Federated-Learning-Based Anomaly Detection for IoT Security Attacks

Viraaji Mothukuri, *Graduate Student Member, IEEE*, Prachi Khare, Reza M. Parizi, *Senior Member, IEEE*, Seyedamin Pouriyeh, *Associate Member, IEEE*, Ali Dehghantanha, *Senior Member, IEEE*, and Gautam Srivastava, *Senior Member, IEEE*

***Abstract*—The Internet of Things (IoT) is made up of billions of physical devices connected to the Internet via networks that perform tasks independently with less human intervention. Such brilliant automation of mundane tasks requires a considerable amount of user data in digital format, which, in turn, makes IoT networks an open source of personally identifiable information data for malicious attackers to steal, manipulate, and perform nefarious activities. A huge interest has been developed over the past years in applying machine learning (ML)-assisted approaches in the IoT security space. However, the assumption in many current works is that big training data are widely available and transferable to the main server because data are born at the edge and are generated continuously by IoT devices. This is to say that classic ML works on the legacy set of entire data located on a central server, which makes it the least preferred option for domains with privacy concerns on user data. To address this issue, we propose the federated-learning (FL)-based anomaly detection approach to proactively recognize intrusion in IoT networks using decentralized on-device data. Our approach uses federated training rounds on gated recurrent units (GRUs) models and keeps the data intact on local IoT devices by sharing only the learned weights with the central server of FL. Also, the approach's ensembler part aggregates the updates from multiple sources to optimize the global ML model's accuracy. Our experimental results demonstrate that our approach outperforms the classic/centralized machine learning (non-FL) versions in securing the privacy of user data and provides an optimal accuracy rate in attack detection.**

***Index Terms*—Federated learning (FL), gated recurrent units (GRUs), Internet of Things (IoT), recurrent neural networks (RNNs), security.**

I. Introduction

The Internet of Things (IoT) is made up of digitally interconnected devices and networks with an ability to perform tasks with a high degree of automation. In the current digital era, industries are rapidly moving toward artificial intelligence (AI)-driven smart solutions to capitalize on users' data. IoT's microservice architecture makes it a preferred choice for ML solutions for several industrial applications. IoT can accommodate AI-enabled services by integrating ML solutions in mini-compatible hardware architectures. Currently, IoT devices are providing improvised smart solutions and enhancing services in various domains such as healthcare [1], intelligent digital assistants [2], smart home [2], [3], and in the industrial domain, also known as the Industrial IoT (IIoT), to name a few.

IoT devices are proven to excel in delivering AI solutions but on the downside, they rely on sensitive user data to perform tasks. The requirement of IoT devices to function with optimal energy consumption makes its microarchitecture style less suitable to deploy computationally heavy security firewalls, making IoT devices more vulnerable to various attacks. As discussed in [4], Mirai and other variations of malware bots can exploit the vulnerabilities in IoT devices and take control over AI functionality of it, and in turn, accessing other non-IoT devices connected to it. This emphasizes the fact that unguarded IoT devices could turn into an open threat to all other network devices interconnected with them. Network protocols in IoT networks are a critical interface that connects physical devices with the digital world. The research work in [5] and [6] explores the vulnerabilities in IoT, and feature-based security risks are explored, and Panchal *et al.* [7] discussed various attacks and their impact on IIoT networks. This emphasizes the fact that Mirai is just one of the attacks and that there has been exponential growth in malicious activities, which are successful in exploiting the vulnerabilities of IoT networks [8]. The idea of microdevices delivering intelligent digital assistance has been tremendously appreciated and proven to reduce the manual work, which, in turn, created a high demand for the production of enormous variants of IoT devices. The eagerness to meet the demands has resulted in the production of different IoT devices with poor architecture choices leading to both heterogeneous environments as well as highly vulnerable IoT devices providing and sharing digital information.

The heterogeneity of IoT devices and the necessity for frequent training to maintain optimal performance makes it a tough task to configure an ML-based anomaly detection method [9], [10]. This, in turn, makes security issues in IoT

Manuscript received December 15, 2020; revised February 23, 2021 and April 14, 2021; accepted May 3, 2021. Date of publication May 5, 2021; date of current version February 4, 2022. *(Corresponding author: Gautam Srivastava.)*

Viraaji Mothukuri, Prachi Khare, Reza M. Parizi, and Seyedamin Pouriyeh are with the College of Computing and Software Engineering, Kennesaw State University, Kennesaw, GA 30004 USA (e-mail: vmothuku@students.kennesaw.edu; pkhare@students.kennesaw.edu; rparizi1@kennesaw.edu; spouriye@kennesaw.edu).

Ali Dehghantanha is with the Cyber Science Lab, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: adehghan@uoguelph.ca).

Gautam Srivastava is with the Department of Mathematics and Computer Science, Brandon University, Brandon, MB R7A 6A9, Canada, and also with the Research Centre for Interneural Computing, China Medical University, Taichung 404, Taiwan (e-mail: srivastavag@brandonu.ca).

Digital Object Identifier 10.1109/JIOT.2021.3077803

a major issue for both developers and end users. Over the years, many research proposals have been made to address the open vulnerabilities of IoT devices [11], [12]. Among those, ML-based solutions [13], [14] for detecting network intrusions have surged. However, ML-based solutions are the least preferred option due to the disadvantages such as:

1) the prerequisite to having the whole set of training data on a central server;
2) security risks involved in transferring raw data from end devices to a central server;
3) training huge volumes of data on a single server can be computationally expensive.

One of the promising and well-adaptable approaches that can rectify these disadvantages in the ML-based approach is federated learning (FL) [15], [16]. In FL, decentralized ML model training keeps the data intact on the edge device and only the learned ML model weights are transferred to a central server. This strategy of FL is proven to secure the privacy of user data [17], making it the preferred approach in comparison to non-FL solutions.

In this article, we propose a decentralized FL approach with an ensembler to enable anomaly detection on the IoT networks. The approach enables on-device training and helps to train the anomaly detection ML model on IoT networks without the need to transfer network data to a centralized server. To ensure optimal results in predicting intrusion in IoT networks, we use long short-term memory (LSTM) and gated recurrent units (GRUs) [which are the improved versions of basic recurrent neural networks (RNNs)] neural network models to efficiently train the ML model on the Modbus network data set [18]. Our experimental results demonstrate a minimized error rate in predicting attacks and a reduced number of false alarms in comparison to the classic (centralized) ML approach. Our contributions in this work can be summarized as follows.

