RAFeL- Robust and Data-Aware Federated Learning-inspired Malware Detection in Internet-of-Things (IoT) Networks

Sanket Shukla
George Mason University
FairFax, Virginia, USA
sshukla4@gmu.edu

Gaurav Kolhe
University of California Davis
Davis , California, USA
gskolhe@ucdavis.edu

Setareh Rafatirad
University of California Davis
Davis , California, USA
srafatirad@ucdavis.edu

Houman Homayoun
University of California Davis
Davis , California, USA
hhomayoun@ucdavis.edu

Sai Manoj PD
George Mason University
FairFax, Virginia, USA
spudukot@gmu.edu

ABSTRACT

Federated Learning (FL) is a decentralized machine learning in which the training data is distributed on the Internet-of-Things (IoT) devices and learns a shared global model by aggregating local updates. However, the training data can be poisoned and manipulated by malicious adversaries, contaminating locally computed updates. To prevent this, detecting malicious IoT devices is very important. Since the local updates are large because of the high volume of data, minimizing the communication overhead is also necessary. This paper proposes a "RAFeL" framework, comprising of two techniques to tackle the above issues, (1) a robust defense technique and (2) a "Performance-aware bit-wise encoding" technique. "Robust and Active Protection with Intelligent Defense (RAPID)" is a defense system that detects malicious IoT devices and restricts the participation of the contaminated local updates computed by these malicious devices. To minimize communication cost, "Performance-aware bit-wise encoding" selects the appropriate encoding scheme for individual split bits based on their significance and effect on FL performance. The results illustrate that the proposed framework shows a 1.2-1.8 × higher compression and has an average accuracy drop of 3% to 10%.

CCS CONCEPTS

• **Security and privacy**;

KEYWORDS

Malware Detection, Federated Learning, IoT

ACM Reference Format:
Sanket Shukla, Gaurav Kolhe, Setareh Rafatirad, Houman Homayoun, and Sai Manoj PD. 2022. RAFeL- Robust and Data-Aware Federated Learning-inspired Malware Detection in Internet-of-Things (IoT) Networks. In *Proceedings of the Great Lakes Symposium on VLSI 2022 (GLSVLSI '22), June 6–8, 2022, Irvine, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3526241.3530378

GLSVLSI '22, June 6–8, 2022, Irvine, CA, USA.

ACM ISBN 978-1-4503-9322-5/22/06.
https://doi.org/10.1145/3526241.3530378

1 INTRODUCTION

With the increasing number of IoT devices, the diversity in the data experienced by the individual devices also increases due to user interactions. Such diversity makes it challenging to deploy data analysis techniques, including machine learning (ML) models for automated tasks, including security challenges [6, 19], as data experienced by individual devices differ in terms of samples and distribution. Deploying one global ML model for all devices for data analysis leads to under-fitting due to the diversity in the experienced data. Similarly, deploying individual ML models for each device for analyzing the captured data results in over-fitting. Therefore, need to secure IoT devices against unseen traces and signatures of malware, calls for a distributed ML technique that can efficiently capture node-level variations yet is aware of patterns experienced by other devices in the network.

The authors in [7] propose Federated Learning (FL) as an alternative technique for efficient and distributed ML. Federated learning is collaborative machine learning in which the server generates a shared global model with the help of a federation of participating devices. The FL framework functions differently from the traditional distributed machine learning [8, 16] in terms of handling a large number of participating devices, communication of local updates, convergence to unified model, and non-uniform distributed data distribution. In FL, all the training data is kept on local devices, thus, disassociating the capability to learn from the need to store and expose the data from the participating devices. The devices are used as nodes performing computation on their local training data to update a global model.

