Memristor Based Federated Learning for Network Security on the Edge using Processing in Memory (PIM) Computing

Shahanur Alam, Chris Yakopcic, and Tarek M. Taha
Dept. Of Electrical and Computer Engineering, *University of Dayton*
Dayton, OH, USA
{alamm8, cyakopcic1, tarek.taha}@udayton.edu

***Abstract*—Artificial Intelligence (AI) is moving towards the edge. Training an AI model for edge computing on a centralized server increases latency, and the privacy of edge users is jeopardized due to private data transfer through a less secure communication channels. Additionally, existing high-power computing systems are battling with memory and data transfer bottlenecks between the processor and memory. Federated Learning (FL) is a collaborative AI learning paradigm for distributed local devices that operates without transferring local data. Local participant devices share the updated network parameters with the central server instead of sending the original data. The central server updates the global AI model and deploys the model to the local clients. As the local data resides only on the edge, these devices need to be protected from cyberattacks. The Federated Intrusion Detection System (FIDS) could be a viable system to protect edge devices as opposed to a centralized protection system. However, on-device training of the model in resource constrained devices may suffer from excessive power drain, in addition to memory and area overhead.**

In this work we present a memristor based system for AI training on edge devices. Memristor devices are ideal candidates for processing in memory, as their dynamic resistance properties allow them to perform multiply-add operations in parallel in the analog domain with extreme efficiency. Alternatively, existing CMOS-based PIM systems are typically developed for edge inference based on pretrained weights, and are not equipped for on-chip training. We show the effectiveness of the system, where successful learning and recognition is achieved completely within edge devices. The classification accuracy of the memristor system shows negligible loss when compared a software implementation. To the best of our knowledge, this first demonstration of a memristor based federated learning system. We demonstrate the effectiveness of this system as an intrusion detection platform for edge devices, although given the flexibility of the learning algorithm, it could be used to enhance many types of on board leaning and classification applications.

Keywords—PIM, memristor, Federated Learning, low power, Intrusion Detection

I. INTRODUCTION

Communication technology is moving towards the edge, and billions of smart devices are connected to the internet through communication channels. Typically, in current infrastructure, distributed mobile devices are sharing information with a cloud computing system. Thus, this existing method is vulnerable to information theft. As a result, several studies have been completed on Machine Learning (ML) and Artificial Neural Network (ANN) based Intrusion Detection System (IDS) development [1-3]. However, a cloud computing IDS still leaves vulnerabilities on the edge devices. Moreover, cloud computing is not suitable for applications that demand low latency and low power [1].

Recently, Federated Learning (FL) was proposed by Google AI as a new learning paradigm for distributed mobile devices [4]. The FL system has many potential applications in edge computing, healthcare, image recognition, and more [5]. In the FL method, the ANN is trained on the mobile devices on local data, and periodically shares the updated network information with the central server instead of sharing the original data. Then the central server adjusts the model in each communication round and deploys the updated model among the clients. In this work we present novel hardware to implement FL with extreme edge efficiency, and we demonstrate the utility of such a system by implementing a collaborative learning environment capable of sharing information about incoming cyber-attacks at the edge.

An FL-based Intrusion Detection System (FIDS) can collaboratively learn cyber intrusion types, share any intrusion knowledge, and make a robust protection system [6-8]. FIDS could be a viable alternative to the existing rule-based IDS (SNORT, BRO, etc) for protecting edge and IoT devices [6]. A rule-based IDS runs state-machine algorithms and requires a million lines of code to find the potential threat. Moreover, the SNORT system has a high false-positive rate, and human intervention is needed to correct it [9]. Thus, running a high volume of IDS programs on resource-constrained edge devices is not feasible.

Although the FL is a revolutionary ML model able to train distributed AI devices without sharing local data, the FL model is trained using the edge device's power supply [10]. The usage of the local devices may suffer from power outages, and these local systems now require additional memory to store network parameters before sending them to the server. Additionally, the fact that network devices could have widely varying power

consumption and computation ability may make the implementation of FL on a large scale challenging [10]. Thus it would be best for the chosen IDS to operate at extreme low power, and be non-volatile. A few works on FL implementation in hardware can be found [10-16] for resource constrained edge applications.

Neuromorphic [17] and Processing In Memory (PIM) [18] computing systems could be a viable solution for implementing FL in IoT or edge devices to address this resource bottleneck. The FL algorithm is implemented in a Spiking Neural Network (SNN) for the IoT and edge applications in [11,12,14,15]. However, no one has yet reported or proposed a FL implementation for on-chip and online learning in a PIM hardware system. PIM is primarily an analog in-memory computing paradigm. Commercially available PIM systems use SRAM cells to store quantized binary weights and target edge AI inference [19], not on-device training.

Non-Volatile Memory (NVM) devices are compatible with in-memory computing and ANNs, as they can efficiently implement vector-matrix multiplication [20]. The memristor is a popular NVM device and can be implemented in a resistive crosspoint array to perform multiplication and addition in the analog domain in a highly parallel fashion [20-24]. With supervised and unsupervised learning algorithms, the memristor-based ANN is implemented for image classification [25], network intrusion, and anomaly detection [24]. There is no reported implementation of the FL algorithm in memristor-based in-memory on-chip computing. Thus, the main contributions of this work are as follows:

1.This work provides the proof of concept for the implementation of FL in-memory training in a highly parallel form in a memristor-based ANN architecture.

