normalization avoids exploiting the batch dimension and is independent of batch size.

3.3 Introduction to layers used in the network

3.3.1 Convolutional layer

The first layer employed for feature extraction from the input is depicted in Fig. 4. In this layer, a convolution operation is performed between the input and a specific filter of size M. The dot product is calculated by sliding the filter across the input data, matching the size of the filter (M), and computing the value of the product between the filter and the corresponding input segments.

This feature map (output layer) serves as input for other layers.

3.3.2 Pooling layer

The layer illustrated in Fig. 5 is employed to reduce the size of the complex feature map, thereby decreasing computational costs. This is accomplished by minimizing the connections between layers and independently processing each feature map. Various types of pooling layers are at one's disposal, such as max pooling, average pooling, and sum pooling. In max pooling, the feature map's largest element is chosen, whereas in average pooling, the output is the average of all values. Sum pooling, on the other hand, utilizes the total sum of values from the feature map as its output.

3.3.3 Group normalization layer

Group normalization normalizes the inputs of a layer by utilizing the mean and variance calculated within a group of channels. This process accelerates the training of the model while maintaining consistency and stability across different input distributions.

3.4 Federated learning

FL is an emerging domain in field of collaborative ML. Among various approaches in FL, we have employed the foundational method known as FedAvg [35]. In this approach, each participating institution conducts training on their local network traffic data and subsequently uploads the model parameters to a central server. The server computes average of these parameters and disseminates the updated parameters back to the clients for the subsequent round of training. This procedure enables collaborative learning while maintaining the privacy and security of each individual client's data.

In FL, the data of participating institutions remain stored locally and are not shared. Instead, only the models or parameters of the local models are shared after local training. Clients maintain complete control and privacy over their own data, and they can opt out of the FL process at any time. Additionally, this approach addresses potential imbalances in data size among clients through weighted averaging. Clients with smaller datasets contribute less to the updated weights. The steps followed in this FL approach are as follows:

1. The parameters of the global model are initialized and distributed to local clients.
2. Clients incorporate these parameters into their local models and train them on their respective local datasets.
3. After finishing local training, the parameters of each local model are uploaded to the central server.
4. The weighted average of the parameters obtained from various clients is determined by the central server, which then adds these averaged parameters to the global model citeref31.

$$f(w) = \sum_{i=1}^{K} \frac{n_i}{n} F_i(w) \tag{2}$$

where $f(w)$ = average parameters, K = number of clients, n_i = size of dataset at the ith client, n = total data size, and $F(w)$ = parameters received from the ith client.

5. Repeat steps 1–4, N-1 times where *N* is the communication rounds in FL.

4 Results and evaluation

4.1 Experimental setup

This section analyzes and compares efficacy of the proposed framework for IDS in different scenarios. The experiment was conducted on an AMD Ryzen 5 processor, using Python 3.0. Additionally, the FL framework was developed using TensorFlow 2.7 and TensorFlow Federated.

4.2 Evaluation metrics and dataset description

This section entails the evaluation metrics and dataset opted for evaluating the proposed framework. Here, accuracy serves as the primary performance metric along with other metrics such as precision, recall, True Negative Rate (TNR), and loss function values are also utilized. Accuracy is defined as the ratio of correctly classified samples to the total number of samples. Precision is a metric that quantifies the proportion of correctly predicted positive samples among all samples classified as positive. The True Positive Rate (TPR), also referred to as sensitivity, measures the proportion of actual positive samples that were accurately predicted among all actual positive samples. TNR, often known as specificity, represents the proportion of actual negative samples that were accurately predicted in actual negative samples. F1 score is computed as the harmonic mean of precision and recall. The F2 score provides a balanced assessment of both precision and recall, making it a useful metric for evaluating the performance of a classification model, especially in scenarios where one class is more critical to detect than the other.