Despite FL being effective in terms of preserving privacy [7], it poses two important challenges. Firstly, FL requires data to be transmitted frequently to and from individual IoT devices, the communication overhead is prohibitively high [1, 14], often leading to energy-depletion of IoT devices. We did a case study demonstrating the energy consumption for various ML models with respect to each round while performing federated learning. A round in FL is a process of obtaining local updates from the participating devices, updating the global model at the FL server, and sending back the updated global model to the devices. Figure 1 illustrates the energy consumption trend, which is almost linear in the case of all the three classifiers. For IoT devices, this energy consumption is considerably high. Thus, it is important to minimize the excess energy consumption by IoT devices. Secondly, despite FL being

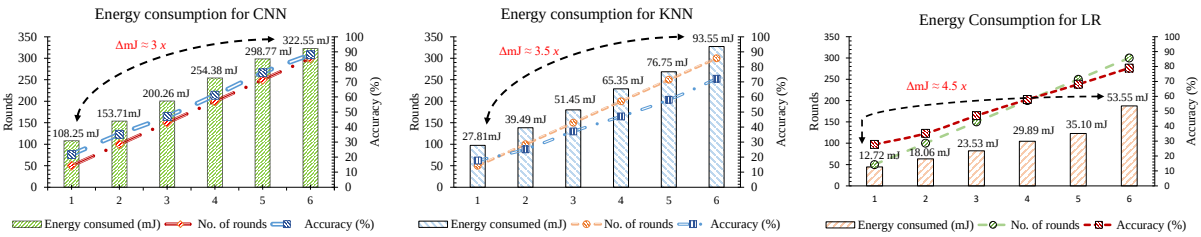

Figure 1: Amount of energy to transfer data (model parameters)

resilient to random device drop-outs, [7] or garbage data sent to the FL server during the update phase, an adversary who crafts and manipulates the training data on IoT devices can severely degrade the quality of the shared global model produced by the FL server [17, 18]. Hence, it is crucial to devise a robust technique on the FL server to withstand adversarial data manipulations by malicious IoT devices that further influence's the shared global model.

Two strategies to minimize the communication overheads, given that FL is resilient to random or truncation errors, are: a) data quantization [5]; and b) data encoding [5]. Data quantization refers to reducing the precision of the data. Though prevalent and efficient, this technique cannot reduce the precision beyond a specific limit to ensure good performance. Lossy transformations (pruning, SVD) [5] encode data by reducing the amount of information in the data, and lossless encoding (Huffman, Z-RLC, ZVC) [5] encodes data by increasing the density of information in its representation. However, these techniques have been considered part of an encoding pipeline rather than a stand-alone method.

The authors in [11] assume the existence of a global model is generated from a trusted dataset and classify the training data as malicious or benign based on its effect on the accuracy of this model. However, a trusted dataset or golden model is impractical due to inefficacy or inability to verify every device. On the other hand, the defense against data manipulation attacks [11] needs access to complete training datasets, thereby compromising user privacy and security. In [9] authors propose to identify malicious training data based on the label of their neighboring data values. This method requires access to the entire training dataset. Hence, the existing defenses are insufficient and cannot be re-purposed straightforwardly to prevent data manipulation attacks in FL systems.

Therefore, securing IoT devices against malware and making FL communication efficient and robust against malware-based data manipulation attacks is of a major concern. This paper presents a "RAFeL" framework which addresses the above mentioned issues by introducing two novel techniques; (1) **Performance-aware bit-wise encoding** - To reduce communication overheads in FL, we deploy a lightweight lossless data-aware encoding technique that encodes data with most appropriate lossless encoding technique according to the bit sparsity and density, (2) **RAPID: Robust and Active Protection with Intelligent Defense** - To mitigate the data manipulations and tampering by malicious adversaries, we integrate the RAPID defense mechanism, adding robustness by filtering out and restricting the participation of malicious IoT devices in FL and generating an error-free global model.

The results demonstrate that the proposed framework: (1) Provides a 1.2-1.8 × high compression rate compared to lossy and lossless encoding techniques, (2) Reduces the communication overhead and (3) has an average accuracy drop of 3% to 10%, thus providing robustness against data manipulation.

