Federated Learning-based Anomaly Detection for IoT Security Attacks

Viraaji Mothukuri, Prachi Khare, Reza M. Parizi, *Senior Member, IEEE*, Seyedamin Pouriyeh, *Associate Member, IEEE*, Ali Dehghantanha, *Senior Member, IEEE*, 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. Huge interest has 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 is widely available and transferable to the main server because data is born at the edge and is 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 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 the 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*—Internet of Things, Security, Federated Learning, Recurrent neural networks, Gated Recurrent Units.**

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 towards 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 Internet of Things (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 micro-architecture 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. Research work in [5], [6] explores the vulnerabilities in IoT and feature-based security risks are explored and authors in [7] discuss 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 micro-devices delivering intelligent digital assistance has been tremendously appreciated and proven to reduce 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 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 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.

V. Mothukuri, P. Khare, R. M. Parizi and S. Pouriyeh are with the College of Computing and Software Engineering, Kennesaw State University, GA, USA, e-mails: vmothuku@students.kennesaw.edu, pkhare@students.kennesaw.edu, rparizi1@kennesaw.edu, spouriye@kennesaw.edu

A. Dehghantanha is with the Cyber Science Lab, University of Guelph, Canada, email: adehghan@uoguelph.ca

G. Srivastava is with Department of Math and Computer Science, Brandon University, Manitoba, Canada and also with the Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan.

Correspondence: Gautam Srivastava: srivastavag@brandonu.ca

Manuscript received 2020; revised 2021.

3) Training huge volumes of data on a single server can be computationally expensive.

One of the promising and well adaptable approaches which 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 paper, we propose a decentralized federated learning 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 (LSTMs) 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 dataset [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:

- Enabling on-device ML training with federated learning to secure data privacy at end-devices.
- Achieving higher accuracy rates and minimized false alarms in attack detection compared to a centralized ML (non-FL) approach.
- Demonstrating the benefits of integrating federated learning with ensembler to achieve optimal results.

The remainder of the paper 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 dataset, metrics, and evaluation results and summarizes our findings. Finally, Section V concludes the paper.

II. Related Work

IoT is proven to be successful in delivering ML solutions in its micro-architecture 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.

The authors in [21] propose 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. 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 implementation of an FL-specific deep learning framework.

The authors in [22] propose an FL-based intrusion detection framework called Deepfed, for identifying threats in cyber-physical systems (CPSs). A combination of convolutional neural networks (CNNs) and Gated recurrent units (GRUs) are 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. 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. Evaluation results on three datasets [24]–[26] show promising results supporting the proposed approach. Similarly, authors in [27], propose FL based approach for wireless intrusion detection (WID) with the awid dataset[2]. In [28], the authors leverage a mimic learning strategy to implement federated learning and combine it with ML-based IDS. Another FL based approach presented by [29] is a FL based intrusion detection system (IDS) using 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 proposed TensorFlow based deep learning framework. Six LSTMs [30] of different layers are used as a threat detection algorithm. 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 is proposed in the [14] research work, where distinct attributes in the dataset are used to identify anomalies in the smart home IoT devices. The authors proposed 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. Authors in [32] use attention-based convolutional

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

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

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

neural network Long Short term memory (LSTMs) for identifying anomalies in time series data if industrial IoT (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, 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 propose 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.

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

A. LSTMs and GRUs

In this part, we give an overview of the deep learning ML models we have used in our proposed approach. Long Short Term Memory Networks (LSTMs) [30], [33], [34] and Gated Recurrent Units (GRUs) [35] is a variation of RNNs which are proposed to address the short-term memory/vanishing gradients problem in a basic variant of recurrent neural networks (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].

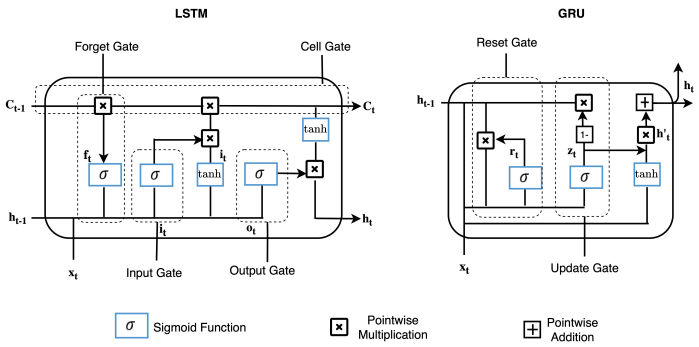

Fig. 1: Illustration of LSTM and GRUs

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

- **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 needs to be kept for future updates.