2. FL for IDS applications in resource constrained edge devices using the proposed memristor-based PIM system.

The rest of the paper is organized as follows: Section II briefly describes the FL system and algorithm. Section III explains the related works and reviews the hardware implementation of FL in the literature. Section IV discusses the dataset that is used for this experiment. Section V discusses the memristor implementation of the FL system, as well as training for FL intrusion detection. Section VI is about the experimental setup for dataset distribution, training, and testing procedures. The results of memristor FL are discussed in section VII, and section VIII presents a power, energy, and timing analysis of the system. Section IX presents a brief conclusion on this work.

II. Federated Learning (FL)

FL is a collaborative and on-device ANN learning paradigm. First, the global model is deployed to all clients. The client devices learn based on local data, and share the updated synaptic weights through a communication protocol. Local data stays in the local device. Thus, helping to preserve the privacy of confidential information. The server receives the shared weights from all of the client devices, computes the average, updates the global model, and deploys the updates to the clients. The process is known as the FederatedAveraging (FedAvg) algorithm [26]. Fig. 1 presents a federated network where n client devices are connected to the central global server through a bidirectional communication channel.

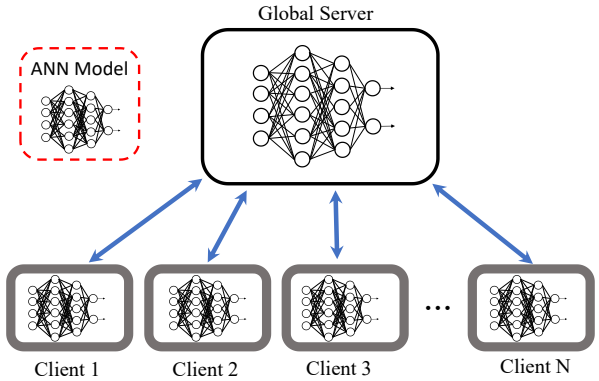

Fig. 1. Federated learning model diagram.

The functional algorithm for the FL [26] is presented in Fig. 2 with the network parameters W_0^G randomly initialized, and the same model is deployed among the clients. The clients are trained on the local data, and the updated weights w_{r+1}^i are then sent to the central server without sharing the actual local data. It is assumed that the clients are performing their local training simultaneously and sharing the information with the server periodically, so that the server does not need to wait for a particular client. In each communication round r, the central server averages the accumulated weights shared by the clients. After performing the averaging, the central server deploys the updated model w_{r+1}^G to the clients, and this process is then repeated until training is complete.

Algorithm: The C_r clients are indexed by i; D is the local data size indexed by d, E is the number of local epochs, R represents the communication rounds indexed as r, η is the learning rate, and θ represents the network parameters. n_i is the sample set on the local device i, which can be varied from device to device and n is the total number of samples present for all clients.

w_0^G ←*Randomly initialize the weights in the server and deploy to the clients*
for *each round r=1,2, 3…, R*
Randomly choose a subset of clients from C_r
for *each client* $i \in C_r$ *execute simultaneously*
for $e \in 1,2,\ldots,E$ *(local epoch)*
for $d \in D$
$w_{r+1}^i \leftarrow w_r^i - \eta \nabla l(w_r^i; b)$
end for
end for
end for
$w_{r+1}^G \leftarrow \sum_i^{C_r} \frac{n_i}{n} w_{r+1}^i \leftarrow$ *Global Averaging*
Test the updated global model with the test dataset
end for

Fig. 2. Functional algorithm for the federated learning system. This algorithm was taken from [26].

III. Related Work

FL is an on-device synergistic learning paradigm, and the training process takes place on-chip, on power constrained edge

devices. When these edge devices utilize traditional computing hardware, they suffer a memory and data transfer bottleneck.

A feasibility study implementing FL in resource constrained devices has been performed on portable devices (such as the Raspberry-PI) to detect emotion from sound [10]. The difficulties of implementing FL in the presence of heterogeneous hardware under a federated system are discussed in [27]. FL on battery-powered devices is studied in [16] using a two-fold FL training strategy. This approach eliminates the problem of missing clients.

The challenges of FL and future directions for implementation are discussed in the review in [10]. The findings of this work suggest that hardware improvement is required for on-device FL beyond algorithm development. Their study also suggests that to overcome the bottlenecks of traditional Von Neumann computing, the developer can explore non-Non Neumann computing paradigms. Neuromorphic [17] and Processing In Memory (PIM) [18] computing systems could be the viable solution for implementing FL in IoT or edge devices to address this bottleneck of resource-constrained computing devices. Neuromorphic and PIM computing systems are non von Neumann computing paradigms, and they consume very low computation power compared to traditional von Neumann computing.