1) On-device ML training with FL to secure data privacy at end devices is enabled.
2) Higher accuracy rates and minimized false alarms in attack detection compared to a centralized ML (non-FL) approach are achieved.
3) The benefits of integrating FL with ensembler to achieve optimal result are demonstrated.

The remainder of this article is structured in the following manner. Section II gives the related work. Section III presents the proposed approach and illustrates the underlying architecture with implementation details. Section IV presents the data set, metrics, and evaluation results, and summarizes our findings. Finally, Section V concludes the article.

II. Related Work

IoT is proven to be successful in delivering ML solutions in its microarchitecture design [11], [19]. The growing popularity and usage of IoT have created many interesting research areas. One such research path is the detection and classification of attacks in IoT networks, and there are several research works proposed to secure IoT networks from malicious attacks [20]. This section will cover the recent studies that are proposed to improve the security of IoT networks using ML techniques.

Nguyen *et al.* [21] proposed an autonomous self-learning system named DÏoT, which is an FL-based approach for detecting IoT devices infected with Mirai malware in IoT smart home networks. The FL part is implemented using Python's *flask* and *flask socketio*. TensorFlow[1] deep learning framework is used to implement FL's global model. The architecture of DÏoT consists of a security gateway and IoT security services. A security gateway is configured as an access point between IoT devices and the Internet. The anomaly detection component is integrated with a security gateway, which monitors the network for abnormal activity. IoT security services maintain a repository of device-specific anomaly detection models and aggregate the model weights updates from IoT devices. When new devices are included in the IoT network, the device-specific anomaly detection retrieves existing anomaly detection models from the repository and enables monitoring of network traffic. Due to self-learning algorithm labelling, the attack is not mandatory as DÏoT learns the pattern in the attack category. As per evaluation results, false alarms are minimized in detecting attacks. However, the approach was limited to single (Mirai) attack types and lacks the implementation of an FL-specific deep learning framework.

Li *et al.* [22] proposed an FL-based intrusion detection framework called Deepfed, for identifying threats in cyber–physical systems (CPSs). A combination of convolutional neural networks (CNNs) and GRUs is leveraged for threat identification, and a security protocol based on the Paillier cryptosystem is used to ensure the security of local and global models during the FL training process. The research work in [23] proposes FedAGRU, an FL-based attention gated recurrent unit. FedAGRU is an improvised federated averaging algorithm that is designed to identify poisoning attacks and eliminate minimal contributing updates for a highly efficient global model with optimal communication costs. The evaluation results on three data sets [24]–[26] show promising results supporting the proposed approach. Similarly, Cetin *et al.* [27] proposed an FL-based approach for wireless intrusion detection (WID) with the awid data set.[2] Al-Athba Al-Marri *et al.* [28] leveraged a mimic learning strategy to implement FL and combine it with the ML-based intrusion detection system (IDS). Another FL-based approach presented by [29] is an FL-based IDS using the TensorFlow federated (TFF[3]) framework.

Among the proposed approaches, ML-based intrusion detection presented in [13] is similar to our work, where a centralized version of anomaly detection is the proposed TensorFlow-based deep learning framework. Six LSTMs [30] of different layers are used as a threat detection algorithm. The evaluation results confirm the efficiency of the model, but it is limited to a centralized version of ML, and our approach enhances it much further by implementing FL and computationally inexpensive GRUs with PySyft [31] deep learning frameworks. Another ML-based anomaly detection

[1] https://www.tensorflow.org

[2] https://icsdweb.aegean.gr/awid/

[3] https://www.tensorflow.org/federated

TABLE I
ACRONYMS

Acronym	Description
GRUs	Gated Recurrent Units
LSTMs	Long Short Term Memory Networks
RNNs	Recurrent Neural Networks
FL	Federated Learning
ML	Machine Learning
CSV	Comma Separated Values
Modbus RTU	Remote Terminal Unit
$sigma/\sigma$	Sigmoid
$tanh$	Tangent hyperbolic
rfc	Random forest classifier
PCAP	Packet capture

is proposed in [14] research work, where distinct attributes in the data set are used to identify anomalies in the smart home IoT devices. The authors proposed the basic artificial neural networks (ANNs) classification algorithm and the logistic regression algorithm for identifying attacks. The focus of the paper is limited to identifying the attack patterns in the data and uses a basic classification algorithm that may not be adaptable to an evolving range of IoT devices. Liu *et al.* [32] used the attention-based convolutional neural network LSTM for identifying anomalies in time-series data of IIoT devices. PySyft and PyTorch[4] deep learning frameworks are used to implement FL, and a gradient compression technique is proposed to improve communication efficiency.

To summarize, the existing research work lags in implementing a decentralized communication efficient framework for anomaly detection in IoT networks. In our approach, we considered those limitations and proposed an FL-based approach for IoT security attacks.

III. PROPOSED APPROACH

In this section, we discuss and illustrate the details of the architecture of ML models and the proposed approach. The list of acronyms used is given in Table I.

A. LSTMs and GRUs

In this part, we give an overview of the deep learning ML models we have used in our proposed approach. LSTM networks [30], [33], [34] and GRUs [35] are a variation of RNNs, which are proposed to address the short-term memory/vanishing gradients problem in a basic variant of RNNs. The architecture of LSTMs and GRUs consists of gates to monitor the information flow and control the learning process, which enables the network to learn from long-term dependencies. Gates act as switches in the network, which helps in retaining long- and short-term information. Anomaly detection [36], speech recognition, speech synthesis, and text generation [37]–[40] are few real-time implementations of LSTMs and GRUs. During the evaluation of our proposed approach, we have experimented with both GRUs and LSTMs, initial FL training rounds in which GRUs models shown in Table II outperformed LSTMs in achieving a higher accuracy rate, and being computationally inexpensive [41].

TABLE II
GRUS

GRUs Model Name	Number of Layers	Dropout	Hidden layer size
GRU-1	2	0.01	256
GRU-2	1	0.00	100
GRU-3	1	0.00	200
GRU-4	2	0.00	100

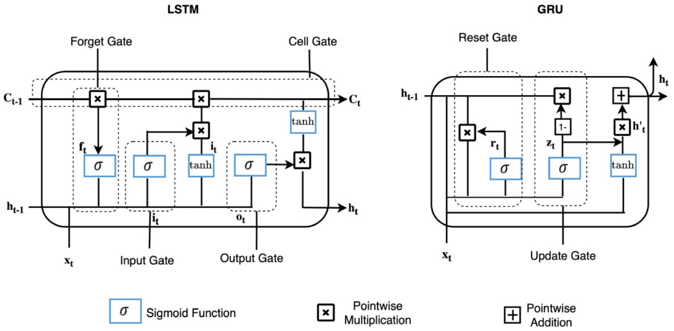

Fig. 1. Illustration of LSTM and GRUs.