- **Tangent hyperbolic (*tanh*)**: An activation function generates values between -1 and 1. This ensures 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.

- **Cell State**: Represents the information retained throughout the memory block of LSTM. C_t represents current cell state or memory cell and C_{t-1} represent previous cell state.

- **Gates of LSTMs**: Based on the human's ability to memorize rhyming patterns, LSTMs are proposed as a memory block where interconnected memory cells collect and retain information for long-term reference. Gates are used to controlling information retention, retrieval, and deletion through memory cells. LSTMs consist of an Input gate, Forget gate, and Output gate. Below are details of each gate.

  – **Forget Gate**: The information which does not contribute towards the learning of LSTM network is discarded for the given cell as shown in Equation 1.

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

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

[4] https://pytorch.org

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

information from previous cell, respectively.

– **Input Gate**: Helps to determine whether the current information is useful enough to retain it in the cell state for future reference. Equations 2, 3, and 4 represent the calculation of input gate where 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.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$
$$\hat{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (3)$$
$$c_t = f_t * c_{t-1} + i_t * \hat{c}_t, \quad (4)$$

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 previous timestamp memory cell, c_t calculates the current cell value which is the information which is predicted as important to save for future reference, respectively.

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

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$
$$h_t = o_t * \tanh(c_t), \quad (6)$$

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

- **Gates of GRUs:** GRUs contains much simpler architecture, in comparison to LSTMs. Input to each memory cell is combined as a single value instead of two and works well with just two gates, the Reset gate and Update gate. GRUs are computationally inexpensive and take less time to train.
  – *Reset Gate*: Similar to forget gate of LSTMs, the information is discarded if it is not useful for future learning/reference using Equation 7.

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

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.

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

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

$$\tilde{h_t} = \tanh(W[r_t * h_{t-1}, x_t]) \quad (9)$$

$$h_t = (1 - z_t * h_{t-1} + z_t * \tilde{h_t}), \quad (10)$$

where i_t is the Results of the sigmoid layer, $\hat{h}_t$ is the Vector which is created by 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. Selection of window size is crucial as the amount of data differs for each window size which contributes towards 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 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 is dependent on the type and size of the dataset.

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

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 subsection), 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 which 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 practical scenario steps for creating Virtual Instances and Pre-process of captured data at a central server will not be necessary as real data for training is available at end-devices.

1) **Virtual Instances**: We replicate the IoT network set-up by creating virtual instances using PySyft[6],[7] [31]. For 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 trained ML model parameters between IoT mobile end-devices and the central FL server. Dataset is divided into n chunk and each is distributed to fl_n virtual instances.
2) **Pre-process of captured data**: As part of data pre-processing at each IoT device/IoT gateway (which is a

[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

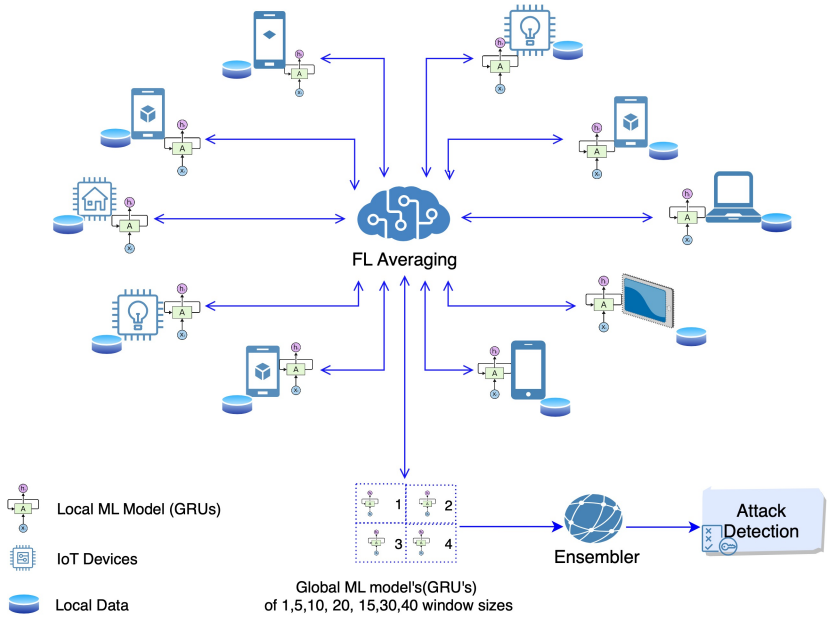

Fig. 2: High-level architecture of the proposed approach

connecting component between a set of IoT devices and digital platform/cloud) network data is captured in *pcap* files and the *pcap* files are converted to CSV files using CICFlowmeter tool [42]. The converted CSV file is processed further to clean the features which do not add value to the learning process. Finally, the processed data is 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 dataset 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}. 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' output's 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 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 given input X. 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 give input data X.