A Spiking Neural Network (SNN) based FL-SNN implementation is proposed in [15] using feedback signals within each SNN instead of backpropagation. A one-shot FL learning method for gesture recognition in a neuromorphic processor is proposed, which preserves local device data [12]. Work in [14] proposed and validated the importance of SNN neuromorphic computing in FL for image recognition tasks while preserving local information.

Memristor-based neuromorphic systems have the potential to alleviate the stringent power constraints of on-chip learning in resource constrained devices. Yet, there is no reported FL implementation in the memristor-based in-memory computing system. However, the implementation of memristor based supervised and unsupervised learning systems is studied extensively [20-25].

FL for intrusion detection systems has previously only been reported using software implementations [6-8]. A FL algorithm for intrusion detection implemented with a multilayer perceptron (MLP) and probabilistic ensemble classifier distributor with the NSL-KDD dataset is reported, achieving 81-84% accuracy depending on various noise parameters. FL for intrusion detection is reported in [6] for centralized and federated setups of nine clients. The more basic FedAvg achieves around 87% accuracy. Training the network with selected features may affect the model's performance, to combat this a feature selection method is applied in the FL system for intrusion detection with accuracy between 49 and 78% on NSL-KDD data with various feature selection of the network packets in [7]. An anomaly-based intrusion detection system is reported with various data distribution and achieved 54-77% accuracy [8]. These intrusion detection examples are all implemented in software, and other than our presented system, there is no reported FL implementation for intrusion detection in hardware.

Alternatively, our work presents a proof of concept for implementing an FL algorithm on a memristor-based PIM system. The protection of edge devices from potential intrusion is another concern, thus we demonstrate how our memristor-based PIM for FL is able to implement a collaborative IDS.

IV. NSL-KDD Dataset

NSL-KDD is a popular dataset for designing and testing ANN-based intrusion detection systems [28]. NSL-KDD provides training and testing datasets separately, containing 125,973 network packets for training and 22,544 packets for testing the intrusion detection model. Both datasets are comprised of normal and malicious packets. Each packet has 43 attributes with nominal, binary, and numeric values. The nominal values are replaced with integers, and data contains five categories which are labeled from 0 to 4. The min-max normalization is performed on the entire dataset to bound the values in [0,1]. Examples of normal and malicious network packets are shown in Fig. 3 in their raw and normalized form.

The malicious packets fall mainly within 4 categories: DoS, Probe, U2R, and R2L. Due to the very low number of samples in the U2R category, this study considers only the other 3 malicious types, as well as Normal packets for training and testing the FL intrusion detection system.

```
0,tcp,http,SF,287,2251,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,3,
7,0.00,0.00,0.00,0.00,1.00,0.00,0.43,8,219,1.00,0.00,0.12
,0.03,0.00,0.00,0.00,0.00,normal,21
```
(a)

```
0,icmp,eco_i,SF,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,16,1.00,0.00,1.00,1.
00,0.00,0.00,0.00,0.00,ipsweep,18
```
(b)

```
0,0,0.04347,0,2.17e-07,1.05e-05,0,0,0,0,0,1,0,0,0,0,0,0,
0,0,0,0.0156,0.01761,0,0.11,0,0,1,0,0.22,0.3568,1,1,0,0.0
1,0.02,0,0,0,0,1
```
(c)

```
0,1,0.10,0,1.304e-08,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.0019,0.00195,0,0,0,0,1,0,0,0.0039,0.0627,1,0,1,1,0,0,0,
0,0.857
```
(d)

Fig. 3. Example network packets (a) and (b) directly from the NSL-KDD dataset and (c) and (d) preprocessed normalized packets ready for training and evaluation.

V. Memristor FL Implementation and Training

The memristor is an emerging NVM device examined significantly for online and on-chip learning [20-25]. This work implements memristor-based FL, and demonstrates how it can be used for network intrusion detection. The MLP network architecture is considered for this experiment.

A. Network Topologies

The general architecture for the MLP is displayed in Fig. 4. The MLP is a supervised network that can classify data samples within a known set of classes. The computation of the MLP is carried out according to equations (1-3). Here b and $f(x)$ denote the bias and activation function, respectively.

$$L_{1j} = f\left(\sum_{i=1}^{41} w_{1(i,j)}.x_i + b_{1j}\right) \quad (1)$$

$$L_{2k} = f\left(\sum_{j=1}^{90} w_{2(j,k)}.h_{1j} + b_{2k}\right) \quad (2)$$

$$L_{3l} = f\left(\sum_{k=1}^{10} w_{3(k,l)}.h_{2k} + b_{3l}\right) \quad (3)$$

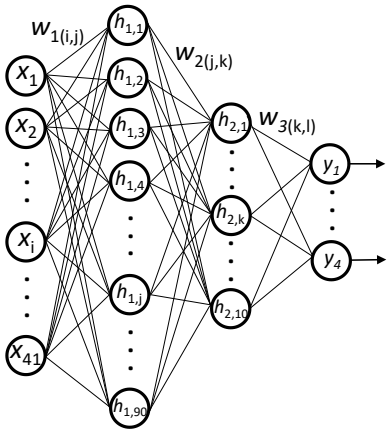

Fig. 4. Basic architecture for the MLP used in this study.