Below are the details of the components of GRUs and LSTMs illustrated in Fig. 1 adopted from a blog.[5]

1) *Sigmoid Function:* σ provides a way to decide whether any information needs to be retained or discarded. σ generates values ranging between 0 and 1, where a value near to 0 will enable information in the network to be forgotten and 1 indicates information that needs to be kept for future updates.
2) *Tangent Hyperbolic (tanh):* An activation function generates values between -1 and 1. This ensures that negative values are mapped strongly negative and positive values are mapped between 0 and 1. For a neural network, the hidden layers tanh function is preferred as its mean value makes the learning much easier for the next layers.
3) *Cell State:* It represents the information retained throughout the memory block of LSTM. C_t represents the current cell state or memory cell and C_{t-1} represent the previous cell state.
4) *Gates of LSTMs:* Based on the human's ability to memorize rhyming patterns, LSTMs are proposed as

[4] https://pytorch.org

[5] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

a memory block where interconnected memory cells collect and retain information for long-term reference. Gates are used for controlling information retention, retrieval, and deletion through memory cells. LSTMs consist of an input gate, forget gate, and output gate. Below are the details of each gate.

a) *Forget Gate:* The information that does not contribute toward the learning of the LSTM network is discarded for the given cell as shown in

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) \tag{1}$$

where f_t is the current value of forget gate, which is the result of the sigmoid function, x_t is the current input for the memory cell, W_f is the weight matrix from the forget gate to the input, b is the forget gate bias, and h_{t-1} is the information from the previous cell, respectively.

b) *Input Gate:* It helps to determine whether the current information is useful enough to retain it in the cell state for future reference. Equations (2)–(4) represent the calculation of the input gate where the current cell state value is determined by the sum of forget gate f_t and previous cell state c_{t-1} product and input gate and current state candidate value $\hat{c}_t$, respectively

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \tag{2}$$

$$\hat{c}_t = \tanh\left(W_c\left[h_{t-1}, x_t\right] + b_c\right) \tag{3}$$

$$c_t = f_t * c_{t-1} + i_t * \hat{c}_t \tag{4}$$

where i_t results from the sigmoid layer representing the input gate, $\hat{c}_t$ is a cell activation function, which is created by the tanh layer, c_{t-1} is the cell state of the previous timestamp memory cell, and c_t calculates the current cell value, which is the information that is predicted as important to save for future reference, respectively.

c) *Output Gate:* This gate decides the final output of the network. In (5) and (6), h_t is calculated by running c_t [derived from (4)] through the tanh activation function

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{5}$$

$$h_t = o_t * \tanh(c_t) \tag{6}$$

where o_t is output gate value, which is set to a positive value below one using the sigmoid function, and h_t is the final output value of the current memory cell, respectively.

5) *Gates of GRUs:* GRUs contain much simpler architecture, in comparison to LSTMs. The input to each memory cell is combined as a single value instead of two and works well with just two gates: a) the reset gate and b) update gate. GRUs are computationally inexpensive and take less time to train.

a) *Reset Gate:* Similar to the forget gate of LSTMs, the information is discarded if it is not useful for future learning/reference using

$$r_t = \sigma\left(W_r\left[h_{t-1}, x_t\right]\right) \tag{7}$$

where r_t is the result of the sigmoid layer for the current timestamp/current memory cell of the reset gate, h_{t-1} is the information from the previous memory cell, and x_t is the input for the current memory cell, respectively.

b) *Update Gate:* Instead of the output and input gates, GRUs use a single gate called the update gate, which determines if the information from the current state needs to be retained for future reference

$$z_t = \sigma\left(W_z\left[h_{t-1}, x_t\right]\right) \tag{8}$$

$$\tilde{h}_t = \tanh\left(W\left[r_t * h_{t-1}, x_t\right]\right) \tag{9}$$

$$h_t = \left(1 - z_t * h_{t-1} + z_t * \tilde{h}_t\right) \tag{10}$$

where i_t is the results of the sigmoid layer, $\hat{h}_t$ is the vector that is created by the tanh layer, and h_{t-1} is the previous cell state value.

We are using seven different window sizes, and the input size for each LSTM/GRU is varied with the selected window size. The selection of window size is crucial as the amount of data differs for each window size, which contributes toward the better performance of the ML model. Increased window size length impacts the training time as the information retained increases in each memory cell of the neural network. Similar to the hyperparameter of the ML model, there is no ground rule, which confirms the relation between the window size and performance of the model. However, there are a few research works [34], which suggest that the impact of window size, layers of GRU/LSTM are dependent on the type and size of the data set.

B. Architecture

We propose a federated learning-based approach for AI-enabled anomaly detection on IoT networks. Fig. 2 illustrates the high-level architecture of the approach, which consists of virtual IoT instances representing IoT devices in a network, a local deep learning model (discussed in the next section), and a local copy of training data of each virtual instance, FL averaging component at a central server, global deep learning model for each window size, and an Ensembler component that consists of a random forest decision tree Ensembler. We discuss the details of each step performed to implement strategies of the proposed approach in the following. In the practical scenario, steps for creating virtual instances and preprocess of captured data at a central server will not be necessary as real data for training are available at end devices.

1) *Virtual Instances:* We replicate the IoT network set up by creating virtual instances using PySyft[6,7] [31]. For the chosen n number of end devices, we create virtual instances (denoted as fl_n), and to simulate the central server, we create a dedicated instance called fl_{average} to enable sharing of the trained ML model parameters between the IoT mobile end devices and central FL

[6] PySyft: A deep learning framework that facilitates decentralized ML training by keeping user data intact on end devices.