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

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

In Equation 12, the prediction function $f(x)$ of rfc gets input from prediction probability values of seven ML models M_{wi} for each label $y = y_{Clean}, y_{MITM}, y_{pingDDos}, y_{modbusqueryflood}, \ldots, y_{synDDos}$ which represents attack category in the dataset (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.

[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 */
$Attack_{Prediction}$ =$Ensembler(M_{w_i})$

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 dataset 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 non-FL/classic ML approach we have used Pytorch deep learning framework. While FL implements Algorithm 1, non-FL-based GRUs setup trains on classic environment set-up of centralized training data.

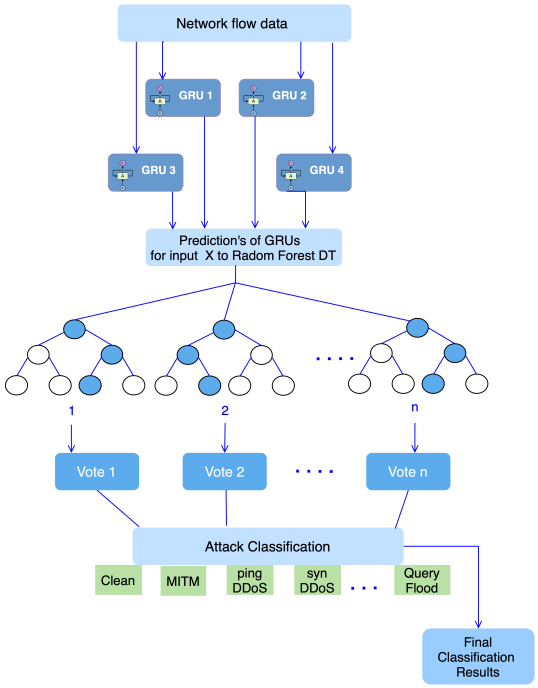

Fig. 3: Illustration of Ensembler in proposed approach

A. Dataset

For evaluation of our approach, we have used a Modbus-based network dataset [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 know 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. 6 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.

Several research works use the Modbus protocol for IoT, especially for IIoT. Authors in [45] suggest 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, research work in [46] proposes an approach to adapt Modbus protocol for different IoT sensor devices. However, Modbus protocol is vulnerable to many attacks [47], of which the dataset we chose contains the below-summarized attacks:

- **Man-in-the-middle Attac**k: 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.

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

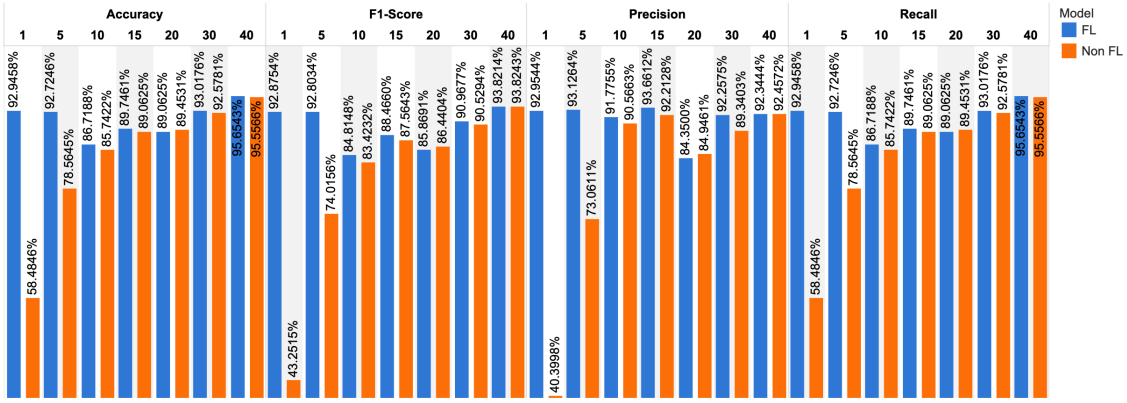

Fig. 4: GRU-1: Model evaluation results of proposed FL approach and Non-FL approach

