CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	FPGA hardware acceleration framework for anomaly-based intrusion detection system in IoT
Authors	Ngo, Duc-Minh;Temko, Andriy;Murphy, Colin C.;Popovici, Emanuel
Publication date	2021-10-12
Original Citation	Ngo, D.-M., Temko, A., Murphy, C. C. and Popovici, E. (2021) 'FPGA hardware acceleration framework for anomaly-based intrusion detection system in IoT', 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), 2021, pp. 69-75. doi: 10.1109/FPL53798.2021.00020
Type of publication	Conference item
Link to publisher's version	10.1109/FPL53798.2021.00020
Rights	© 2021, IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Download date	2026-11-14 15:07:15
Item downloaded from	https://hdl.handle.net/10468/12935

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

FPGA Hardware Acceleration Framework for Anomaly-based Intrusion Detection System in IoT

Duc-Minh Ngo
Department of Electrical and Electronic Engineering
University College Cork,
Cork, Ireland
120220051@umail.ucc.ie

Andriy Temko
Department of Electrical and Electronic Engineering
University College Cork,
Cork, Ireland
ATemko@ucc.ie

Colin C. Murphy
Department of Electrical and Electronic Engineering
University College Cork,
Cork, Ireland
ColinMurphy@ucc.ie

Emanuel Popovici
Department of Electrical and Electronic Engineering
University College Cork,
Cork, Ireland
E.Popovici@ucc.ie

***Abstract*—This study proposes a versatile framework for real-time Internet of Things (IoT) network intrusion detection using Artificial Neural Network (ANN) on heterogeneous hardware. With the increase in the volume of exchanged data, IoT networks' security has become a crucial issue. Anomaly-based intrusion detection systems (IDS) using machine learning have recently gained increased popularity due to their generation ability to detect new attacks. However, the deployment of anomaly-based AI-assisted IDS for IoT devices is computationally expensive. In this paper, a hierarchical decision-making approach for IDS is proposed and evaluated on the new IoT-23 dataset, with improved accuracy over the software-based methods. The inference engine is implemented on the Xilinx FPGA System on a Chip (SoC) hardware platform for high performance, high accuracy attack detection (more than 99.43%). For the resulting implemented design, the processing time of the ANN model on FPGA with an xc7z020clg400 device is 6.6 times and 40.5 times faster than GPU Quadro M2000 and CPU E5-2640 2.60GHz, respectively.**

***Index Terms*—Security, FPGA, IoT-23 Dataset, Neural Networks, Anomaly Detection**

I. Introduction

In recent years, the fourth industrial revolution (or Industry 4.0) has significantly impacted IoT networks' developments. IoT found many applications, including manufacturing, medical, financial, smart agriculture, automotive applications, just to name a few. As a result, the number of devices interconnected through IoT networks grew significantly. According to Statista [1], the number of IoT devices is predicted to rise to 38.6 billion in 2025 and 50 billion in 2030, respectively. Privacy and security issues have become one of the most critical concerns for IoT technology's scalability and widespread use.

A state-of-the-art solution for IoT security is an anomaly-based intrusion detection system (IDS). Compared to traditional IDS, anomaly-based IDS are more memory efficient since the former classify infected packets by comparing them with known patterns [2] (stored in an extensive database). In contrast, the latter can produce the same result by using lightweight mathematical models. Additionally, hardware accelerators have become a potential upgrade for IoT devices in both edge and cloud-based environments [3, 4, 5]. Machine Learning (ML) techniques have widely been used to implement various intelligent applications in IoT devices such as speech recognition [6], encryption [7], biosignal monitoring [8], etc. These real-time applications require not only high-performance but also low power consumption hardware to operate sustainably.