[7] https://github.com/OpenMined/PySyft

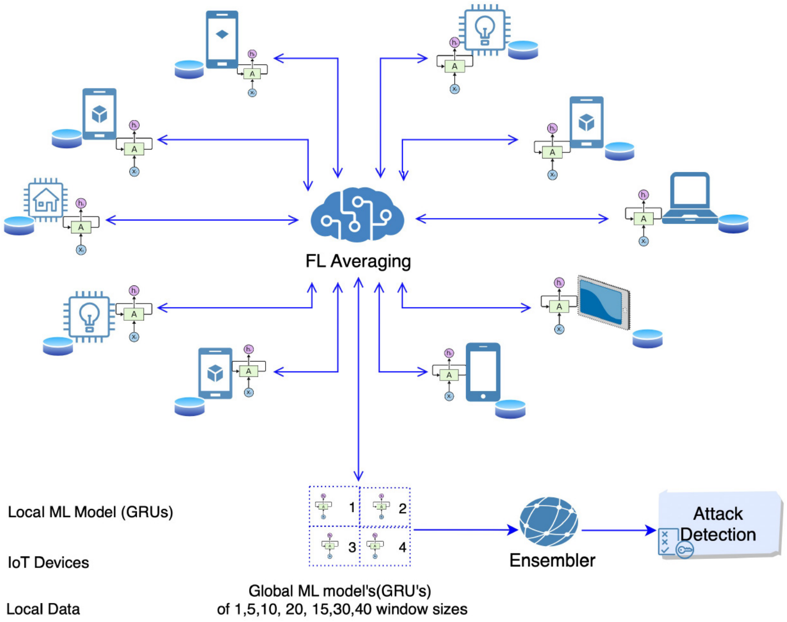

Fig. 2. High-level architecture of the proposed approach.

server. The data set is divided into n chunk and each is distributed to fl_n virtual instances.

2) *Preprocess of Captured Data:* As the part of data preprocessing at each IoT device/IoT gateway (which is a connecting component between a set of IoT devices and digital platform/cloud), network data are captured in *pcap* files and the *pcap* files are converted to CSV files using a CICFlowmeter tool [42]. The converted CSV file is processed further to clean the features that do not add value to the learning process. Finally, the processed data are divided into n chunks and distributed among the virtual instances (fl_n) of IoT end devices.

3) *FL Training:* FL training is executed asynchronously with the available IoT instances. Each client node executes training rounds with their copy of the data set and shares trained local ML model's weights with fl_{average} aggregating instance. In our work, we have used 4-GRUs with layer size and hidden size as listed in Table II. In FL, training rounds are defined as the individual epochs executed on each end device. Algorithm 1 lists formal steps of our proposed approach, and below is a brief description of the FL training logic and the steps driving the approach.
   a) Define window size W_i.
   b) Define virtual instances representing IoT end devices fl_i.
   c) Define GRU network ML model parameters GRU_{ML} for each window size W_i.
   d) Share GRU_{ML} with each virtual instance fl_i.
   e) At each instance fl_i, training rounds are executed on GRU_{ML} and the trained local ML model updates are shared with fl_{average}. To replicate the practical scenario, we have implemented training rounds on a multiprocessor. Each virtual instance fl_i executes training rounds on a separate processor and periodically shares learned local model weights m_{wi} to fl_{average}.
   f) fl_{average} virtual instance acts as an aggregating component on the central server and listens for incoming local model updates m_{wi}. The global ML model M_{wi} of each window size is obtained by combining local ML model weights.
   g) Send a copy of global ML models to each end device.

4) *Ensembler:* Ensemble Learning [43] provides an efficient way to combine ML models' outputs to achieve a higher accuracy rate. This is often attributed to the proven ideology that combining multiple ML models achieves greater/optimal results in comparison to the performance of a single ML model. We use the random forest decision tree classifier (*rfc*[8])[44] to ensemble seven global ML models M_{wi}. For input network data, for example, X with n columns (say X) $X = X_1, \ldots, X_n$, each M_{wi} predicts the probability values $h_1, h_2, \ldots, h_n$ of each label Y for the given input X. The Ensembler combines the probability values of M_{wi} to form a ensemble prediction function $f(x)$, which takes predictions as voting for the labels from each model. Equation 11 represents probability prediction calculation for the given input data X

$$h_i = \tilde{y}_i(M_{wi}(X)) \tag{11}$$

$$f(x) = \arg\max_{y\epsilon Y} \sum_{j=1}^{J} I\big(y = h_j(x)\big). \tag{12}$$

In (12), the prediction function $f(x)$ of *rfc* gets input from prediction probability values of seven ML models M_{wi} for each label $y = y_{\text{Clean}}, y_{\text{MITM}}, y_{\text{pingDDos}},$

[8] Random forest decision tree classifier.

Algorithm 1: Anomaly Detection With Federated Learning

```
Input: ML Local models of GRUs
Output: Network flow anomaly detection
$W = w_1, w_5, w_{10}, w_{15}, w_{20}, w_{30}, w_{40}$    /* window sizes */
$FL = fl_1, fl_2 ... fl_n$    /* Virtual IoT devices */
$m_{w_i} = m_{w1}, m_{w5}, m_{w10}, m_{w15}, m_{w20}, m_{w30}, m_{w40}$
    /* local model weights for each window size */
$M_{w_i} = M_{w1}, M_{w5}, M_{w10}, M_{w15}, M_{w20}, M_{w30}, M_{w40}$
    /* Global ML model weights for each window size */
Function $FL_{Training}$ ($maxtrainingrounds$):
    while $fl_i$ in $FL$ do
        foreach $w_i$ in $W$ do
            $m_{w_i}$ = train($fl_i$ in data($w_i$))    /* Train LSTM with local data */
    return $m_{w_i}$
    EndFunction
Function $fl_{average}$ ($m_{w_i}$):
    foreach $w_i$ in $W$ do
        $M_{w_i} = fl_{average}(m_{w_i})$
    return $M_{w_i}$
    EndFunction
Function $Ensembler$ ($M_{w_i}$):
    $networkdata$    /* New flows in-network data */
    foreach $w_i$ in $W_i$ do
        $lstmpredictions = m_{w_i}(newnetworkdata)$
        $anomalydetectionflag = Ensembler(lstmPredictions)$
    EndFunction
$m_{w_i} = FL_{Training}(maxtrainingrounds)$
$M_{w_i} = fl_{average}(m_{w_i})$
while $fl_i$ in $FL$ do
    foreach $w_i$ in $W$ do
        $m_{w_i} = M_{w_i}$    /* replace local ML */
$Attack_{Prediction} = Ensembler(M_{w_i})$
```

$y_{\text{modbusqueryflood}}, \ldots, y_{\text{synDDos}}$, which represents attack category in the data set (presented in Section IV-A). *rfc* considers each probability as a voting from each ML model and predicts the label with high confidence as output. Fig. 3 illustrates the integration of ensembler with our proposed approach.