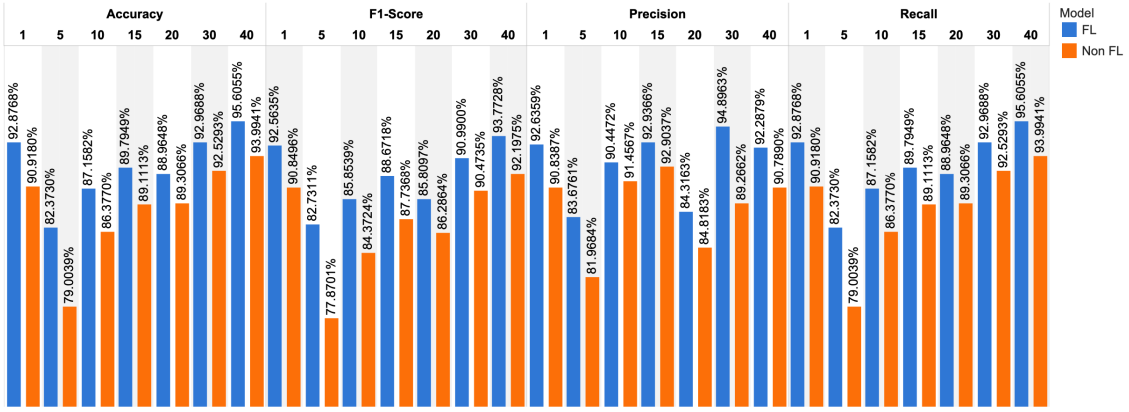

Fig. 5: GRU-2: Model evaluation results of proposed FL approach and Non-FL approach

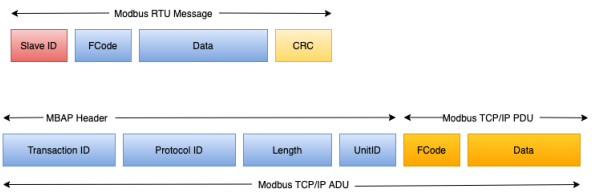

Fig. 6: Modbus Message Format

- **Ping DDoS Flood Attack (Internet control message protocol - ICMP)**: Most common variant of Distributed Denial of service (DDoS) attack, where continuous pings from the attacker overwhelm the server making it go offline and deny further connection requests.
- **Modbus Query Flood Attack**: A variant of 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.
- **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. 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 ML space, the performance of trained model predictions are 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.

- **Accuracy**

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

- **Recall**

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

- **Precision**

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

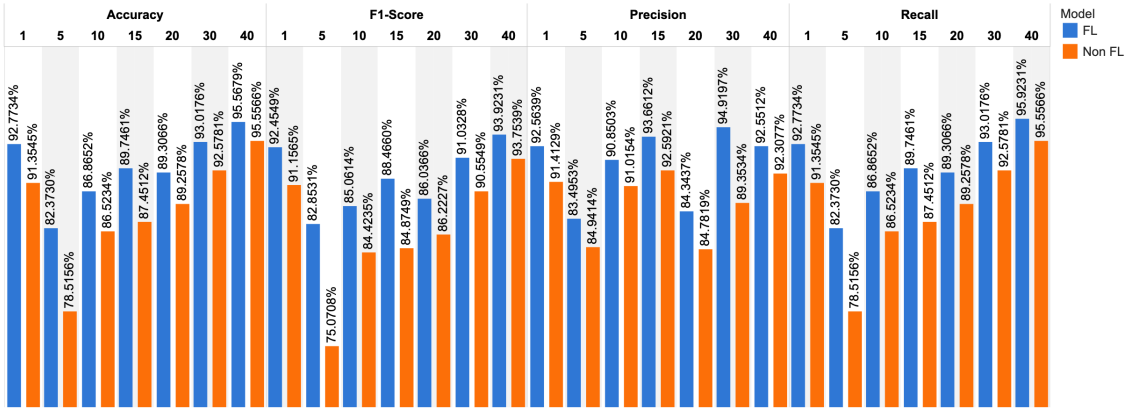

Fig. 7: GRU-3: Model evaluation results of proposed FL approach and Non-FL approach