2 RELATED WORK

To improvise and optimize the communication overhead in FL, in [7] authors propose two ways to reduce the uplink communication costs. The Structured updates directly learn an update from a restricted space parametrized using a smaller number of variables, and (2) Sketched updates learn a full model update and then compress it using a combination of quantization and random rotations subsampling before sending it to the server. Although these methods reduce communication costs, they do not guarantee security against malicious attacks that corrupt the training data. Additionally, existing model compressions schemes such as [3] can reduce the bandwidth necessary to download the shared global model. The above-discussed method is not data-aware, which plays a critical role and can lead to efficient compression. Data manipulation or missing data in the FL can be handled by removing the outliers from the training data [13] or, in the distributed setting, from the participants' models [2], or requiring participants to share the data for centralized training [4]. All the above defenses need the defender to inspect either the training data or the resulting models (which leak the training data [10]. None of these techniques is applicable in the case of FL, where user privacy and security are the primary concern and prohibit users from explicitly sharing the raw information. This work addresses the above shortcomings by proposing "Performance-aware bit-wise encoding" and "RAPID" defense.

3 THREAT MODEL

We consider n devices in the network. Each of these devices has mutually exclusive local data . Among n devices, we consider f ($f < n/2$) devices to be compromised by the malware based data poisoning attacks, i.e., around 10-30% of the devices are deemed to be compromised. In each experiment, the attacker poisons the training data of the device (which becomes a malicious device post-attack), intending to influence the global model such that it classifies a source data label (e.g., Virus) as a target data label (e.g., Benign). The malicious device uses this poisoned training data to compute the local updates from its local model. As an alternate, one can assume that the data sent by the compromised local node to the FL server is manipulated by the malware, eventually leading to a low-performance global model.

4 PROPOSED FRAMEWORK

The overview of the proposed "RAFeL" framework is illustrated in Figure 2a. "Performance-aware bit-wise encoding" technique is outlined in Figure 2b and "RAPID" technique is shown in Figure 2c.

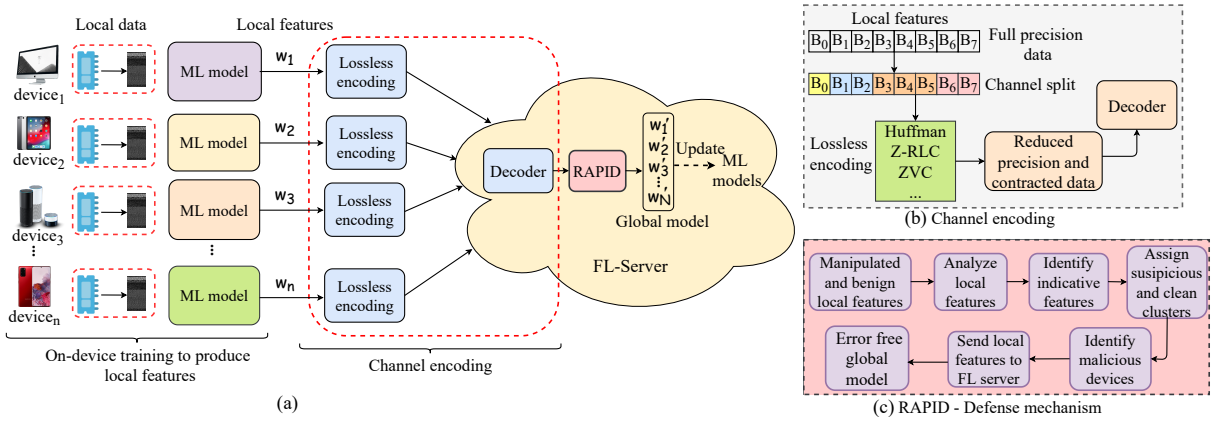

Figure 2: (a) Overview of the proposed framework, (b) Performance-aware bit-wise encoding, (c) RAPID defense mechanism

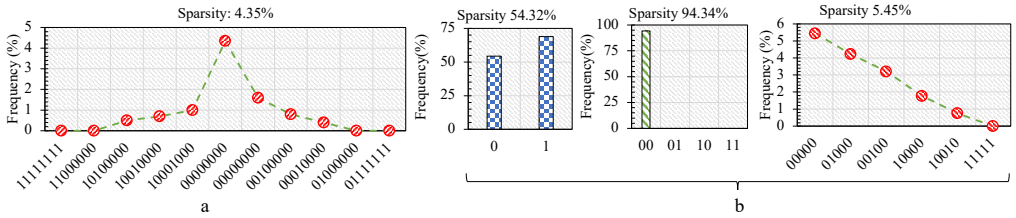

Figure 3: Changes in data characteristics by splitting bits (a) $chunk_0 = b_0\ b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7$ **, (b)** $chunk_0 = b_0$, $chunk_1 = b_1\ b_2$, $chunk_2 = b_3\ b_4\ b_5\ b_6\ b_7$