B. Memristor Based Implementation

The basic neuron circuit we use in this work is displayed in Fig. 5, and this pattern can be repeated within a crossbar to define an entire layer of neurons. Thus, this memristor based neuron layer can perform multiply and addition operations very efficiently in parallel in the analog domain [20]. This neuron circuit requires two memristors to represent a single synaptic weight.

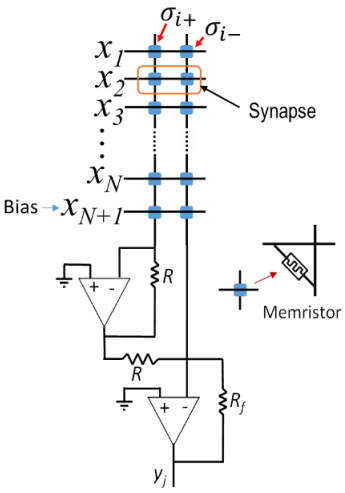

Fig. 5. Memristor based neuron circuit.

For a given input, a positive weight value is observed when $\sigma_{i+} > \sigma_{i-}$. Otherwise, the weight observed is negative [21,22,24]. This circuit is able to carry out multiply-add operations according to equation (4). The memristor devices modeled are assumed to have a resistance ratio of approximately 200 and a write threshold voltage of 1.3V, where $\sigma_{max} = 2 \times 10^{-5}\ \Omega^{-1}$ and $\sigma_{min} = 1 \times 10^{-7}\ \Omega^{-1}$.

$$DP_j = \sum_{i=1}^{N+1} x_i \times \left(\sigma_{ij}^{+} - \sigma_{ij}^{-}\right) \quad (4)$$

$$y_j = f(DP_j) \quad (5)$$

In Fig. 5, each input column is connected to a virtually grounded op-amp. The output y_j in equation (5) represents the neuron output after accounting for the activation function, approximating the rail voltages to squeeze the DP_j values effectively.

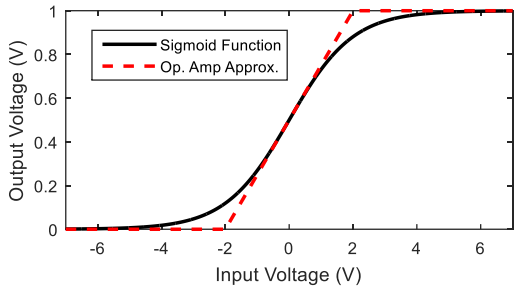

Fig. 6. Sigmoid activation function overlaid with the utilized approximation.

Equation (6) displays the typical sigmoid function used in deep learning. Equation (7) displays the approximated sigmoid activation function generated by the op-amps when the V_{DD} and V_{SS} of the second op-amp stage are connected to 0 and 1 V respectively. The plot in Fig. 6 displays an overlay of $f(x)$ and $g(x)$ to visually compare these two activation functions. Fig. 7 presents the crossbar layers implemented in the FL experiment client devices.

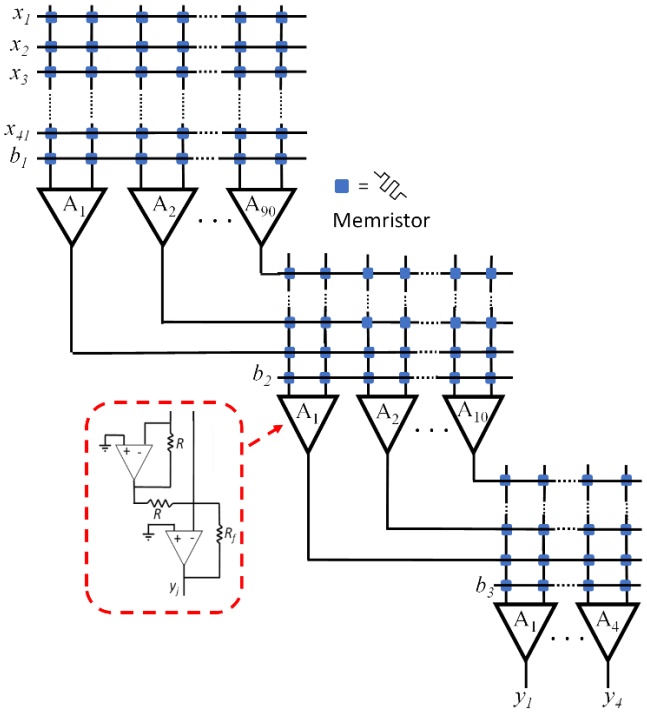

Fig. 7. Memristor crossbar circuits connected to build a multilayered neural network with inputs (x) and outputs (y). The column amplifier circuit is shown in the inset.

$$f(x) = \frac{1}{1+e^{-x}} \quad (6)$$

$$g(x) = \begin{cases} 1, & x > 2 \\ 0.25x + 0.5, & |x| \le 2 \\ 0, & x < 2 \end{cases} \quad (7)$$

C. Memristor Crossbar Training for FL

The training algorithm utilized is as follows:

i) Initialize the memristors with low random conductance in the central server and deploy the MLP model to the mobile devices or clients.