This paper proposes an efficient anomaly-based IDS framework for IoT gateways and edge devices. More specifically, the training phase is implemented on a GPU to generate neural network models which are then tested with a hierarchical decision-making approach. The inference phase is accelerated by the SoC Zynq platform, in which the Programmable Logic (PL) path contains the selected model. Furthermore, the experiment aims at the new IoT-23 dataset [9]. This is the first framework to examine the IoT-23 dataset using high-efficiency neural network architecture on FPGA to classify raw packets to the best of the authors' knowledge.

II. Methodology

A. Neural Networks

Neural Networks [10] (NN) are computer modelling systems that retrieve complex information by imitating the human brain's functionality. Fig. 1 illustrates a system overview using NN models on the new IoT-23 dataset. In the training phase, the labelled data are used to choose the suitable model architectures and train model parameters (weight and bias values). In the inference stage, the trained model receives input features and produces output values.

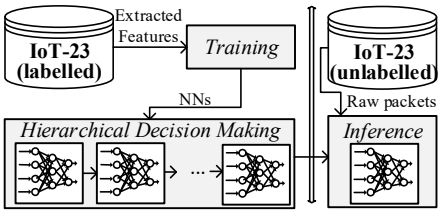


Fig. 1. System overview in training, hierarchical decision-making, and inference phases using ANN on the new IoT-23 dataset.

The hierarchical decision-making approach [11] is widely used in many areas to reduce problem complexity. This paper uses multiple NN models to improve accuracy and solve data

unbalancing issue. Besides the mentioned advantages, this method does have trade-offs by introducing overheads such as increased computational latency and power consumption.

B. Dataset

The IoT-23 dataset [9] contains network traffic from IoT devices to offer a large dataset, including labelled malware-infected and benign flows. Table I shows the total of 325,307,990 labelled records belonging to sixteen categories. Each record comprises eighteen features belonging to 2 types, which are static and flow-based features. Static features can be extracted from the network header fields of a packet. There are five features of this type in the IoT-23 dataset: IP source, IP destination, source port, destination port, and protocol. Flow-based features are calculated from packets in the same flow (or in a time window).

TABLE I
THE LABELLED FLOW IN THE IOT-23 DATASET, DIVIDED INTO FIVE GROUPS

Group	Label	Number of records
1	C&C-Mirai	2
	PartOfAHorizontalPortScan-Attack	5
	C&C-HeartBeat-FileDowbload	11
	FileDownload	18
	C&C-Torii	30
	C&C-FileDownload	53
	C&C-HeartBeat-Attack	834
	C&C-PartOfAHorizontalPortScan	888
	Attack	9,398
	C&C	21,995
	C&C-HeartBeat	33,673
2	DDoS	19,538,713
3	Benign	30,858,735
4	Okiru-Attack	13,609,470
	Okiru	47,381,241
5	PartOfAHorizontalPortScan	213,852,924

These labels are separated into five groups in Table I. The first group has attack types with less than 35,000 records each and mainly includes *C&C* infected flows. The second and third groups contain only *DDoS* and *Benign* records, respectively, while the last two groups hold *Okiru* and *PartOfAHorizontal-PortScan* (PHPS) marked records.

C. Related Work

Anomaly-based detection mechanisms have been used to detect unseen attacks [12, 13]. Several survey papers [14, 15, 16] have analyzed prominent techniques, including data mining, ML, and DL, to protect IoT networks from security vulnerabilities. The literature [17, 18] has proposed a federated self-learning anomaly detection in IoT networks based on the generated dataset. Similarly, researchers in [19] have applied ML algorithms on a generated dataset for IoT anomaly detection and achieved an accuracy of 98%

The IoT-23 dataset [9] has recently been tested with supervised learning methods. For instance, authors in [20, 21] have evaluated an ensemble method to combine Deep Neural Networks (DNN) and Long Short-Term Memory (LSTM) for improving accuracy. Additional research [22] has also combined IoT-23 with two other datasets to perform anomaly prediction. They have proposed a universal feature set to detect botnet attacks. Nonetheless, the feature extraction process in this system is tool-based, which might cause dependency and bottle-neck related issues [23]. Malware and botnet analysis on the IoT-23 dataset has been adopted in [24] and [25], respectively. Random Forest (RF) has also been selected in [26] as the best algorithm for detecting anomalies.