4.1 Proposed Robust FedProx Aggregation at FL Server

In this work, we use state-of-the-art Federated Prox (FedProx) [15] to perform the weight aggregation of the ML models local updates at the FL server. FedProx algorithm tackles heterogeneity by adding a proximal term (discussed later) in federated networks. FedProx permits devices to share variable amounts of work based on their resource availability by adding a proximal term (where ω is features, p_k is the probability, μ is tolerance rate, $F_N(\omega)$ is a local function). Instead of just minimizing the local function $F_N(\omega)(\cdot)$, device N uses its local solver of choice to approximately minimize the objective function h_N.

A proximal term effectively limits the impact of variable local updates. The advantages of adding proximal terms are: (1) It addresses the issue of statistical heterogeneity by restricting the local updates to be closer to the initial (global) model without any need to manually set the number of local epochs; and (2) it incorporates variable amounts of local work resulting from systems heterogeneity.

As per the threat model (discussed in section 3), the malicious samples will continue to execute and bypass the on-device malware detection and classification process. Execution of this malicious sample will manipulate the value of locally computed updates transmitted to the FL server. The malicious IoT devices under the influence of the data manipulation and poisoning attacks will show a similar distribution pattern of its locally computed updates. In contrast, the IoT devices that are not influenced by any malicious attack will show a similar distribution pattern of their locally computed updates. The distribution pattern of the locally computed updates by malicious IoT devices is represented by indicative features j. Moreover, when the IoT device is under any malicious attack, $S_I(j)$ (indicates features manipulation) is set to 1 and $\delta(j) \nsim \delta(j)$ (where $\delta \rightarrow$ distribution of training data, $\delta \rightarrow$ distribution of local model updates) is within statistical error bounds (Line 2 - Line 7) in Algorithm 1. Thus, locally computed features from most honest devices will exhibit a similar distribution in an attack scenario, while malicious devices will exhibit an anomalous distribution.

In response we modify state-of-the-art FedProx [15] algorithm by integrating RAPID (Algorithm 1) to suffice our needs. When the server receives the local features, then if the distribution of training data is similar to the distribution of local updates, we execute FedProx aggregation. Suppose the local updates sent to the server are not similar in distribution. In that case, it is a sign of being manipulated by an adversary, which is handled by executing the RAPID defense mechanism. In this way, the modified FedProx contributes to generating an error-free global model.

Initially, the defense mechanism identifies the indicative features by analyzing the distribution of the local updates using cosine similarity. The key challenge is to extract the indicative features that change distribution when the device is under a malicious attack. To decide whether a particular local feature is indicative or not, it groups the values for that local update uploaded by all devices into different clusters (Line 8 - Line 11) in Algorithm 1. We calculate the euclidean distance of n data-points (p_z, q_z refer to data-points), referred as D_E. If the distance between the centers of two clusters is greater than a threshold value α, the local feature is marked as indicative (Line 12 - Line 13) in Algorithm 1. The value of α depends on the original distribution δ of the training dataset.

The cluster which has the majority participants is marked as clean or non-malicious (benign) C_{clean} because the number of malicious samples encountered by devices are comparatively less as compared to benign samples (e.g., out of 10K samples, only 2-5% samples will be malicious). The benign samples will exhibit a similar distribution of local features by performing on-device training.