ii) For an input packet x at a local device:

a) Compute DP_j and y_j at the neuron outputs of each crossbar layer. The input is inserted in the memristor array on the horizontal wires.

b) For a neuron j, compute the error δ_j based on the output y_j and the target t_j using equation (8) with the derivative of the activation function $g(x)$.

$$\delta_j = (y_j - t_j)g'(DP_j) \quad (8)$$

c) Back propagate the error toward the input from each hidden layer neuron j according to equation (9). The computed error is inserted from the columns of the memristor array.

$$\delta_j = \sum_k \delta_k \, w_{k,j} g(DP_j) \quad (9)$$

d) Compute the weight update Δw_j according to the error function and update weight layers with learning rate η and Stochastic Gradient Descent (SGD) process. The gradients are computed in the digital domain and these are translated to write pulse of varying length then amplitude, that are able to update each memristor by the correct amount (according to the theoretical weight update equations).

e) Send the updated weight to the central server through the prescribed communication protocol in each communication round.

f) Receive the updated model from the central server and repeat the same training process on the local data.

iii) Return to step (ii) until the global model reaches satisfactory accuracy.

This approach allows us to simulate the way in-situ on-chip learning would be utilized within the memristor crossbars. Thus memristor resistance is tuned and optimized during the training process. In addition to the energy and throughput advantages, on-chip training is beneficial because it accounts for the variation in resistance present across an array of memristor devices [29].

VI. Experiment Setup

This experiment considers six edge clients in the federated network under a central server. First, a randomly initialized model is deployed to the clients to train on the individual datasets, then all clients are evaluated with a single test dataset to determine the model performance after each communication round. The experiment is performed for the ideal and extreme federated cases [4,26]. In an ideal federated case, all participants receive an equal number of classes for training their local devices. The ideally distributed dataset is known as the Independent and Identical Distributed (IID) dataset. If any clients do not receive data from all of the training categories, then the training system is based on a non-IID dataset [4,26]. Even if a client in the FL system does not receive data from all categories, once the central server updates the global model, this client will be able to recognize these categories in the future. Thus, a local device can perform inference on the packets that were not present during its own training procedure.

In this study, all features (except the label) in the network packets are used to train the model. At first, all of the data categories are distributed randomly in subgroups, and each subgroup contains 1,000 network packets. Then each subgroup is randomly sent to the participants, and it is assumed that training on the local devices is performed simultaneously.

This study includes two experiments. In the first experiment, all of the clients are trained with the IID dataset with all four classes (Normal, DoS, Probe, R2L) of the NSL-KDD dataset, and each client receives a random training dataset. In the second experiment, the clients receive a subset of the non-IID dataset that contains random samples, but each client may not receive all the dataset categories. The non-IID experiment was performed in various combinations: (1) all participants received two random classes; (2) each participant receives only one random class to learn (considered an extreme random case); (3) each client receives data from an unknown number of classes, it may be all four or it may be fewer.

After performing each of the FL training processes, the model is tested with a single test set to measure the various training processes fairly. The classification accuracy of the trained model is evaluated with equation (10).

$$Accuracy = \frac{\sum Total\ Population - \sum Wrong\ Detection}{\sum Total\ Population} \quad (10)$$

VII. Results and Discussion

The training result for each client in the model is presented in Fig. 8 and Fig. 9 for IID and non-IID training datasets, respectively. The results present the squared error in each communication round converging for individual clients.

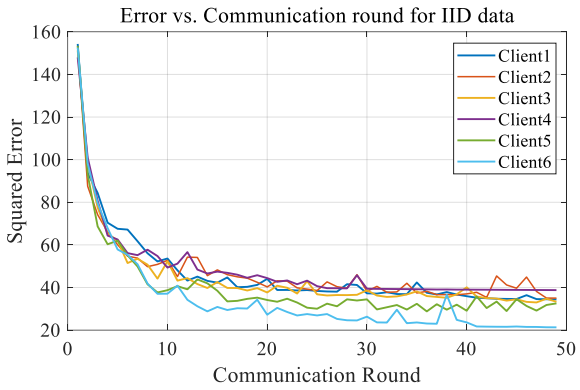

Fig. 8. Training error vs. epochs for each client in the memristor based FL system for IID data.

In Fig. 8, the convergence progress is similar for all six clients. Nevertheless, the convergence pattern within different clients is not entirely uniform, as the source of IID data may be different. The differences in convergence are exaggerated in local devices if they receive a non-IID dataset, as presented in Fig. 9.

Alternatively, Figs. 10 through 14 present the performance of the global model during each of the communication rounds, after receiving data from the memristor based clients. In the FL setup, the local devices train the model using local data, and then send the updated weights to the central server during each communication round. The local devices each train on their own local data for 5 epochs between each communication rounds for these experiments.

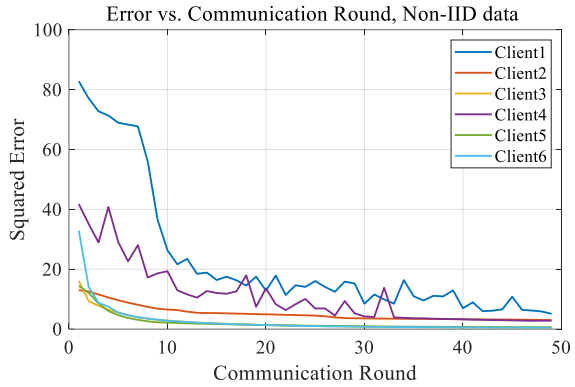

Fig. 9. Training error vs. epochs for each client in the memristor based FL system for non-IID data.