Extensive results on the latest intrusion detection dataset X-IIoTID [36] which is specifically designed for the industrial settings, with an 80–20 training–testing split, indicate the supremacy of the proposed approach. It contains 820,834 instances, with 421,417 observations for normal and 399,417 for attacks. The dataset includes 59 features such as network traffic features (source and destination address, port, bytes and packets transferred, data transfer rate, etc.) and features related to the edge gateway’s resources (CPU and memory loads, I/O activities, average and standard waiting time, user time, idle time, etc.). The dataset has 19 classes: 18 attacks and 1 normal that are processed to two major classes, attack and normal. These training data are distributed among 10 clients while maintaining equal distribution for each client. According to [36], the X-IIoTID dataset is an extensive set of network traffic data, host resources, logs, and alerts that records the behaviors of new IIoT connection protocols, devices that have just been introduced, a variety of attack types and situations, and different attack protocols. Since the dataset is intended to be device- and connectivity-agnostic, it is extremely reflective of the heterogeneous and interoperable real-world IIoT systems. In order to assist in the creation of all-encompassing security solutions for IIoT systems, it seeks to give a realistic and accurate representation of the activities, connectivities, and potential cyber-attack behaviors of these systems. While the dataset aims to reflect the changes and heterogeneity of network traffic and systems' activities, it may not encompass all possible IIoT systems and attacks, particularly given the rapid evolution of technologies and tactics in this domain.

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

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

$$\text{Recall (TPR)} = \frac{TP}{TP+FN}$$

$$\text{TNR} = \frac{TN}{TN+FP}$$

$$\text{F1 score} = \frac{2(precision \times recall)}{precision+recall}$$

$$\text{F2 score} = \frac{5(precision \times recall)}{(4 \times recall)+precision}$$

4.3 Result evaluation

4.3.1 Evaluation in centralized environment

In a centralized environment, we employed four traditional classification methods and three DL methods. The traditional classification methods utilized include logistic regression, NB, KNN, and SVM. The DL approaches are as follows:

1. **MLP:** A Multi-Layer Perceptron neural network [37] is employed for classification, featuring a single hidden layer with 64 units and an output layer with a **'sigmoid'** activation function.
2. **CNN:** A CNN [38] is utilized for classification, consisting of three 1D convolutional layers (filters = 64, stride = 2, and kernel size = 3) each followed by an average pooling layer (pool size = 3 and stride = 2).
3. **Proposed Model:** The ResNet-based model discussed in Sect. 3.2 is used for classification purposes.

In DL methods, we used batch size = 32, and the number of epochs = 32. The batch size and number of epochs are selected based on a greedy search.

The confusion matrices for each approach are presented in Table 4, while other metrics are illustrated in Fig. 6 and Table 3. Logistic regression achieves an accuracy of 91.43%, exhibiting high precision and a reasonable F1 score. Conversely, Naïve Bayes has an accuracy of 84.69%, which is comparatively lower than the other models. KNN attains an accuracy of 98.29%, with high precision, F1 score, and recall, albeit a slightly lower True Negative Rate (TNR). SVM demonstrates strong performance with an accuracy of 97.89% and high precision but lower recall and F1 score compared to KNN. MLP achieves an accuracy of 98.47% with high precision but lower recall than KNN.

The DL-based model, CNN, displays high performance with an accuracy of 99.19%, achieving high precision, F1 score, recall, and TNR. Our proposed model

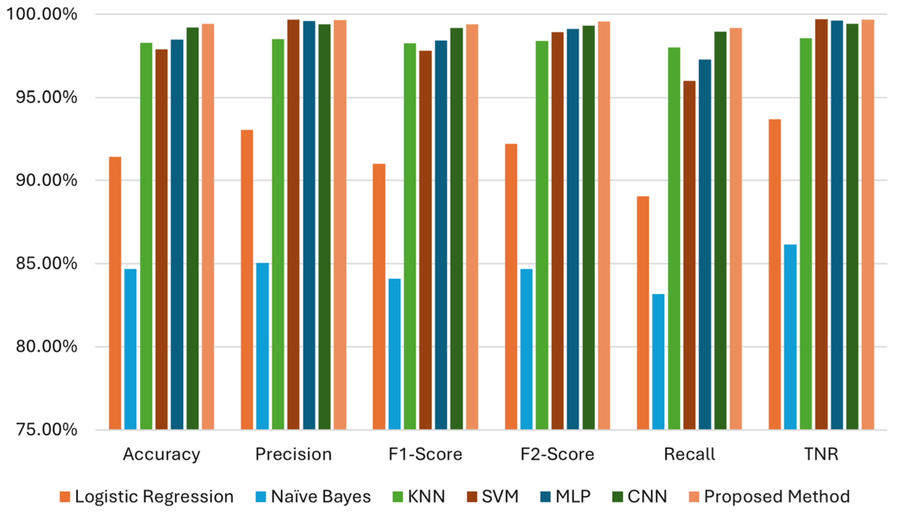

Fig. 6 Comparison of various classification approaches in the centralized environment