Algorithm 1 Pseudo-code for 'RAPID' - defense mechanism against device/node under malicious attack

1: **Input**: N (number of devices), f (fraction of total devices to be malicious), δ (distribution of training data), δ (distribution of local model features), τ (error-rate tolerance), α (distance threshold), $S_I(j)$ (indicates features manipulation)
2: **for** x = 0, ..., N: **do**
3: **if** $\delta(j) \sim \delta(j)$ (j = indicative features)
4: $S_I(j) \rightarrow 0$ **do**
5: Run the federated averaging algorithm
6: **else** $(\delta(j) \nsim \delta(j))$
7: $S_I(j) \rightarrow 1$ **do**
8: $D_E = \sum_{z=1}^{n}(p_z - q_z)^{1/2}$ (Calculate Euclidean distance)
9: C_{clean} and $C_{suspicious} \Rightarrow$ (Clustering $\delta(i), \delta(i)$)
10: Analyse the δ for benign and malicious users
11: Form clusters by grouping the values from δ
12: **if** D_E $(C_k{:}C_{(k-1)}) > \alpha$ **do** $C_k \Rightarrow$ Clusters
13: $\delta = j$ (indicative feature)
14: **if** $U_x \in C_{suspicious} > \tau$ **do**
15: $U_x \Rightarrow M_U$ (malicious)
16: **else**
17: $U_x \Rightarrow B_U$ (benign)
18: **end if**
19: **end if**
20: **end for**

In contrast, the other cluster is marked as suspicious $C_{suspicious}$ (Line 14 - Line 17) in Algorithm 1. All the devices $U_0, \cdots, U_N$ in the suspicious group are suspected to be malicious M_U but are not confirmed. In contrast, the devices which are not in the suspicious group are marked as benign B_U. The defense mechanism marks a device in the suspicious group as malicious if it appears for more than τ times (on finishing all rounds) such as $(U_0, \cdots, U_N \Rightarrow M_U$ (if $\rightarrow$ occurrence $> \tau$ = 40%). The RAPID sends the local updates, excluding the manipulated local features for federated aggregation. Finally, the FL server generates an error-free and accurate global model after executing the FedProx algorithm discussed below.

Hence, using the modified FedProx enhances our framework's robustness. Our next objective is to minimize the communication costs using "Performance-aware bit-wise encoding".

4.2 Performance-aware bit-wise encoding

"Performance-aware bit-wise encoding" is different from a traditional data encoding technique. It splits the full precision data into different chunks of data as shown in (Figure 2b) and applies a specialized data encoding technique for each chunk. The motivation behind using "Performance-aware bit-wise encoding" is different portions of values (weights, activations, gradients) have different characteristics in machine learning and deep learning. The most significant bits (MSB) tend to be very sparse when represented in bits, while sparsity keeps reducing as the bits become less significant. The main redundancy in machine learning data is sparsity in both values and bits [12]. The sparsity in values is the proportion of zero-valued symbols in a dataset. The sparsity in bits is the proportion of zero-bits in a signed magnitude binary representation. Machine learning data sources have high value-level sparsity.

Even dense data has high bit-level sparsity because most non-zero machine learning data values are minimal regardless of the source, as shown in Table 1. The change in data characteristics by splitting bits into chunks is visualized in Figure 3. The value sparsity of the original machine learning data is 4.35% (Figure 3a). Since the information content and sparsity are distributed unevenly across the bits, we can break values into chunks with more distinct characteristics. Therefore, we take advantage and split values into multiple chunks and apply a customized encoding technique to each chunk. Hence, the most significant chunk's sparsity increases considerably (Figure 3b), and the entropy of the source is shifted to the least significant chunk.

The decoder is deployed and used to decode the encoded data sent to the FL server. The decoder replicates the encoder but in reverse order. The input compressed symbols are stored in an 8-bit buffer. A stop code detector checks for stop code sequences in the encoded symbols. Each chunk then inspects the remaining symbol sequentially to deduce which part of the symbol was generated by what chunk in the encoder. Once all chunks have inspected the data, the symbol width can be computed, and the appropriate number of bits is removed from the buffer. Each chunk's symbol is decompressed and passed to the recombining logic that forms the complete output symbol. The decoded data will undergo the robust FedProx aggregation discussed above. The experimental results showcase that"Performance-aware bit-wise encoding" has a significant contribution in reducing the communication cost.

5 EXPERIMENTAL RESULTS

5.1 Experimental setup

The proposed framework is executed on a system running Ubuntu 18.04 with Linux 5.3 kernel. We prefer non-IID sampling for the dataset and to impose statistical heterogeneity, the data is distributed unevenly among devices created using virtual machines. We evaluated for 200 rounds because it gave better results than other # rounds, and the model did not converge beyond 200 rounds. The FL parameters vary as follows: clients per round (10, 25, 50) and # epochs (20, 30, 40).