Performance is measured on the global model after each communication round is completed. Fig. 10 presents the model performance for IID data where the local devices are guaranteed to receive all four categories in their local data, yet the sources of the samples may be different. The global accuracy is greater than 98% for both the baseline software and the memristor based PIM implementations. There is a negligible difference in accuracy between the memristor implementation and the software approach. In the IID case, all local devices exhibit high accuracy (96% to 98%). However, if the local devices are trained with the non-IID dataset, the performance is affected drastically in the local model due to inadequate knowledge of unknown categories.

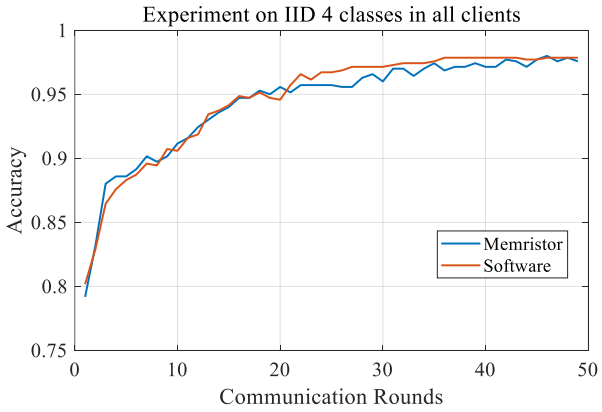

Fig. 10. Accuracy vs. communication round of the memristor based FL system when IID dataset is received by the local devices.

In Fig. 11, the advantage of the FL system over standalone edge learning of the client devices is presented to establish the importance of the FL system for distributed learning systems when the edge devices possess insufficient data to train the local IDS model. The individual IDS performance and collaborative performance are presented in Fig. 11 to show how the FL system can perform much better than any of the clients individually. Fig. 11 clearly demonstrates the advantages of collaborative learning.

Fig. 12 presents the performance of the model using the non-IID dataset. In this case, each client received two randomly selected categories of data for training. If the local models receive fewer classes for training, weight values will be biased toward learned data, and thus the model accuracy is degraded. However, testing accuracy achieved was about 77% in the memristor system, compared to 81% in the traditional software implementation. However, the accuracy of both the memristor and software implementations are better than [7,8] and similar to [6]. Furthermore, the accuracy may be increased further by implementing a deeper network in future development.

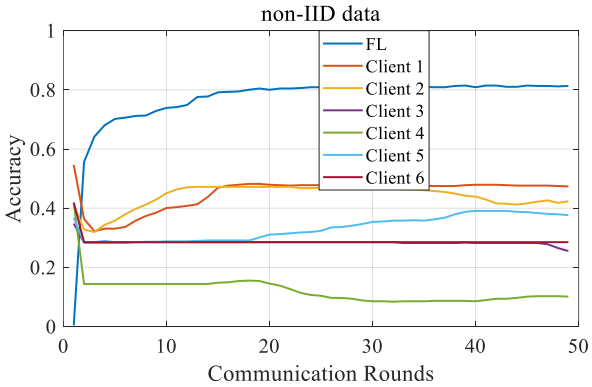

Fig. 11. Accuracy of the FL system compared with and the accuracy of each individual client for the non-IID dataset case.

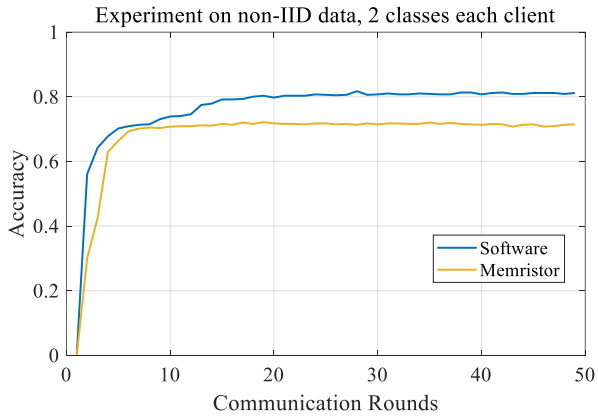

Fig. 12. Accuracy vs. communication round of the FL system for the non-IID dataset received by local devices. Each device received two randomly selected categories of data.

In the most extreme examination of this system, each client receives only one randomly selected class for training the local models. The resulting global model performance is presented in Fig. 13. Although the accuracy is comparatively lower than that of Figs. 11 and 12, the performance is still better than the results reported in [7, 8].

Finally, Fig. 14 presents the performance of the FL model when one local device receives all four of the data categories, and each of the other five clients receive only one category. The results in Figs. 13 and 14 demonstrate the benefit of FL in a distributed system of edge devices. In Fig. 13, each device received only one category, and the performance of the federated model deteriorated significantly. However, if any local device receives more data classes from which to learn, then the global performance of the model improves for all clients.