Table 3 Comparison of various classification techniques in centralized environment

Model	Accuracy	Precision	F1 score	F2 score	Recall	TNR
Logistic regression	91.43%	93.05%	91.00%	92.22%	89.04%	93.70%
Naïve Bayes	84.69%	85.05%	84.10%	84.67%	83.17%	86.14%
KNN	98.29%	98.49%	98.24%	98.39%	98.00%	98.57%
SVM	97.89%	99.68%	97.79%	98.92%	95.98%	99.70%
MLP	98.47%	99.58%	98.41%	99.11%	97.26%	99.61%
CNN	99.19%	99.40%	99.17%	99.31%	98.94%	99.43%
Proposed method	99.43%	99.65%	99.41%	99.56%	99.18%	99.67%

surpasses all other models with an accuracy of 99.43% and high-performance metrics. Having a high recall indicates a low amount of False Negatives, thus reducing the possibility of missed attacks. To balance precision and recall, F-beta scores are used, allowing us to optimize our models for specific requirements. The F2 score prioritizes recall over precision, doubling its weight. Our proposed approach demonstrates high recall and F2 score, indicating a rare occurrence of missed attacks.

In general, the comparison of these models reveals that DL-based models, such as CNN and our proposed model, outperform traditional ML models, including logistic regression, NB, KNN, and SVM, for accuracy and other performance metrics. Ultimately, the proposed model excels over all other models in terms of accuracy and performance metrics, rendering it a more reliable and efficient option for intrusion detection in industrial IoT networks. A ResNet is easily able to handle more complexity without vanishing gradient issues as compared to DL-based models. This

Table 4 Confusion matrix of various classification methods

Classification methods	Confusion matrix		
Logistic regression	**Actual class**		
	Predicted class	Attack	Normal
	Attack	71,124	5310
	Normal	8756	78,974
Naïve Bayes	**Actual class**		
	Predicted class	Attack	Normal
	Attack	66,436	11,681
	Normal	13,447	72,603
K-Nearest-Neighbors (KNN)		**Actual class**	
	Predicted class	Attack	Normal
	Attack	78,287	1204
	Normal	1596	83,080
SVM	**Actual class**		
	Predicted class	Attack	Normal
	Attack	76,674	249
	Normal	3209	84035
Multi-Layer Perception (MLP)	**Actual class**		
	Predicted class	Attack	Normal
	Attack	77697	327
	Normal	2186	83957
Convolutional Neural Network (CNN)	**Actual class**		
	Predicted class	Attack	Normal
	Attack	79035	477
	Normal	848	83807
Our proposed method	**Actual class**		
	Predicted class	Attack	Normal
	Attack	79226	279
	Normal	657	84005

allows ResNet to capture complex hierarchical features and patterns without much degradation.

4.3.2 Evaluation in federated environment

The hyperparameters, such as batch size and number of epochs, are chosen through a greedy search approach. The dataset is distributed among 10 clients to simulate a federated environment, with each client performing four epochs of local training in each communication round. The FL process encompasses 20 communication rounds. The comparator models are described as:

1. **Logistic regression equivalent model:** As we know, the logistic function is:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \tag{3}$$

where L = the maximum value the function can take (in binary classification value is in the range of 0 to 1, so $L = 1$). x_0 controls where on the x-axis the growth should be, $f(x_0) = \frac{L}{2}$. k decides the change of function from the minimum to the maximum value. We employ a gradient descent algorithm for loss calculation and effective classification in logistic regression-based classification. Thus, we utilized a neural network closely resembling logistic regression for classification. In this model, we employed the Stochastic Gradient Descent (SGD) optimizer function and a single-unit dense layer with a sigmoid function as the output layer, which is directly connected to the input layer. The sigmoid function is a special case of the logistic function with $L = 1$ (for binary classification), $x_0 = 0$, and $k = 1$. Consequently, it can be used for binary classification.