The authors in [27] have mentioned SoC as a future platform for securing IoT devices. Accelerating semantic caching on FPGA has been used in [28] and [29] for IoT security monitoring. The work in [30] compares ASIC and FPGA for IoT security based on studies from the CAESAR competition [31]. ANN for intrusion detection systems is proposed on an SoC platform in [32]. A comparison with the developed FPGA-based system is presented in Section III-C.

III. PROPOSED FRAMEWORK

There are four phases in the framework: data preparation, training, offline inference and online inference. The word "Offline" indicates the static learning process when administrators work with stored data, while "Online" means the action of processing stream data in a real-time context.

A. Data preparation

Our proposed framework uses a supervised learning neural network approach to classify network attacks. Fig. 2 shows that there are three main modules in the data preparation phase, which are:

- *The classification* module: scans the labelled data, then collects records that have the same label to a new file. The sixteen created files correspond to the benign label and fifteen attacks in the IoT-23 dataset. Then the result is divided into file groups, as illustrated in Table I.
- *The scaling module*: transforms extracted features of each record in the alphanumeric form to float values (between 0 and 1) using min-max normalization.
- *The shuffle* module: selects and rearranges records according to different training plans. In addition, data balancing can be applied in this module for better data distribution.

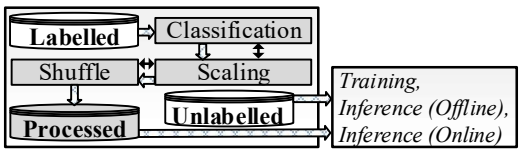


Fig. 2. The data flow modelling of the Data Preparation phase.

The processed data are stored separately and later can be used in other phases.

B. Training

The GPU-based training phase is described in Fig. 3. *The feature selection* module selects a custom input feature set

TABLE II
INFERENCE RESULTS (IN %) OF THE THREE SELECTED PLANS ON THE IOT-23 DATASET

Plan	Indexed group	Input feature set	Output label	Accuracy	Precision	Recall	F1-score	MbA
1	3 and 5	All features	Benign and PHPS	99.75	99.98	99.73	99.86	0.00
2	All 5 groups	All features	Benign and 4 group labels	99.55	99.93	99.57	99.75	0.02
			Benign and 15 attack labels[1]	**99.68**	**99.99**	**99.65**	**99.82**	**0.01**
3	All 5 groups	Static features	Benign and 4 group-attack labels	99.50	99.75	99.69	99.72	0.10
			Benign and 15 attack labels[2]	**99.50**	**99.90**	**99.55**	**99.72**	**0.03**

[1]The best model that covers all 16 types of attacks. [2]The selected model for testing on FPGA hardware.

from each processed record for training. *GPU-based Trainer* module follows the Tensorpack [33] structure for training NN models. *NN models* module contains trained models for use in the inference phase.

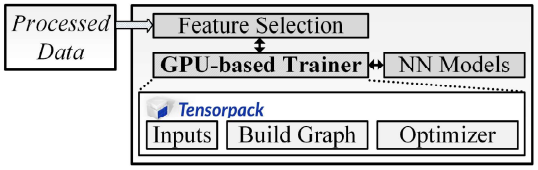


Fig. 3. GPU-based acceleration for training NN models.

The hidden layer is configured with 40 neurons, each followed by a hardware efficient ReLU (Rectified Linear Unit) activation function; then, the softmax activation function is used in the output layer. The processed data are distributed at the rate of 50% for training, 25% for validation, and 25% for testing/inference. The NN models were trained using the cross-entropy loss function and Adam optimizer with a batch size of 256, and the learning rate is 0.01. The average training and validation time for 1 epoch is 1,183s and 516s on the Quadro M2000 GPU, respectively.