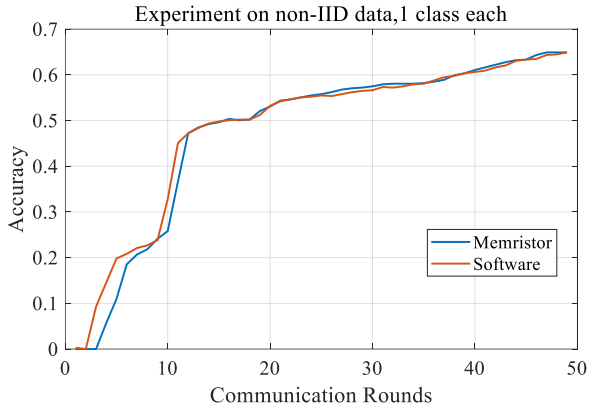

Fig. 13. Accuracy vs. communication round of the FL system for the non-IID dataset received by the local devices. Each client received only one randomly selected dataset category for training their local model.

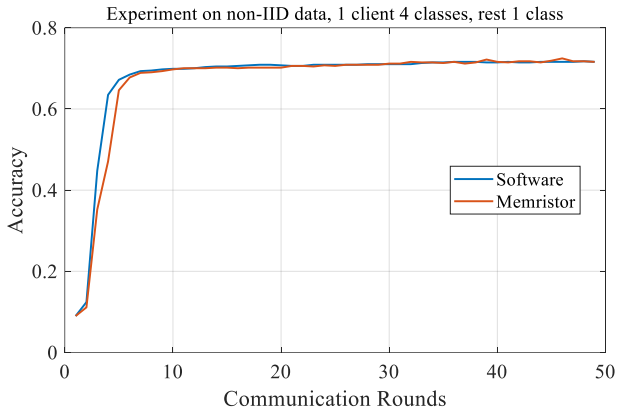

Fig. 14. Accuracy vs. communication round of FL for non-IID dataset received by the local devices. One client received all four categories for training, and the other clients each received only one category randomly selected from the dataset.

VIII. System Power, Energy, and Timing Analysis

The energy, power, and timing of the memristor crossbar and peripheral circuitry has been estimated for the FL learning in each local device (see Table I). The software version is run on an ASUS-Tinkerboard as a reference low power device for measuring the robustness of the memristor-based in-memory computing system.

TABLE I. Power, Energy, and Timing Analysis of the Memristor-Based PIM System of a Single Local Device and in Tinkerboard

Metrics	MEM-PIM	Tinkerboard	Efficiency
Area (mm^2)	0.626	NA	NA
Training Time (s)	2.12×10^{-6}	8.1×10^{-4}	382
Training Energy (J)	22.4×10^{-9}	4.05×10^{-3}	1.81×10^{5}
Training Power (W)	10.6×10^{-3}	5	471.7
Inference Time (s)	2.88×10^{-7}	3.71×10^{-4}	1288.2
Inference Energy (J)	1.08×10^{-9}	1.86×10^{-3}	1.72×10^{6}
Inference Power (W)	3.74×10^{-3}	5	1336.9

The Tinkerboard typically consumes 5W during processing. The energy is computed based on its time required to perform the training and recognition operations. The power, energy, and timing results are presented in Table I for a single local device in the case of inference on the KSL-KDD network packets. Based on the measured data and comparison, it can be determined that the memristor-based PIM system performs better than the Tinkerboard in all metrics. Table I shows the data for a single local device in the distributed systems.

IX. Conclusion

A memristor-based federated learning system is presented for collaborative learning at the edge. To demonstrate the effectiveness of this approach, a network intrusion detection dataset is learned that has the potential to secure edge computing devices with a small fraction of the resources consumed by alternative approaches. This is the first implementation of an FL algorithm in a memristor-based in-memory computing system. The training process was traditional backpropagation in the crossbar system, and the local SGD method was utilized as the optimization method. The latency of communicating between the local devices and the central server is not considered in this study. Future work may emphasize the numerical precision and latency for data transfer to the server through the communication channel during the training process. This work may be extended by studying an FL implementation in a memristor-based PIM circuit that considers reduced numerical precision.

References

[1] C. Gustavo, M. Saeteros, W. Onate, and M. V. Garcia. "Fog computing at industrial level, architecture, latency, energy, and security: A review." *Heliyon* 6, no. 4 (2020): e03706.

[2] D. P. Vinchurkar, and A Reshamwala. "A review of intrusion detection system using neural network and machine learning." *J. Eng. Sci. Innov. Technol* 1 (2012): 54-63.

[3] R. Samrin, and D. Vasumathi. "Review on anomaly based network intrusion detection system." In *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pp. 141-147. IEEE, 2017.

[4] J. Konecny, , H. B. McMahan, X. Y. Felix, P. Richtárik, A. T. Suresh, and D. Bacon. "Federated learning: Strategies for improving communication efficiency." *arXiv preprint arXiv:1610.05492* (2016).

[5] Q. Yang, Y. Liu, T. Chen, and Y. Tong. "Federated machine learning: Concept and applications." *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, no. 2 (2019): 1-19.