$$Sigmoid(v) = \frac{1}{1 + e^{-v}} \tag{4}$$

2. **Quasi-SVM model:** The Quasi-SVM model is an adaptation of the SVM that has been implemented using the Keras library of TensorFlow, developed by the Google Brain Team [39]. This model involves utilizing a Random Fourier Features layer to nonlinearly transform the input features, which effectively kernelizes the linear model. Subsequently, a linear model is trained on top of the transformed features, and hinge loss is employed to replicate the SVM kernel within the neural network.
3. **Fifty-Layeryed model:** Our third model is an MLP model consisting of 50 dense layers connected in series. The initial 25 hidden layers have 64 nodes each, and the remaining 25 hidden layers have 32 nodes each. The output layer has a single node with a sigmoid activation function.
4. **Proposed model:** The final model used in our experiments is our modified ResNet model as described in Sect. 3.2. The comparison results of these models are shown in Fig. 7 which shows the accuracy of each model in different communication rounds. The loss of each model during the learning process is shown in Fig. 8, and the metrics of each model in the last round of communication are presented in Fig. 6 showcasing supremacy of the ResNet-based model. Additionally, this model does not suffer from the vanishing gradient problem even with 50 layers, unlike the 50-layered MLP model whose accuracy does not increase with communication rounds. Furthermore, the loss of this model remains fixed, and the precision and recall values are zero, leading to an accuracy of 50%, indicating that this model always predicts false values. This behavior confirms that the gradients of the 50-layered MLP model are not being updated due to the vanishing gradient problem. Note that the F1 score value cannot be calculated for the 50-layered MLP model shown in Fig. 9. This is because the model has 50 fully connected

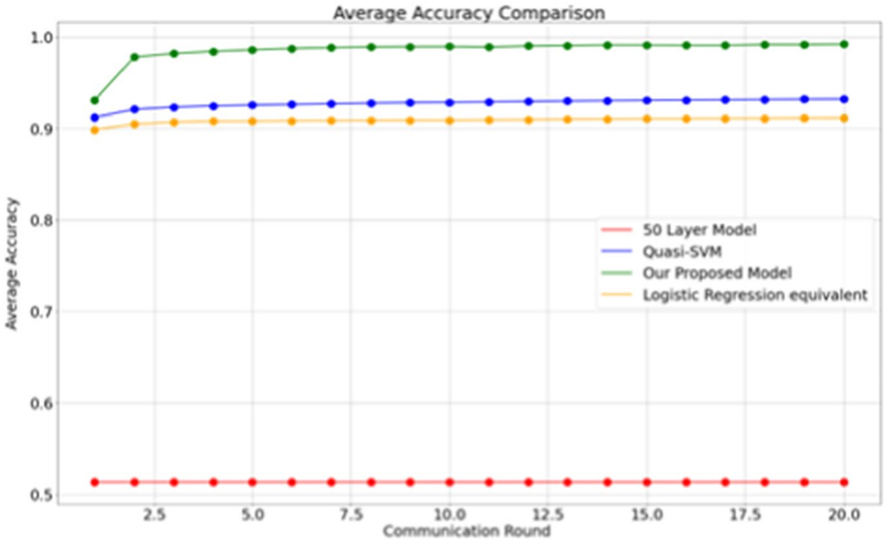

Fig. 7 Accuracy comparison of various models in the federated environment

layers without any skip connections, which causes the vanishing gradient problem and results in the model not learning anything. Therefore, the final output always remains the same as the initial weight of the output neuron, which is zero. As a result, all predictions are false, and the count of True Positives is zero. This leads to both precision and recall being zero. The F1 score is the harmonic mean of precision and recall; as both values are zero, their harmonic mean is undefined, resulting in an undefined F1 score value. Similarly, the F2 score is also rendered undefined for the MLP model.

The Quasi-SVM model employs hinge loss, while binary cross-entropy loss is utilized for the remaining models. In this comparison, our focus is on the loss of these three models. As descripted in Table 5, the proposed model exhibits a significantly lower loss in comparison with other models.