IV. Evaluation Results

To evaluate the performance of our proposed approach, we compare it with a non-FL (classic ML) version (using the same deep learning algorithms) based on the data set and evaluation metrics presented in this section. Our environment set up is configured on the lambda GPU (graphics processing unit) server hosted `Ubuntu 18.0.0 LTS` operating system. The deep learning framework we have used is PySyft [31] for FL features, and GRUs as our ML neural network. For the non-FL/classic ML approach, we have used the Pytorch deep learning framework. While FL implements Algorithm 1, non-FL-based GRUs setup trains on classic environment set up of centralized training data.

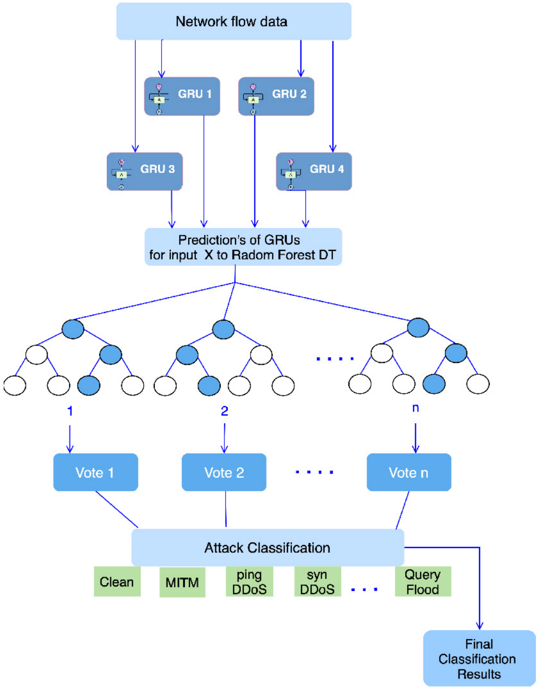

Fig. 3. Illustration of Ensembler in the proposed approach.

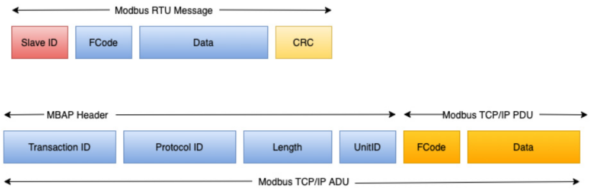

Fig. 4. Modbus message format.

A. Data Set

For the evaluation of our approach, we have used a Modbus-based network data set [18]. Modbus is a decades-old protocol, which provides an efficient way to communicate with physical devices that lack an inbuilt communication protocol. Modbus is used to establish request–response-based communication between devices and is a well-known protocol for many legacy industrial applications. The automation strategies in the industrial domain aim to resolve the interoperability issue using a combination of IoT devices and Modbus protocol. Fig. 4 illustrates the message format in Modbus RTU and Modbus TCP/IP protocols. We have used CICFlowmeter[9] [42] to extract the ML readable CSV from captured network traffic data.

[9] https://github.com/ahlashkari/CICFlowMeter

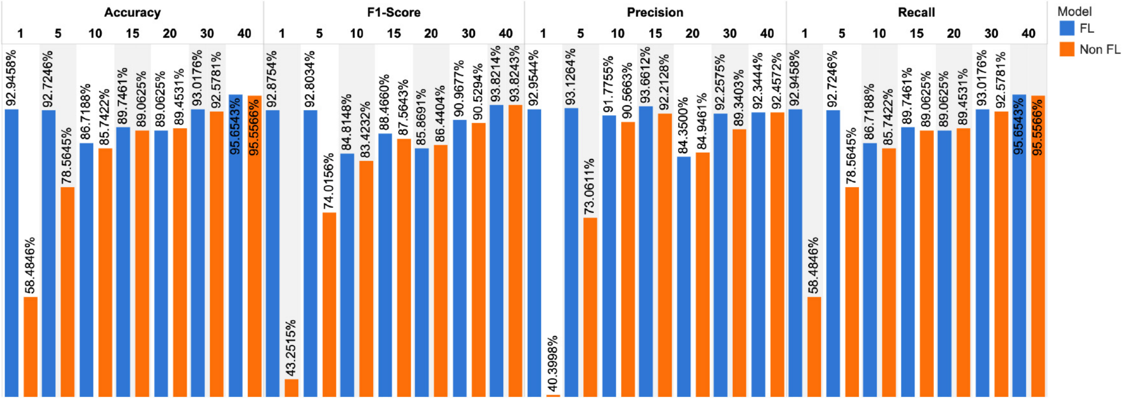

Fig. 5. GRU-1: Model evaluation results of the proposed FL approach and non-FL approach.

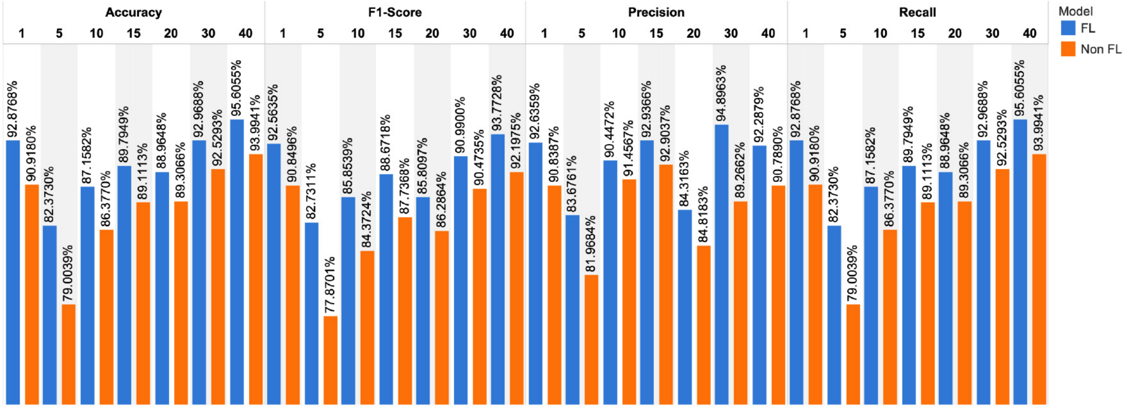

Fig. 6. GRU-2: Model evaluation results of the proposed FL approach and non-FL approach.

Several research works use the Modbus protocol for IoT, especially for IIoT. Jaloudi [45] suggested that the combination of Modbus TCP with IoT-specific message queuing telemetry transport (MQTT) brings in interoperability to industrial devices, such as Internet-based monitoring and industrial control systems. IoT gateway is an interface connecting application layer information to the back-end server, and the research work in [46] proposes an approach to adapt the Modbus protocol for different IoT sensor devices. However, the Modbus protocol is vulnerable to many attacks [47], of which the data set we chose contains the below-summarized attacks.