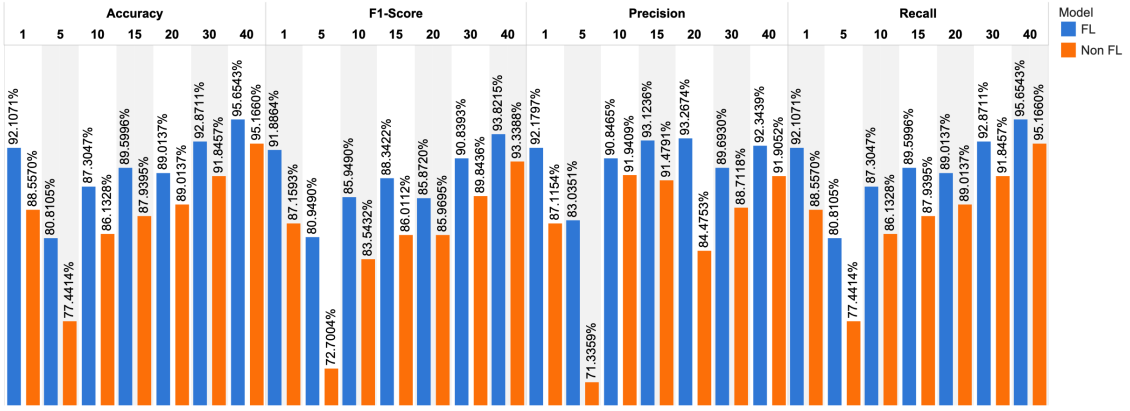

Fig. 8: GRU-4: Model evaluation results of proposed FL approach and Non-FL approach

- **F1-Score**

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

- **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:

$$\begin{aligned} TrainingTime(FL) = Time[&t_r(fl_1) + t_r(fl_2)...t_r(fl_n)] \\ &+ Time[c(fl_1) + c(fl_2)...c(fl_n)] \\ &+ Time[fl_{average}] \end{aligned}$$

$$\begin{aligned} TrainingTime(NonFL) = Time[&training(dataSource_1) \\ &+ training(dataSource_2)... \\ &+ training(dataSource_n)] \end{aligned}$$

where fl_1, fl_2,...fl_n denotes FL clients, Time($t_r(fl_i)$) represents time taken for training round of i^{th} FL client, Time(c(fl_i) is for time taken for central server to communicate for i_{th} FL client training round $training(dataSource_i)$ is the time taken for training i_{th} data source and Time($fl_{average}$) is for total time taken for execution on FL averaging algorithm.

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

C. Results

Evaluation results of our proposed approach in comparison to non-FL implementation are listed in Figs. 4, 5, 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 share computational power and parallel computing saves overall training time taken to reach optimal 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 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. Average accuracy of each

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

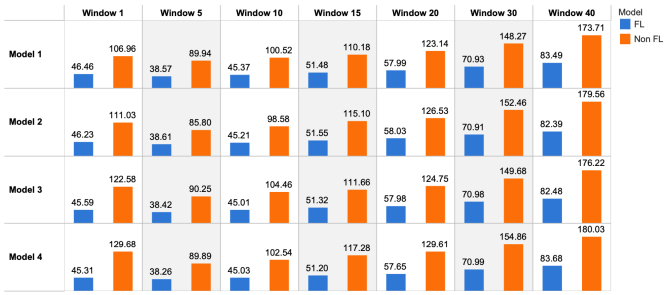

Fig. 9: Training Time: For Different Window Sizes for proposed FL approach and Non-FL approach

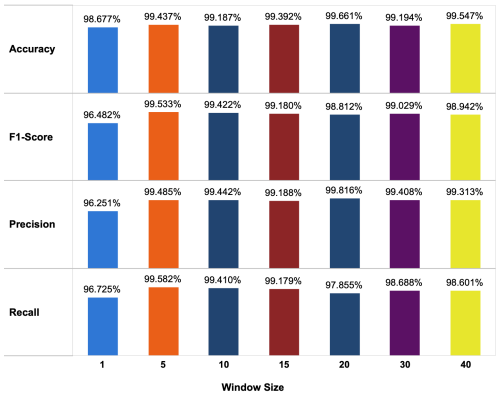

Fig. 10: Cross validation results: For Different Window Sizes of proposed FL approach

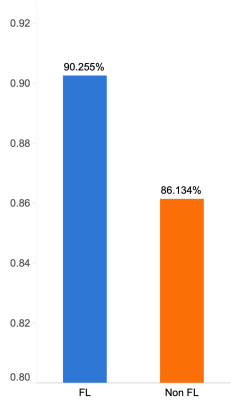

Fig. 11: Average Accuracy of proposed FL approach and Non-FL approach

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 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 dataset. This helps the global model to get pre-trained and the live data from IoT devices keeping the global model up-to-date 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, evaluation results suggest that in the practical scenario with a productionized version of IoT networks our proposed approach can further enhance attack detection classification.