5 Conclusion and future works

In conclusion, this paper introduces a unique federated network intrusion detection approach grounded in ResNet, offering a collaborative learning framework with ensured privacy for each participating entity. By distributing the training process across local clients, this method not only promotes privacy but also shares the training costs among participants. Our experimental results showcase the model's commendable accuracy of 99.16%. Beyond network intrusion detection, the proposed method demonstrates its potential applicability to efficiently classify high-dimensional large datasets in various other domains. Although the proposed approach has

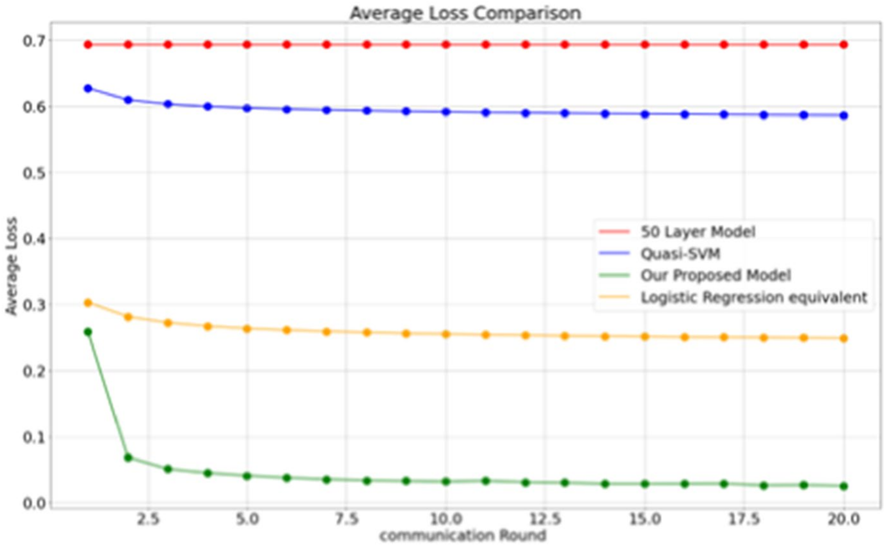

Fig. 8 Loss comparison of various models in the federated environment

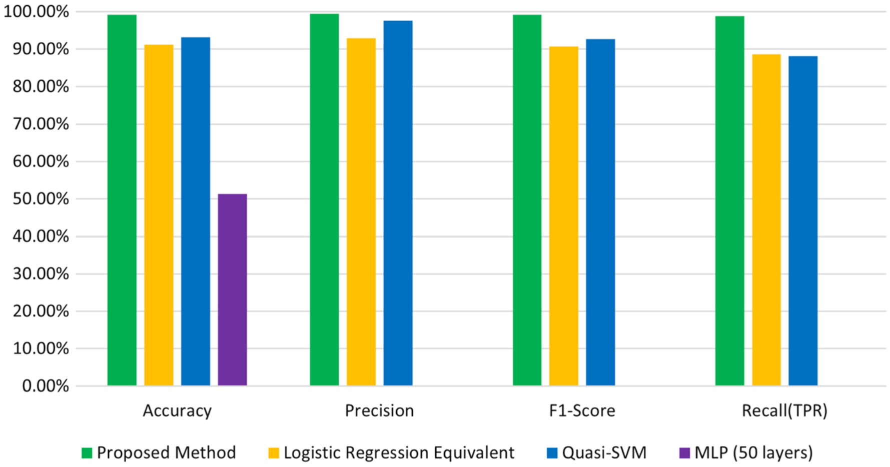

Fig. 9 Comparison of various classification approaches in the federated environment

its benefits, some concerns need to be addressed. One of the main issues is that it may not be secure against federated learning attacks like data poisoning and model poisoning. While it provides better privacy than centralized training, there is still a need to share a small portion of data to initialize the shared federated model, which can raise privacy concerns. In the future, we would like to address the above-mentioned security and privacy concerns and also focus on generalizing by analyzing its complexity and testing its efficacy for other intrusion circumstances.

Table 5 Comparison of various classification techniques in federated environment

Model	Accuracy	Precision	Recall (TPR)	F1 score	F2 score
Logistic regression equivalent	91.13%	92.86%	90.67%	91.75%	92.41%
Quasi-SVM	93.18%	97.62%	92.64%	95.06%	96.48%
MLP (50 layers)	51.34%	0.00%	0.00%	?	?
Proposed method	99.16%	99.48%	99.14%	99.30%	99.41%

'?' = not defined

Author Contributions NC was involved in concept, design, and writing—original draft. MR was involved in concept, design, implementation, and writing. PV contributed to concept, review, and writing. NM participated in concept and implementation. NB helped with concept, design, and review.