We have created five NN models from three different training plans, as shown in Table II. The first two plans use all the provided features for testing the software-based implementations, while the third plan generates NN models from static features for deploying into FPGA SoC. We define the *MbA* metric for measuring the percentage of the number of *Misclassified between Attack* records. According to the inference results, the first training plan recorded the highest accuracy of 99.75%; however, this model can only classify benign and PHPS. The model[1] is considered the best model, which achieves 99.68% and 99.82% accuracy and F1-score, respectively, with only 0.01% of MbA. The third plan produces models for implementing on FPGA. These models for FPGA have an accuracy of 99.50%, which is close to the model in the second plan.

C. Offline Inference

This paper has combined two models: the model[1] in Table II (denoted as Model 1) and the first plan's model (Model 2). While Model 1 is used for global detection, Model 2 is better at classifying between Benign and PHPS records. Fig. 4 illustrates the flow-chart of our hierarchical decision-making for testing NN models.

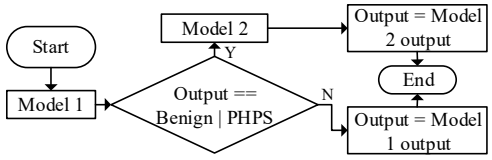


Fig. 4. Hierarchical decision-making flow-chart

We have compared both our single NN and multiple NNs solutions to related works in Table III. The authors in [20] have achieved 99.70% accuracy by using DNN. They have improved this result to 99.99% by applying LSTM in [21]; however, F1-score is not mentioned in this research. Authors in [22] and [26] have reported RF as the best algorithm for detecting attacks with 100% and 99.50% accuracy, respectively. Authors in [25] have achieved accuracy with only 0.12% greater than our single NN model. However, their experiments have not applied to the full IoT-23 dataset [22, 25]. By using the hierarchical decision-making approach, the accuracy is close to that in [25], where the authors reported 99.80%. The F1-score from this method is 99.89%, which is the highest result among the related works.

TABLE III
RESULT COMPARISON TO RELATED WORKS IN THE IOT-23 DATASET

System	Model	Accuracy (%)	F1-score (%)
[20]	DNN	99.70	87.00
[21]	LSTM	99.99	N/A
[22]	RF	100.00	100.00
[25]	NN	99.80	99.85
[26]	RF	99.50	99.00
Our system	NN	99.68	99.82
	Multiple NNs	**99.79**	**99.89**

IV. HARDWARE IMPLEMENTATION

The online inference phase is implemented on the Zynq-7000 SoC architecture, as shown in Fig. 5. The Processing System (PS) has basic components for a computer system: Multiplexer Input/Output (MIO), Memory (MEM), Interconnects and an Application Processor Unit (APU). The system is booted from an SD card, and the application is loaded into the APU. Here, network packets will be sent to the APU for extracting static features in the *Feature Extraction* module. Then, the extracted records are grouped together

in the *Buffer Initialization* module. Processed results from the FPGA are delivered back to the *Monitoring* module for analysis. Programmable Logic (PL) or FPGA for accelerating NN has six main blocks, which are

- *PS Reset* block: configures the control signals from the PS to reset other blocks in the PL system.
- *The Interconnect* block: assigns user-defined values to registers in PL through General-Purpose Ports (GP0), which is configured as a 32-bit AXI-lite interface.
- *AXI Memory Interconnect 0 (AMI0)* block: receives grouped records which are stored in MEM, using High-Performance Ports (HP0). Then, data are sent to the *DMA* block by a Memory-Mapped to Stream (MM2S) channel.
- *AXI Memory Interconnect 1 (AMI1)* block: is similar to the AMI0 block, except that data are transferred in the opposite direction, from *DMA* to this block by a Stream to Memory-Mapped (S2MM) channel.
- *DMA* block: connects directly to the *NN* block using an AXI Stream interface for sending/receiving data to/from the *NN* block.
- *NN* block: represents the NN model implemented on FPGA for hardware acceleration. At this stage, only one model is run at a time. The hierarchical decision-making method on FPGA is a promising research opportunity that needs further investigation and should be included in future works.