1) *Man-in-the-Middle Attack:* As the name suggests, during communication between two parties, the third-party entity, the attacker, impersonates as either sender/receiver and tries to steal information or tries to perform actions as sender/receiver. Thus, the attacker gains access to control the traffic and creates fake transactions.
2) *Ping DDoS Flood Attack (Internet Control Message Protocol—ICMP):* Most common variant of the Distributed Denial-of-Service (DDoS) attack, where continuous pings from the attacker overwhelm the server making it go offline and deny further connection requests.
3) *Modbus Query Flood Attack:* A variant of the DDoS attack in Modbus [48], where the attacker sends a flood of messages to overwhelm the end device and make it unavailable to serve genuine message packets.
4) *SYN DDoS Attack:* In a Syn attack, repeated syn packets are sent to the server to initiate a connection handshake in an attempt to keep all the ports busy and disabling the server to offer more open connection ports to accept connections. The SYN DDoS attack is usually achieved using a bot that sends repeated connection requests by hiding the actual device Internet protocol (IP) address and sends many requests with spoof IPs. The famous Mirai attack of IoT is categorized as SYN DDoS.

B. Evaluation Metrics

In the ML space, the performance of trained model predictions is compared with actual values, and based on the comparison results, true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) are calculated. TP and TN represent the number of instances that the ML model predictions match with real labels/actual values while FP and FN count the number of instances where the ML model has predicted incorrect values. We have evaluated our approach and compared the results against the peer approach using the following metrics.

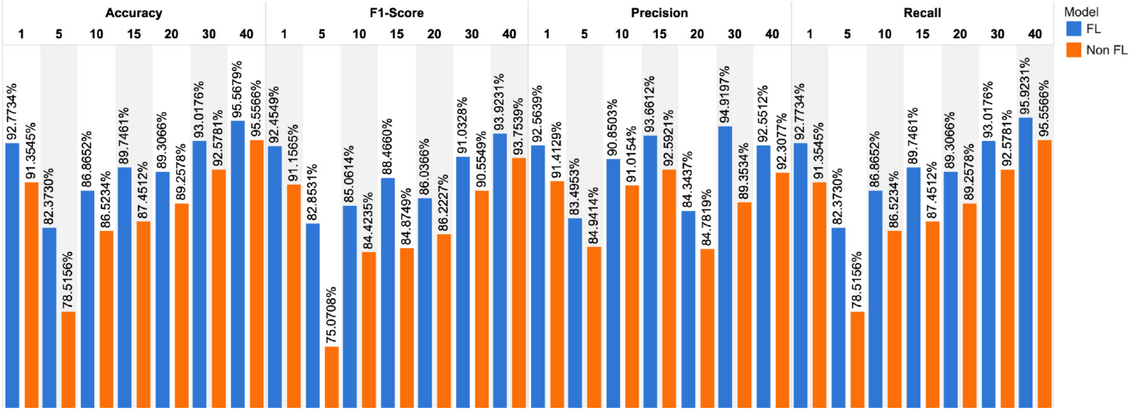

Fig. 7. GRU-3: Model evaluation results of the proposed FL approach and non-FL approach.

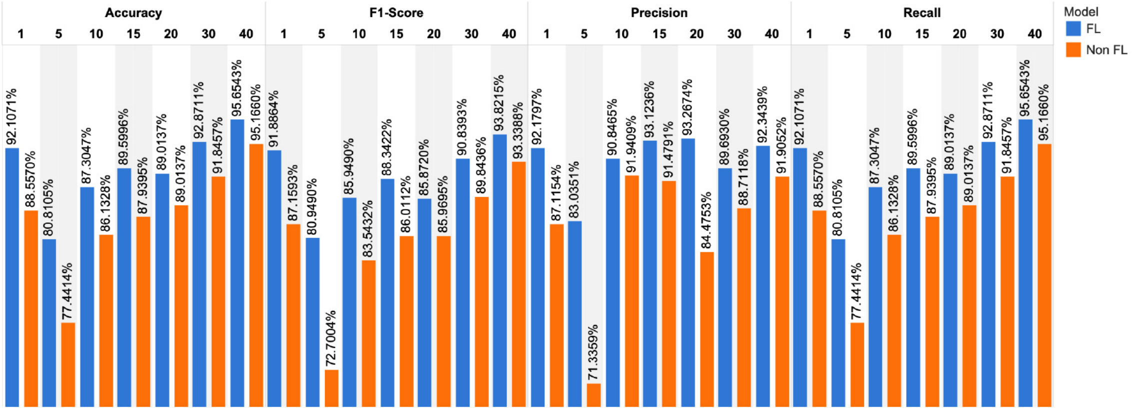

Fig. 8. GRU-4: Model evaluation results of the proposed FL approach and non-FL approach.

1) *Accuracy:*

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

2) *Recall:*

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

3) *Precision:*

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

4) *F1-Score:*

$$F1 - \text{Score} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}}.$$

5) *Training Time:* The total amount of time taken to complete a training round in FL includes additional time taken for the FL aggregation logic and communication time while interacting with each FL client. The below formulas give the calculations

$$\begin{aligned}
&\text{Training Time(FL)}\\
&\quad = \text{Time}\big[t_r(fl_1) + t_r(fl_2) \cdots t_r(fl_n)\big]\\
&\qquad + \text{Time}\big[c(fl_1) + c(fl_2) \cdots c(fl_n)\big] + \text{Time}\big[fl_{\text{average}}\big]\\
&\text{Training Time(Non FL)}\\
&\quad = \text{Time}\big[\text{training}(\text{dataSource}_1)\\
&\qquad\qquad + \text{training}(\text{dataSource}_2) \cdots\\
&\qquad\qquad + \text{training}(\text{dataSource}_n)\big]
\end{aligned}$$

where fl_1, $fl_2, \ldots, fl_n$ denotes FL clients, Time($t_r(fl_i)$) represents time taken for training round of the ith FL client, Time(c(fl_i) is for time taken for the central server to communicate for ith FL client training round, *training*(*dataSource*$_i$) is the time taken for training ith data source, and Time(fl_{average}) is for total time taken for execution on the FL averaging algorithm.

We have used Skorch[10] and scikit-learn[11] for evaluating the trained ML models. As our approach is based on the Pytorch deep learning framework, we have leveraged skorch wrapper to avail the evaluation packages provided by the scikit-learn framework.