Funding Open Access funding provided by the IReL Consortium.

Data availability Not applicable.

Declaration

Conflict of interest There is no conflict of interest among the authors of the manuscript.

References

1. Buczak AL, Guven E (2016) A survey of data mining and ML methods for cyber security intrusion detection. IEEE Commun Surv Tutor 18(2):1153–1176
2. Samrin R, Vasumathi D (2017) Review on anomaly based network intrusion detection system. In: 2017 International Conference on Electrical Electronics Communication Computer and Optimization Techniques (ICEECCOT). IEEE, pp 141–147
3. Naseer S, Saleem Y, Khalid S, Bashir MK, Han J, Iqbal MM, Han K (2018) Enhanced network anomaly detection based on deep neural networks. IEEE Access 6:48231–48246
4. Yin C, Zhu Y, Fei J, He X (2017) A DL approach for intrusion detection using recurrent neural networks. IEEE Access 5:21954–21961
5. Mcmahan HB, Moore E, Ramage D, Hampson S, Arcas B (2017) Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS). JMLR: W &CP; 54
6. Verma P, Breslin JG, O'Shea D (2022) FLDID: FL Enabled deep intrusion detection in smart manufacturing industries. Sensors 22:8974. https://doi.org/10.3390/s22228974
7. Wang K, Li J (2022) An intrusion detection method integrating KNN and transfer extreme learning machine. In: 2022 2nd Asia-Pacific Conference on Communications Technology and Computer

Science (ACCTCS), Shenyang, China, pp 221–226. https://doi.org/10.1109/ACCTCS53867.2022.00053

8. Wazirali R (2020) An improved intrusion detection system based on KNN hyperparameter tuning and cross-validation. Arab J Sci Eng 45:10859–10873. https://doi.org/10.1007/s13369-020-04907-7
9. Farnaaz N, Jabbar M (2016) RF modeling for network intrusion detection system. Proc Comput Sci 89:213–217
10. Waskle S, Parashar L, Singh U (2020) Intrusion detection system using PCA with RF approach. In: 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC). Coimbatore, India 2020:803–808. https://doi.org/10.1109/ICESC48915.2020.9155656
11. Mebawondu OJ, Popoola OS, Ayogu II, Ugwu CC, Adetunmbi AO (2022) Network intrusion detection models based on NB and C4.5 algorithms. In: 2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON), Lagos, Nigeria, pp 1–5. https://doi.org/10.1109/NIGERCON54645.2022.9803086
12. Sharmila BS, Nagapadma R (2019) Intrusion detection system using NB algorithm. In: 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Bangalore, India, pp 1–4. https://doi.org/10.1109/WIECON-ECE48653.2019.9019921
13. Zhang R, Song Y, Wang X (2022), Network intrusion detection scheme based on IPSO-SVM Algorithm. In: 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), Dalian, China, pp 1011–1014. https://doi.org/10.1109/IPEC54454.2022.9777568
14. Reddy RR, Ramadevi Y, Sunitha K.N (2016) Effective discriminant function for intrusion detection using SVM. In: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, pp 1148–1153
15. Pang J, Liu D, Peng Y, Peng X (2017) Anomaly detection based on uncertainty fusion for univariate monitoring series. Measurement 95:280–292
16. Santoro D, Escudero-Andreu C, Kyriakopoulos KCG, Aparicio-Navarro FJ, Parish DJ, Vadursi M (2017) A hybrid intrusion detection system for virtual jamming attacks on wireless networks. Measurement 109:79–87
17. Akashdeep S, Manzoor I, Kumar N (2017) A feature reduced intrusion detection system using ANN classifier. Expert Syst Appl 88:249–257
18. Park SH, Park HJ, Choi YJ (2020) RNN-based prediction for network intrusion detection. In: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, pp 572–574. https://doi.org/10.1109/ICAIIC48513.2020.9065249
19. Torres P, Catania C, Garcia S, Garino CG (2016) An analysis of recurrent neural networks for botnet detection behavior. In: IEEE Biennial Congress of Argentina (ARGENCON). IEEE, Buenos Aires, Argentina 2016:1–6
20. Wang W, Zhu M, Zeng X, Ye X, Sheng Y (2017) Malware traffic classification using CNN for representation learning. In: 2017 International Conference on Information Networking (ICOIN). IEEE, Da, Nang. Vietnam, pp 212–717
21. Wang W, Sheng Y, Wang J, Zeng X, Ye X, Huang Y, Zhu M (2018) HAST-IDS learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access 6:1792–1806
22. Al-Qatf M, Lasheng Y, Al-Habib M, Al-Sabahi K (2018) DL approach combining sparse autoencoder with SVM for network intrusion detection. IEEE Access 6(52843):52856
23. Bontemps L, McDermott J, Le-Khac NA et al. (2016) Collective anomaly detection based on long short-term memory recurrent neural networks. In: International Conference on Future Data and Security Engineering. Springer, pp 141–152
24. Zhao R, Yin Y, Shi Y, Xue Z (2020) Intelligent intrusion detection based on FL aided long short-term memory. Phys Commun 42:101157
25. Verma P, Bharot N, Breslin JG, O'Shea D, Vidyarthi A, Gupta D (2023) Zero-day guardian: A dual model enabled FL framework for handling zero-day attacks in 5G enabled IIoT. IEEE Trans Consum Electron. https://doi.org/10.1109/TCE.2023.3335385
26. ElKashlan M, Elsayed MS, Jurcut AD, Azer M (2023) A machine learning-based intrusion detection system for IoT electric vehicle charging stations (EVCSs). Electronics 12:1044. https://doi.org/10.3390/electronics12041044
27. Awajan A (2023) A novel deep learning-based intrusion detection system for IoT networks. Computers 12:34. https://doi.org/10.3390/computers12020034