[6] E. M. Campos, F. S. Pablo, A. G. Vidal Vidal, J. L. H. Ramos, J. B. Bernabe, G. Baldini, and A. Skarmeta. "Evaluating Federated Learning for intrusion detection in Internet of Things: Review and challenges." *Computer Networks* (2021): 108661.

[7] Y. Qin, and M. Kondo. "Federated Learning-Based Network Intrusion Detection with a Feature Selection Approach." In *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pp. 1-6. IEEE, 2021.

[8] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad. "Internet of things intrusion detection: Centralized, on-device, or federated learning." *IEEE Network* 34, no. 6 (2020): 310-317.

[9] A. Shahid, J. M. Zain, M. F. Zolkipli, Z. Inayat, S. Khan, B. Anthony, and V Chang. "From intrusion detection to an intrusion response system: fundamentals, requirements, and future directions." *Algorithms* 10, no. 2 (2017): 39.

[10] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini. "A survey on federated learning for resource-constrained IoT devices." *IEEE Internet of Things Journal* 9, no. 1 (2021): 1-24.

[11] N. Skatchkovsky, H. Jang, and O. Simeone. "Federated neuromorphic learning of spiking neural networks for low-power edge intelligence." In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8524-8528. IEEE, 2020.

[12] K. Stewart, and Y. Gu. "One-Shot Federated Learning with Neuromorphic Processors." *arXiv preprint arXiv:2011.01813* (2020).

[13] D. S. Johnson, W. Lorenz, M. Taenzer, S. Mimilakis, S. Grollmisch, J. Abeßer, and H. Lukashevich. "Desed-fl and urban-fl: Federated learning datasets for sound event detection." *arXiv preprint arXiv:2102.08833* (2021).

[14] Y. Venkatesha, Y. Kim, L. Tassiulas, and P. Panda. "Federated Learning with Spiking Neural Networks." *arXiv preprint arXiv:2106.06579* (2021).

[15] N. Skatchkovsky, H. Jang, and O. Simeone. "Spiking neural networks—Part III: Neuromorphic communications." *IEEE Communications Letters* 25, no. 6 (2021): 1746-1750.

[16] Z. Xu, L. Li, and W. Zou. "Exploring federated learning on battery-powered devices." In *Proceedings of the ACM Turing Celebration Conference-China*, pp. 1-6. 2019.

[17] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank. "A survey of neuromorphic computing and neural networks in hardware." *arXiv preprint arXiv:1705.06963* (2017).

[18] D. Ielmini, and H. S. P. Wong. "In-memory computing with resistive switching devices." *Nature Electronics* 1, no. 6 (2018): 333-343.

[19] S. Bavikadi, P. R. Sutradhar, K. N. Khasawneh, A. Ganguly, and S. M. P. Dinakarrao, 2020, September. A review of in-memory computing architectures for machine learning applications. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI* (pp. 89-94).

[20] Burr, Geoffrey W. "A role for analogue memory in AI hardware." *Nature Machine Intelligence* 1, no. 1 (2019): 10-11.

[21] C. Yakopcic and T. M. Taha, "Analysis and Design of Memristor Crossbar Based Neuromorphic Intrusion Detection Hardware," IEEE/INNS International Joint Conference on Neural Networks (IJCNN), pp. 1-7, Rio de Janeiro, Brazil, July, 2018

[22] R. Hasan and T. M. Taha, "Enabling Back Propagation Training of Memristor Crossbar Neuromorphic Processors", International Joint Conference on Neural Networks (*IJCNN*), Beijing, July 2014

[23] O. Krestinskaya, K. N. Salama, and A. P. James. "Learning in memristive neural network architectures using analog backpropagation circuits." *IEEE Transactions on Circuits and Systems I: Regular Papers* 66, no. 2 (2018): 719-732.

[24] M .S. Alam, B. R. Fernando, Y. Jaoudi, C. Yakopcic, R. Hasan, T. M. Taha, and G. Subramanyam. "Memristor Based Autoencoder for Unsupervised Real-Time Network Intrusion and Anomaly Detection." In *Proceedings of the International Conference on Neuromorphic Systems*, pp. 1-8. 2019.

[25] L. Huang, J. Diao, H. Nie, W. Wang, Z. Li,Q. Li, and H. Liu, 2021. Memristor Based Binary Convolutional Neural Network Architecture With Configurable Neurons. *Frontiers in neuroscience*, *15*, p.328.

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas. "Communication-efficient learning of deep networks from decentralized data." In *Artificial intelligence and statistics*, pp. 1273-1282. PMLR, 2017.

[27] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini. "Federated learning for resource-constrained iot devices: Panoramas and state-of-the-art." *arXiv preprint arXiv:2002.10610* (2020).

[28] NSL-KDD data: https://www.unb.ca/cic/datasets/nsl.html

[29] C. Yakopcic, T. M. Taha, D. J. Mountain, T. Salter, M. J. Marinella, M. McLean, "Memristor Model Optimization Based on Parameter Extraction from Device Characterization Data," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 5, pp. 1804-1095, May, 2020.