5.2 Performance Analysis

Table 1: ML data characteristics after applying Performance-aware bit-wise encoding technique

ML Model	Bits per value	Value sparsity	Bit sparsity	Entropy density	Accuracy
CNN	11.8	34.96%	76.88%	55.23%	96.45%
Mobile-Net	10.7	37.42%	78.21%	59.43%	89.54%
RF	9.6	40.42%	80.21%	61.77%	91.43%
LR	9.2	42.78%	83.81%	64.92%	94.98%
KNN	10.2	37.93%	74.64%	56.43%	91.12%
MLP	10.9	38.13%	74.77%	59.43%	92.45%

5.2.1 ***Performance-aware bit-wise encoding.*** Table 1 illustrates data statistics after "Performance-aware bit-wise encoding" being applied on the quantised data. However, the value sparsity and bit sparsity ranges from 34% to 42% and 74% to 84%, respectively. Moreover, the model's performance, i.e., the classification accuracy, is also significant enough. Thus, splitting bits into chunks and applying a customized encoding algorithm does not impact the model's performance.

Figure 4 shows compression rates normalized to Shannon limit. Among all the ML models used, the architecture of CNN is most dense in values. Still, the distribution of non-zero values is concentrated around zero, leading to a high geometric mean compression rate. The scale on the y-axis shows compression rate normalized to

Shannon limit, where '0' means no compression and '1' means maximum compression rate given by Shannon limit. The effective compression rate enables us to compare gains and losses from compression more directly. It can be observed that the "Performance-aware bit-wise encoding" technique achieves 80% to 85% of the gain suggested by the Shannon limit from encoding. The proposed method delivers approximately 1.5 × better compression rate than Huffman encoding, 1.2 × more compression than Run Length Encoding, and 1.6-1.8 × higher compression than L-Ziv, LZMA, and Z-lib. These results demonstrate that combining lossy and lossless encoding techniques to encode 32-bit floating-point data can achieve better compression results than solely using lossy or lossless compression techniques. This reduces the communication overheads by nearly 18% to 22% which minimize the power consumption by 25-30%.

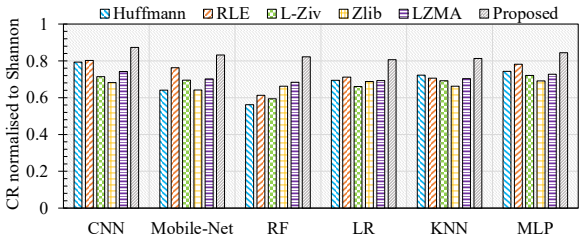

Figure 4: A comparison of compression schemes. Results are mean value of combination of all chunk splits

5.2.2 ***RAPID - Defense Mechanism.*** To measure the severity of data manipulation attacks on the proposed RAPID defense, we perform several experiments by varying the fraction of malicious devices f. The malicious devices range from 10% to 40% of all the participating devices (n) considering $f < n/2$ (n is 100, 200, 300 and 400). In each experiment, the malicious device manipulates its training data to influence the global model.

Table 2: Effect of malicious devices on model performance post RAPID defense integration

Fraction of malicious devices (%)	0	10	20	30	40
Accuracy % (CNN)	96.45	93.55	90.66	87.76	86.80
Accuracy % (Mobile-Net)	89.54	86.94	84.34	81.75	81.39
Accuracy % (RF)	91.43	88.96	86.12	83.47	82.97
Accuracy % (LR)	94.98	92.22	89.47	86.71	85.76
Accuracy % (KNN)	91.12	88.75	85.92	83.37	82.55
Accuracy % (MLP)	92.45	90.23	87.36	84.68	82.70

Table 2 shows the model's accuracy post RAPID integration when the fraction of devices are malicious. The accuracy drop for the ML classifiers used is approximately 3% when 10% of the devices are malicious. The maximum accuracy drop is nearly 10%, with 40% devices being malicious. This shows that the accuracy drop increases with an increase in the fraction of malicious users. Therefore, integrating RAPID defense yields a negligible drop in the model's accuracy, with even 40% devices being malicious.