28. Idrissi MJ, Alami H, El Mahdaouy A, El Mekki A, Oualil S, Yartaoui Z, Berrada I (2023) Fed-ANIDS: federated learning for anomaly-based network intrusion detection systems. Expert Syst Appl 234(121000):0957–4174. https://doi.org/10.1016/j.eswa.2023.121000
29. He K, Zhang X, Ren, Sun S, J (2016) Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), PIEAS, Islamabad, Pakistan, 26–27 August pp 770–778
30. Szegedy C, Vanhoucke V, Ioffe, S, Shlens J Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June, pp 2818–2826
31. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein MM et al (2014) Imagenet large scale visual recognition challenge. Int J Comput Vis 115:1–37
32. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. Comput Sci 9:1–14
33. He K, Zhang X, Ren S, Jian S (2015) Delving deep into rectifiers: Surpassing human-level performance on imageNet classification. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December pp 1–11
34. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition, arXiv [cs.CV]
35. McMahan B, Moore E, Ramage D, Hampson S, Arcas BAY (2017) Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp 1273–1282
36. Al-Hawawreh M, Sitnikova E, Aboutorab N (2022) X-IIoTID: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things. IEEE Internet of Things J 9(5), pp 3962–3977, 1 March. https://doi.org/10.1109/JIOT.2021.3102056
37. Bisong E (2019). The multilayer perceptron (MLP). https://doi.org/10.1007/978-1-4842-4470-8_31
38. O'Shea K and Nash R (2015) An introduction to CNNs. ArXiv e-prints
39. Keras Quasi SVM (2021) [Online]. Available: https://keras.io/examples/keras_recipes/quasi_svm/. [Accessed: Mar. 26, 2023]

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Nisha Chaurasia[1] · Munna Ram[1] · Priyanka Verma[3] · Nakul Mehta[1] · Nitesh Bharot[2]

✉ Priyanka Verma
priyanka.verma@ul.ie

Nisha Chaurasia
chaurasian@nitj.ac.in

Munna Ram
munnar.it.20@nitj.ac.in

Nakul Mehta
nakulm.it.20@nitj.ac.in

Nitesh Bharot
nitesh.bharot@universityofgalway.ie

[1] Dr B R Ambedkar NIT Jalandhar, Jalandhar 144008, Punjab, India

[2] Data Science Institute, University of Galway, University Road, Galway H91TK33, Ireland

[3] Department of Electronics and Computer Engineering, University of Limerick,
Castletroy, Limerick V94 T9PX, Ireland