C. Results

The evaluation results of our proposed approach in comparison to non-FL implementation are listed in Figs. 5, 6, 7, and 8. In comparison to the non-FL version, our proposed version with FL achieves greater accuracy with a limited number of epochs. The existence of multiple fl_n computing instances shares computational power and parallel computing saves overall training time taken to reach the optimal

[10] https://github.com/skorch-dev/skorch
[11] https://scikit-learn.org/

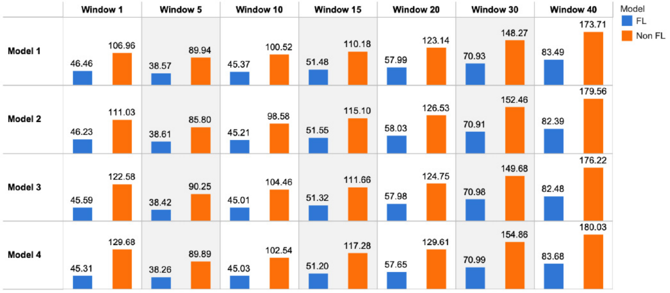

Fig. 9. Training Time: For different window sizes for the proposed FL approach and non-FL approach.

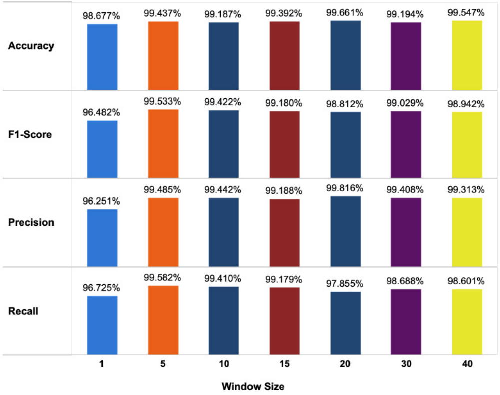

Fig. 10. Cross-validation results: For different window sizes of the proposed FL approach.

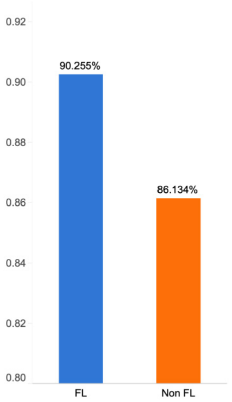

Fig. 11. Average accuracy of the proposed FL approach and non-FL approach.

performance state. The average of the total time taken for each epoch is shown in Fig. 9, as the FL approach shares computational resources with fl_n end-devices total time that is considerably less even with the additional time taken for FL averaging.

Fig. 10 illustrates the performance of FL integration with ensembler for all the window sizes. The average accuracy of each is 99.5%, making the anomaly detection rate high with a minimal number of false alarms. The overall average accuracy of FL in comparison to non-FL is illustrated in Fig. 11.

To summarize, evaluation results emphasize the fact that our proposed approach outperforms non-FL implementation. For demonstration purposes, we have implemented FL with the ML model built from scratch, but in realistic practical productions implementation of FL, the global model can be trained with known attacks and sample training data set. This helps the global model to get pretrained and the live data from IoT devices keeping the global model up-to-date,z and optimizes the performance to the maximum level and predicts the attack probability more accurately. Our approach was evaluated with virtual instances that are incapable of generating live logs and rely on the data distributed before the training round. Even with limitations of virtual instances and add-on local model averaging time, the evaluation results suggest that in the practical scenario with a productionized version of IoT networks, our proposed approach can further enhance the attack detection classification.

D. *Implications*

This article provides useful practical implications for adapters of FL. The proposed approach can serve as a good starting point to design architecture for migrating non-FL-based approaches to FL. The results section gives insight into the performance of GRUs for different window sizes and layer sizes, which can be further utilized to avoid cold start problems in FL. Our analysis in this work can help to fasten the process of promoting an ML product to FL in a production environment. Moreover, the proposed FL-based solution mitigates the disadvantages of non-FL-based intrusion detection systems while the decentralized training empowers end-device data security and enables sharing of computational resources that might very well result in efficient and greener ML-based products.

V. Conclusion and Future Work

In this article, we proposed a federated learning-based anomaly detection for accurate identification and classification of attacks in IoT networks. The FL implementation part of our proposed approach shares computational power with on-device training, and different layers of GRUs ensure higher accuracy rates in classifying attacks. The performance of the approach is improved further with the ensembler, which combines the predictions from different layers of GRUs. The FL benefits of user data privacy add a secure layer in IoT networks, making IoT devices more reliable. Our evaluation results demonstrate that our proposed approach outperforms the non-FL version of intrusion detection algorithms. Our future work is to enhance the proposed approach with a testbed of IoT devices and evaluate it with live data from device-specific data sets, which can classify all known and unknown vulnerabilities of IoT devices.

References

[1] L. Catarinucci *et al.*, “An IoT-aware architecture for smart healthcare systems,” *IEEE Internet Things J.*, vol. 2, no. 6, pp. 515–526, Dec. 2015.

[2] T. Ammari, J. Kaye, J. Y. Tsai, and F. Bentley, “Music, search, and IoT: How people (really) use voice assistants,” *ACM Trans. Comput. Hum. Interact.*, vol. 26, no. 3, p. 17, 2019.

[3] H. Ghayvat, S. Mukhopadhyay, X. Gui, and N. Suryadevara, “WSN- and IoT-based smart homes and their extension to smart buildings,” *Sensors*, vol. 15, no. 5, pp. 10350–10379, 2015.

[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[5] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, 3rd Quart., 2019.

[6] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1606–1616, Apr. 2019.

[7] A. C. Panchal, V. M. Khadse, and P. N. Mahalle, "Security issues in IIoT: A comprehensive survey of attacks on IIoT and its countermeasures," in *Proc. IEEE Global Conf. Wireless Comput. Netw. (GCWCN)*, Lonavala, India, 2018, pp. 124–130.

[8] W. Al Amiri, M. Baza, M. Mahmoud, K. Banawan, W. Alasmary, and K. Akkaya, "Privacy-preserving smart parking system using blockchain and private information retrieval," in *Proc. IEEE Int. Conf. Smart Appl. Commun. Netw. (SmartNets)*, Sharm El Sheikh, Egypt, 2020, pp. 1–6.

[9] J. Wu, M. Dong, K. Ota, J. Li, and W. Yang, "Application-aware consensus management for software-defined intelligent blockchain in IoT," *IEEE Netw.*, vol. 34, no. 1, pp. 69–75, Jan./Feb. 2020.

[10] H. Liang, J. Wu, S. Mumtaz, J. Li, X. Lin, and M. Wen, "MBID: Micro-blockchain-based geographical dynamic intrusion detection for V2X," *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 77–83, Oct. 2019.