Table 3: Post synthesis hardware results for proposed framework with different ML classifiers (@100MHz)

Classifier	Power (mW)	Energy (mJ)	Area (mm^2)
CNN	82.45	5.12	4.5
Mobile-Net	72.64	2.35	2.25
RF	45.63	2.79	1.81
LR	46.54	2.29	1.55
KNN	56.46	3.21	1.46
MLP	54.55	3.37	1.27

5.3 ASIC Implementation of Proposed Technique for different Classifiers

The experiments are implemented on a Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit, 28 nm SoC running at 1.5 GHz. The power, area, and energy values are reported at 100 MHz. We used Design Compiler Graphical by Synopsys to obtain the area for the models. Power consumption is obtained using Synopsys Primetime PX. The post-layout area, power, and energy are summarized in Table 3. Among all the classifiers, CNN consumes the highest power, energy, and area on-chip (Table 3).

6 CONCLUSION

In this paper, we integrated "Performance-aware bit-wise encoding" and "RAPID" defense techniques in Federated Learning. The experiments and results show that the implemented techniques exhibit effective and lightweight lossless encoding and robustness against data manipulation attacks in machine learning. We demonstrated that, on average, "Performance-aware bit-wise encoding" outperformed all lightweight and some complex encoding methods applied to machine learning data. The case studies provide evidence that the proposed Federated Learning-inspired framework provides 1.2-1.8 × higher compression rates and robustness with a negligible accuracy drop of just 3% to 10%, even in the presence of more than 40% compromised devices in the network.

REFERENCES

[1] C. Adrián Martínez and et al. 2010. Malware detection based on Cloud Computing integrating Intrusion Ontology representation. In *IEEE Latin-American Conference on Communications.*

[2] Clement Fung and et al. 2018. Mitigating Sybils in Federated Learning Poisoning. *ArXiv* (2018).

[3] Song Han and et al. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv:1510.00149* (2015).

[4] Jamie Hayes and et al. 2018. Contamination attacks and mitigation in multi-party machine learning. In *Advances in neural information processing systems.*

[5] U. Jayasankar and et al. 2021. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University - Computer and Information Sciences* (2021).

[6] Robert Karam and et al. 2017. Mixed-granular architectural diversity for device security in the Internet of Things. In *2017 Asian Hardware Oriented Security and Trust Symposium (AsianHOST).*

[7] Jakub Konečnỳ and et al. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[8] Chenxin Ma and et al. 2017. Distributed optimization with arbitrary local solvers. *Optimization Methods and Software* (2017).

[9] Fabrice Muhlenbach and et al. 2004. Identifying and Handling Mislabelled Instances. *J. Intell. Inf. Syst.* (2004).

[10] Milad Nasr and et al. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. *IEEE Symposium on Security and Privacy (SP)* (2019).

[11] B. Nelson and et al. 2009. Misleading Learners: Co-opting Your Spam Filter. *Springer US* (2009).

[12] Miloć Nikolić and et al. 2018. Characterizing Sources of Ineffectual Computations in Deep Learning Networks. In *IEEE International Symposium on Workload Characterization (IISWC).*

[13] Mingda Qiao and et al. 2018. Learning Discrete Distributions from Untrusted Batches. *ArXiv* (2018).

[14] Noelle Rakotondravony and et al. 2017. Classifying malware attacks in IaaS cloud environments. *Journal of Cloud Computing* (2017).

[15] Anit Kumar Sahu and et al. 2018. Federated Optimization for Heterogeneous Networks.

[16] Ohad Shamir and et al. 2014. Communication-efficient distributed optimization using an approximate newton-type method. In *ICML.*

[17] Sanket Shukla and et al. 2021. On-device Malware Detection using Performance-Aware and Robust Collaborative Learning. In *(DAC).*

[18] Gang Wang and et al. 2014. Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers. In *23rd USENIX Conf. on Security Symposium.*

[19] Kan Xiao and et al. 2016. Hardware trojans: Lessons learned after one decade of research. *ACM Transactions on Design Automation of Electronic Systems* (2016).