D. Implications

This paper 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 paper, 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 datasets which can classify all known and unknown vulnerabilities of IoT devices.

References

[1] L. Catarinucci, D. De Donno, L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and L. Tarricone, "An iot-aware architecture for smart healthcare systems," *IEEE internet of things journal*, vol. 2, no. 6, pp. 515–526, 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, pp. 17–1, 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. 10 350–10 379, 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 Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2702–2733, 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 of Things Journal*, vol. 6, no. 2, pp. 1606–1616, 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 *2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN)*, 2018, pp. 124–130.

[8] W. Al Amiri, M. Baza, M. Mahmoud, b. K. Banawan, W. Alasmary, and K. Akkaya, "Privacy-preserving smart parking system using blockchain and private information retrieval," *Proc. of the IEEE International Conference on Smart Applications, Communications and Networking (SmartNets 2019)*, 2020.

[9] J. Wu, M. Dong, K. Ota, J. Li, and W. Yang, "Application-aware consensus management for software-defined intelligent blockchain in iot," *IEEE Network*, vol. 34, no. 1, pp. 69–75, 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 Communications Magazine*, vol. 57, no. 10, pp. 77–83, 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 of Things*, p. 100129, 2019.

[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 *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, 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 of Things Journal*, vol. 7, no. 9, pp. 8852–8859, 2020.

[14] N. K. Sahu and I. Mukherjee, "Machine learning based anomaly detection for iot network: (anomaly detection in iot network)," in *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, 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. 140 699–140 725, 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 Generation Computer Systems*, 2020.

[17] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2019.

[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 *International Conference on Critical Information Infrastructures Security*. Springer, 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 Transactions on Emerging Topics in Computational Intelligence*, pp. 1–10, 2020.

[20] M. S. Mekala and V. Perumal, "Machine learning inspired phishing detection (pd) for efficient classification and secure storage distribution (ssd) for cloud-iot application," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 202–210.

[21] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A. Sadeghi, "DÏot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 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 Transactions on Industrial Informatics*, pp. 1–1, 2020.

[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. 217 463–217 472, 2020.

[24] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[25] R. Panigrahi and S. Borah, "A detailed analysis of cicids2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3.24, pp. 479–482, 2018.

[26] I. Almomani, B. Kasasbeh, and M. AL-Akhras, "Wsn-ds: A dataset for intrusion detection systems in wireless sensor networks," *Journal of Sensors*, vol. 2016, pp. 1–16, 01 2016.

[27] B. Cetin, A. Lazar, J. Kim, A. Sim, and K. Wu, "Federated wireless network intrusion detection," in *2019 IEEE International Conference on 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 *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 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 Network*, vol. 34, no. 6, pp. 310–317, 2020.

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

[31] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," *arXiv preprint arXiv:1811.04017*, 2018.

[32] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[33] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3, 2000, pp. 189–194 vol.3.

[34] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Google*, 2014.

[35] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[36] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89. Presses universitaires de Louvain, 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 *2019 22nd International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2019, pp. 1–6.

[38] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," in *2017 international conference on advances in computing, communications and informatics (icacci)*. IEEE, 2017, pp. 1643–1647.

[39] Y. Luan and S. Lin, "Research on text classification based on cnn and lstm," in *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. IEEE, 2019, pp. 352–355.

[40] H. Xiao, M. A. Sotelo, Y. Ma, B. Cao, Y. Zhou, Y. Xu, R. Wang, and Z. Li, "An improved lstm model for behavior recognition of intelligent vehicles," *IEEE Access*, 2020.

[41] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[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 *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.

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

[44] L. Breiman, "Random forests," *Machine learning*, 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 *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2019, pp. 632–637.

[47] Z. Drias, A. Serhrouchni, and O. Vogel, "Taxonomy of attacks on industrial control protocols," in *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, 2015, pp. 1–6.

[48] S. Bhatia, N. Kush, C. Djamaludin, J. Akande, and E. Foo, "Practical modbus flooding attack and detection," in *Proceedings of the Twelfth Australasian Information Security Conference - Volume 149*, ser. AISC '14. Australian Computer Society, Inc., 2014, p. 57–65.