[11] H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, M. Aledhari, and H. Karimipour, "A survey on Internet of Things security: Requirements, challenges, and solutions," *Internet Things*, to be published.

[12] M. Baza, A. Salazar, M. Mahmoud, M. Abdallah, and K. Akkaya, "On sharing models instead of data using mimic learning for smart health applications," in *Proc. IEEE Int. Conf. Informat. IoT Enabling Technol. (ICIoT)*, Doha, Qatar, 2020, pp. 231–236.

[13] M. Saharkhizan, A. Azmoodeh, A. Dehghantanha, K.-K. R. Choo, and R. M. Parizi, "An ensemble of deep recurrent neural networks for detecting IoT cyber attacks using network traffic," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8852–8859, Sep. 2020.

[14] N. K. Sahu and I. Mukherjee, "Machine learning based anomaly detection for IoT network: (Anomaly detection in IoT network)," in *Proc. 4th Int. Conf. Trends Electron. Informat. (ICOEI)*, Tirunelveli, India, 2020, pp. 787–794.

[15] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

[16] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Feb. 2021.

[17] A. Hard *et al.*, "Federated learning for mobile keyboard prediction," 2019. [Online]. Available: arXiv:1811.03604.

[18] I. Frazão, P. H. Abreu, T. Cruz, H. Araújo, and P. Simões, "Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process," in *Proc. Int. Conf. Crit. Inf. Infrastruct. Security*, 2018, pp. 230–235.

[19] M. S. Mekala, A. Jolfaei, G. Srivastava, X. Zheng, A. Anvari-Moghaddam, and P. Viswanathan, "Resource offload consolidation based on deep-reinforcement learning approach in cyber-physical systems," *IEEE Trans. Emerg. Topics Comput. Intell.*, early access, Dec. 28, 2020, doi: 10.1109/TETCI.2020.3044082.

[20] C. Thirumallai, M. S. Mekala, V. Perumal, P. Rizwan, and A. H. Gandomi, "Machine learning inspired phishing detection (PD) for efficient classification and secure storage distribution (SSD) for cloud-IoT application," in *Proc. IEEE Symp. Comput. Intell. (SSCI)*, Canberra, ACT, Australia, 2020, pp. 202–210.

[21] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Dallas, TX, USA, 2019, pp. 756–767.

[22] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615–5624, Aug. 2021.

[23] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, "Intrusion detection for wireless edge networks based on federated learning," *IEEE Access*, vol. 8, pp. 217463–217472, 2020.

[24] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, Ottawa, ON, Canada, 2009, pp. 1–6.

[25] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3, pp. 479–482, 2018.

[26] I. Almomani, B. Al-Kasasbeh, and M. AL-Akhras, "WSN-DS: A dataset for intrusion detection systems in wireless sensor networks," *J. Sens.*, vol. 2016, pp. 1–16, Sep. 2016.

[27] B. Cetin, A. Lazar, J. Kim, A. Sim, and K. Wu, "Federated wireless network intrusion detection," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2019, pp. 6004–6006.

[28] N. A. Al-Athba Al-Marri, B. S. Ciftler, and M. M. Abdallah, "Federated mimic learning for privacy preserving intrusion detection," in *Proc. IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom)*, Odessa, Ukraine, 2020, pp. 1–6.

[29] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Netw.*, vol. 34, no. 6, pp. 310–317, Nov./Dec. 2020.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[31] T. Ryffel *et al.*, "A generic framework for privacy preserving deep learning," 2018. [Online]. Available: arXiv:1811.04017.

[32] Y. Liu *et al.*, "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6348–6358, Apr. 2021.

[33] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Netw. IJCNN Neural Comput. New Challenges Perspect. New Millennium*, vol. 3. Como, Italy, 2000, pp. 189–194.

[34] H. Sak, A. W. Senior, and F. Beaufays, *Long Short-Term Memory Recurrent Neural Network Architectures For Large Scale Acoustic Modeling*, Google, Mountain View, CA, USA, 2014.

[35] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014. [Online]. Available: arXiv:1406.1078.

[36] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proc. Eur. Symp. Artif. Neural Netw. Comput. Intell. Mach. Learn.*, vol. 89, 2015, pp. 89–94.

[37] E. H. Bahadur, A. K. M. Masum, A. Barua, M. G. R. Alam, M. A. U. Z. Chowdhury, and M. R. Alam, "LSTM based approach for diabetic symptomatic activity recognition using smartphone sensors," in *Proc. IEEE 22nd Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dhaka, Bangladesh, 2019, pp. 1–6.

[38] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in *Proc. Int. Conf. Adv. Comput. Commun. Informat. (ICACCI)*, Udupi, India, 2017, pp. 1643–1647.

[39] Y. Luan and S. Lin, "Research on text classification based on CNN and LSTM," in *Proc. IEEE Int. Conf. Artif. Intell. Comput. Appl. (ICAICA)*, Dalian, China, 2019, pp. 352–355.

[40] H. Xiao *et al.*, "An improved LSTM model for behavior recognition of intelligent vehicles," *IEEE Access*, vol. 8, pp. 101514–101527, 2020.

[41] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014. [Online]. Available: https://arxiv.org/abs/1412.3555.

[42] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related," in *Proc. 2nd Int. Conf. Inf. Syst. Security Privacy (ICISSP)*, 2016, pp. 407–414.

[43] T. G. Dietterichl, "Ensemble learning," in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Cambridge, MA, USA: MIT Press, 2002, pp. 405–408.

[44] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[45] S. Jaloudi, "Communication protocols of an industrial Internet of Things environment: A comparative study," *Future Internet*, vol. 11, no. 3, p. 66, 2019.

[46] F. Shu, H. Lu, and Y. Ding, "Novel modbus adaptation method for IoT gateway," in *Proc. IEEE 3rd Inf. Technol. Netw. Electron. Autom. Control Conf. (ITNEC)*, Chengdu, China, 2019, pp. 632–637.

[47] Z. Drias, A. Serhrouchni, and O. Vogel, "Taxonomy of attacks on industrial control protocols," in *Proc. Int. Conf. Protocol Eng. (ICPE) Int. Conf. New Technol. Distrib. Syst. (NTDS)*, Paris, France, 2015, pp. 1–6.

[48] S. Bhatia, N. Kush, C. Djamaludin, J. Akande, and E. Foo, "Practical modbus flooding attack and detection," in *Proc. 12th Aust. Inf. Security Conf. Vol. 149*, 2014, pp. 57–65.