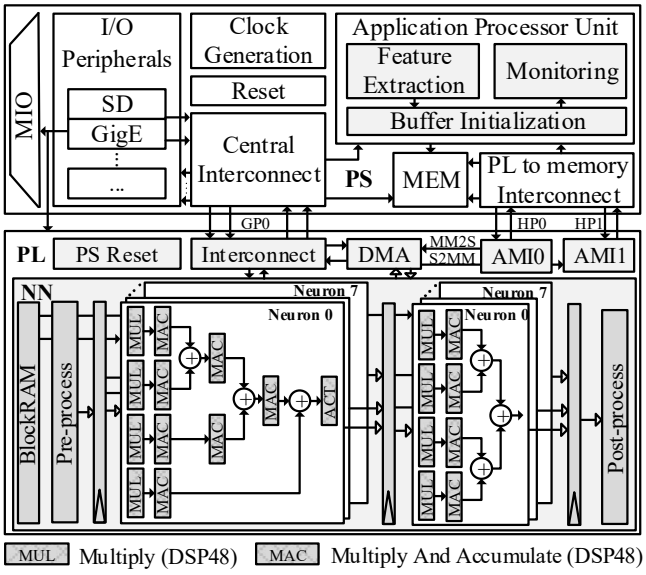


Fig. 5. Neural network inference acceleration on Zynq-7000 SoC architecture.

In the NN block, the trained parameters are loaded into BlockRAM and the input for neurons is extracted from the input stream by the *pre-process* module. The hidden and output layers are designed using pipeline strategy and resource sharing technique in which there are 8 neurons in each layer. Additionally, DSP48 blocks are used to implement both MUL and MAC operands. Finally, the *post-process* module produces the output stream and controls the *t_last* signal. For boosting the performance of the NN on FPGA, two additional techniques have been implemented, which are:

- *Hyper-parameter Quantization*: transforms each trained parameter of an NN model to integer values. For instance, for a layer with weight values in the range of $[-w, w)$, a quantized value $w_{quantized}$ from w_i, is calculated by

$$w_{quantized} = \lfloor 2^n \frac{w_i}{w} \rfloor \quad (1)$$

  where n is the number of bits for storing a weight value in the FPGA. The second model of the third plan mentioned in Table II was selected for testing on FPGA hardware. Table IV compares the results between the software-based (1) and FPGA (2) implementations of this mode, where testing used $n = 20$. The accuracy and F1-score of the model on FPGA hardware are decreased slightly by 0.07% and 0.03%, respectively, while the MbA score remained the same at 0.03%.
- *HLS Pragmas* are applied for optimizing our implementation in C++ into the RTL. For instance, *HLS Dataflow* is used for pipelining calculations between layers, *HLS Pipeline* and *HLS Unroll* for parallel computation of neurons in each layer.

TABLE IV
RESULTS (IN %) OF SOFTWARE-BASED AND FPGA IMPLEMENTATIONS

	Accuracy	Precision	Recall	F1-score	MbA
(1)	99.50	99.90	99.55	99.72	0.03
(2)	**99.43**	**99.53**	**99.84**	**99.69**	**0.03**

V. RESULTS

Table V illustrates the hardware resource usage of the FPGA implementation on a PYNQ-Z2 board using an xc7z020clg400 device and Vivado HLS version 2018.2. DSP blocks are the most used on-chip resource (83.64%); other on-chip resources are below 43.57% each. In addition, the maximum frequency of the design is 104MHz, with power consumption at 1.88W.

A. FPGA Performance

The PL configuration time is 340ms on average, including the time for loading NN parameters into BlockRAM. The line graph in Fig. 6 shows the processing time and the estimated records per second *(rps)* that the proposed system can handle. The horizontal and the left-vertical axes present the number of records per test and the processing time, respectively. The right vertical axis shows the system's estimated performance in a *million records per second*. By doubling the number of records sent each time, the performance is likely to be doubled due to small fluctuations in processing time between 0.407ms and 0.446ms. The estimated performance is 2.3 million records per second at 2^0 record sent, and this number increases exponentially to 2,407.3 and 38,312.4 million records per second by sending 2^{10} and 2^{14} records correspondingly. The DMA buffer size greatly affected the performance as the

TABLE V
THE IMPLEMENTED RESULTS OF FPGA HARDWARE ON VIVADO 2018.2

Hardware Resources Usage			
Resources	**Utilization**	**Available**	**Utilization (%)**
LUT	11,603	53,200	21.81
LUTRAM	242	17,400	1.39
FF	16,461	106,400	15.47
BRAM	61	140	43.57
DSP	184	220	83.64
Timing and Power			
F_max	**PL Static power**	**Dynamic power**	**Total power**
104MHZ	0.15W	1.73W	1.88W

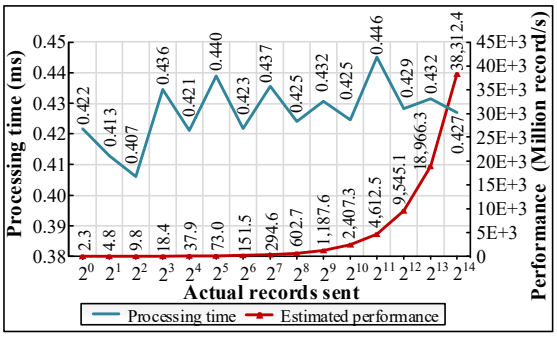


Fig. 6. The relation between the DMA buffer size and the performance of the PynQ-Z2 SoC board

amount of overhead in transferring data between PS and PL is greater than the PL processing time.

The average inference time after 10 trials are shown in Table VI. To ensure a fair comparison with the work in [32], the number of records for sending in DMA buffer size was made equal to the number in these works (22,544 records).

TABLE VI
COMPARISON IN INFERENCE TIME ON DIFFERENT PLATFORMS AND BETWEEN RELATED WORKS

System	Platform	Time (ms)
[34]	CPU i3 2.4 GHz	240.14
[32]	ARM A9 667MHz	1,458.10
	FPGA accelerator 76MHz	9.02
Current Work	Intel Xeon CPU E5-2640 2.60GHz	17.83
	Quadro M2000 with 4G of memory	2.94
	FPGA accelerator 100MHz	**0.44**

The authors in [32] and [34] used the same NSL-KDD dataset for experimenting with their NN models, which are configured as 29-21-2 and 6-7-1, respectively. Our FPGA implementation is 20.5 times faster than the same platform in [32] and 545.7 times faster than the CPU implementation in [34]. This test was also conducted on Intel Xeon CPU E5-2640 2.60GHz and GPU Quadro M2000 with 4G of memory; the NN on FPGA hardware outperformed both CPU and GPU with 40.5 times and 6.6 times faster processing time, respectively.

B. Network Throughput

FPGA inference performance increases when using greater DMA buffer sizes. However, one should not use large DMA buffer sizes because a real-time IDS needs to provide a prediction as soon as it has received network packets. Here, a 381.4MB pcap file with 398,000 packets was selected for this experiment. The processing time of the PS and PL are recorded in Table VII. The network throughput (TP) with buffer sizes from 1 to 32 records in each transaction is also calculated.

TABLE VII
NETWORK THROUGHPUT IN PACKETS PER SECOND (PKTS/S) WITH THE CORRESPONDING PROCESSING TIME OF THE PS AND PL

Record	**1**	**2**	**4**	**8**	**16**	**32**
PS (s)	2,376.1	2,375.3	2,374.5	2,367.0	2,368.1	2,363.2
PL (s)	153.2	77.9	39.3	19.8	10.1	5.0
TP (pkts/s)	157	162	164	166	167	168

From Table VII, the network throughput depends greatly on the PS processing time, where the feature extraction module is placed. This time is between 2,363.2s and 2,376.1s for extracting a pcap file with 398,000 packets in which each packet has an average size of 958.3 bytes. If each packet is extracted and sent immediately to the FPGA, the throughput is approximately 157 pkts/s. Finally, the PL's processing time, in this case, is 153.2s, which contributed to 6.1% of the system's total processing time.

VI. CONCLUSIONS AND FUTURE WORK

This paper presents a comprehensive framework for a heterogeneous hardware implementation of anomaly detection in IoT networks. The framework includes dataset pre-processing, GPU-based training acceleration, offline inference using hierarchical decision-making, and the FPGA-based inference system with high-performance, real-time intrusion detection. The proposed algorithms are tested with the new IoT-23 dataset using NN models and achieved the highest accuracy of 99.68%. The accuracy of detection is increased further to 99.79% by applying the hierarchical decision-making method. The inference phase is implemented on an FPGA SoC with the xc7z020clg400 device, which is 6.6 times and 40.5 times faster than the Intel Xeon CPU E5-2640 2.60GHz and the GPU Quadro M2000 in processing time, respectively. Future work will consider simplifying the feature extraction process into vectors [35] for combining it with a Convolutional Neural Network (CNN) on FPGA hardware. Furthermore, FPGA network specialized hardware (e.g., NetFPGA-SUME) will also be explored for studying high-performance intrusion detection systems.

ACKNOWLEDGMENT

This research is supported in part by a grant from Science Foundation Ireland INSIGHT Centre for Data Analytics (Grant number 12/RC/2289-P2) which is co-funded under the European Regional Development Fund.

References

[1] Statista, *Number of internet of things (IoT) connected devices worldwide in 2018, 2025 and 2030*, https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/ visited on Mar. 27, 2021.

[2] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[3] I. Tudosa, F. Picariello, E. Balestrieri, L. De Vito, and F. Lamonaca, "Hardware security in IoT era: The role of measurements and instrumentation," in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4. 0&IoT)*. IEEE, 2019, pp. 285–290.

[4] Microsoft, *Deploy ML models to field-programmable gate arrays (FPGAs) with Azure Machine Learning*, https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-fpga-web-service visited on Mar. 27, 2021.

[5] Q. Liang, P. Shenoy, and D. Irwin, "AI on the edge: Rethinking ai-based IoT applications using specialized edge architectures," *arXiv preprint arXiv:2003.12488*, 2020.

[6] C. Gao, S. Braun, I. Kiselev, J. Anumula, T. Delbruck, and S. Liu, "Real-time speech recognition for IoT purpose using a delta recurrent neural network accelerator," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[7] S. Maitra, D. Richards, A. Abdelgawad, and K. Yelamarthi, "Performance evaluation of IoT encryption algorithms: Memory, timing, and energy," in *2019 IEEE Sensors Applications Symposium (SAS)*, 2019, pp. 1–6.

[8] C. P. Antonopoulos and N. S. Voros, "A data compression hardware accelerator enabling long-term biosignal monitoring based on ultra-low power IoT platforms," *Electronics*, vol. 6, no. 3, p. 54, 2017.

[9] Stratosphere Laboratory, *A labeled dataset with malicious and benign IoT network traffic*, https://www.stratosphereips.org/datasets-IoT23 visited on Mar. 27, 2021.

[10] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[11] S. P. Sethi and Q. Zhang, *Hierarchical decision making in stochastic manufacturing systems*. Springer Science & Business Media, 2012.

[12] R. Bhatia, S. Benno, J. Esteban, T. Lakshman, and J. Grogan, "Unsupervised machine learning for network-centric anomaly detection in IoT," in *Proceedings of the 3rd ACM CoNEXT workshop on Big Data, Machine Learning and Artificial Intelligence for Data Communication Networks*, 2019, pp. 42–48.

[13] M. Fahim and A. Sillitti, "Anomaly detection, analysis and prediction techniques in IoT environment: A systematic literature review," *IEEE Access*, vol. 7, pp. 81 664–81 681, 2019.

[14] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (IoT) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[15] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[16] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 686–728, 2018.

[17] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for IoT," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[18] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based IoT intrusion detection system," in *NDSS Workshop on Decentralized IoT Systems and Security*, 2020.

[19] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "Mqttset, a new dataset for machine learning techniques on mqtt," *Sensors*, vol. 20, no. 22, p. 6578, 2020.

[20] V. Dutta, M. Choraś, M. Pawlicki, and R. Kozik, "A deep learning ensemble for network anomaly and cyber-attack detection," *Sensors*, vol. 20, no. 16, p. 4583, 2020.

[21] V. Dutta, M. l Choras, M. Pawlicki, and R. l Kozik, "Detection of cyberattacks traces in iot data," *Journal of Universal Computer Science*, vol. 26, no. 11, pp. 1422–1434, 2020.

[22] F. Hussain, S. G. Abbas, U. U. Fayyaz, G. A. Shah, A. Toqeer, and A. Ali, "Towards a universal features set for IoT botnet attacks detection," *arXiv preprint arXiv:2012.00463*, 2020.

[23] D. Storcheus, A. Rostamizadeh, and S. Kumar, "A survey of modern questions and challenges in feature extraction," in *Feature Extraction: Modern Questions and Challenges*. PMLR, 2015, pp. 1–18.

[24] N.-A. Stoian, "Machine learning for anomaly detection in IoT networks: Malware analysis on the IoT-23 data set," B.S. thesis, University of Twente, 2020.

[25] M. Hegde, G. Kepnang, M. Al Mazroei, J. S. Chavis, and L. Watkins, "Identification of botnet activity in IoT network traffic using machine learning," in *2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*. IEEE, 2020, pp. 21–27.

[26] R. Thamaraiselvi and S. A. S. Mary, "Attack and anomaly detection in iot networks using machine learning," 2020.

[27] S. Kumar, S. Sahoo, A. Mahapatra, A. K. Swain, and

K. K. Mahapatra, "Security enhancements to system on chip devices for IoT perception layer," in *2017 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*. IEEE, 2017, pp. 151–156.

[28] L. d'Orazio and J. Lallet, "Semantic caching framework: An FPGA-based application for iot security monitoring," *Open Journal of Internet Of Things (OJIoT)*, vol. 4, no. 1, pp. 150–157, 2018.

[29] J. Lallet, E. Casseau, L. d'Orazio *et al.*, "Mascara (modular semantic caching framework) towards FPGA acceleration for iot security monitoring," in *International Workshop on Very Large Internet of Things (VLIoT 2020)*, 2020.

[30] N. Samir, A. S. Hussein, M. Khaled, A. N. El-Zeiny, M. Osama, H. Yassin, A. Abdelbaky, O. Mahmoud, A. Shawky, and H. Mostafa, "Asic and FPGA comparative study for IoT lightweight hardware security algorithms," *Journal of Circuits, Systems and Computers*, vol. 28, no. 12, 2019.

[31] CAESAR, *Cryptographic competitions*, https://competitions.cr.yp.to/caesar.html visited on Feb. 15, 2021.

[32] L. Ioannou and S. A. Fahmy, "Network intrusion detection using neural networks on FPGA SoCs," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2019, pp. 232–238.

[33] Y. Wu, *Tensorpack Documentation*, https://tensorpack.readthedocs.io/en/latest/ visited on Mar. 27, 2021.

[34] M. Idhammad, K. Afdel, and M. Belouch, "Dos detection method based on artificial neural networks," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 4, pp. 465–471, 2017.

[35] E. L. Goodman, C. Zimmerman, and C. Hudson, "Packet2vec: Utilizing word2vec for feature extraction in packet data," *arXiv preprint arXiv:2004.14477*, 2020.