Evaluation of lightweight machine learning-based NIDS techniques for industrial IoT

Baron, A.; Le Jeune, L.; Hellemans, W.; Rabbani, M.M.; Mentens, N.; Andreoni, M.

Evaluation of Lightweight Machine Learning-Based NIDS Techniques for Industrial IoT

Alex Baron[1,2], Laurens Le Jeune[2,3], Wouter Hellemans[2(✉)], Md Masoom Rabbani[2], and Nele Mentens[2,4]

[1] Department of Mathematics, University of Padova, Padua, Italy
[2] ES&S-COSIC, KU Leuven, Leuven, Belgium
{alex.baron,laurens.jeune,wouter.hellemans, md.masoom.rabbani,nele.mentens}@kuleuven.be
[3] EAVISE-PSI, KU Leuven, Leuven, Belgium
[4] LIACS, Leiden University, Leiden, The Netherlands

Abstract. Internet of Things (IoT) devices have revolutionized communication, transportation, healthcare, and many other fields. In particular, the adoption of these devices has propelled the growth of Industry 4.0 to an exponential pace. However, while this vast pool of interconnected devices broadens the opportunities for better business and better lives, it also attracts the attention of cybercriminals. Nevertheless, it has been shown that the resource-constrained nature of these devices inhibits the deployment of traditional security measures.

To this end, we investigate how various lightweight Machine Learning-based intrusion detection systems (IDSs) can be implemented on resource-constrained IoT devices. Specifically, we train various decision tree and neural network-based models and implement them on Raspberry Pi and Field-Programmable Gate Array (FPGA) platforms. Furthermore, we evaluate our implementations on the IoT-23 and TON_IoT datasets and compare the results in terms of classification performance, throughput and resource consumption. We show that tree-based models surpass the neural network-based models in classification performance and throughput but that hardware acceleration on FPGA can aid in closing the gap in terms of throughput. As such, this work opens the path for the deployment of a real-time distributed IDS on low-cost devices.

Keywords: Intrusion Detection System · IIoT · Machine Learning · Embedded platforms · FPGA

This work is partially supported by the COllective Research NETworking (CORNET) project "TrustedIOT: Trusted Computing Architectures for IoT Devices". The Belgian partners are funded by VLAIO under grant number HBC.2021.0895. This work is also partially supported by Cybersecurity Initiative Flanders (VR20192203). Additionally, W.H. is a SB PhD fellow at FWO (Research Foundation Flanders) under grant agreement 1SH3824N. This paper is based on the work of [4].

M. Andreoni (Ed.): ACNS 2024 Workshops, LNCS 14586, pp. 246–264, 2024.
https://doi.org/10.1007/978-3-031-61486-6_15

1 Introduction

The relentless advancement of modern informatization has led Internet of Things (IoT) to play a remarkable role in several aspects of our daily lives. The possibility of being able to manipulate a wide range of devices, all under the control of a single centralized device, such as a smartphone, is making this market grow exponentially over the last few years. IoT has a total potential economic impact of 3.9 trillion to 11.1 trillion dollars a year by 2025 [20]. According to Cisco, there will be 500 billion devices connected to the internet by the year 2030 [29]. These devices may range from home appliances such as thermostats and smart light bulbs to sensors in healthcare or industry.

While very innovative, IoT is also very sensitive to safety problems, as attacks on IoT networks can have devastating consequences. For example, after infecting hundreds of thousands of IoT devices, the Mirai botnet targeted the infrastructure of Dyn, which controlled a significant portion of the internet's Domain Name System (DNS) infrastructure, and brought down sites such as Twitter, Netflix and Reddit [33].

Although traditional defense techniques relying on authentication, encryption and access control may in specific situations be useful, IoT networks need another layer of protection [5]. Intrusion Detection System (IDS) can provide this protection by analyzing system events in an effort to uncover malicious activity, such as unauthorized file or resource access [1]. Recently Machine Learning (ML) techniques have seen increasing application to improve IDS detection performance [8,21].

With the advancement of technology and the growth of IoT networks, there is a continuous search for highly effective solutions in the field of information security [5]. The purpose of this paper is to present a new perspective in the field of *Edge Machine Learning* regarding the protection of IoT networks, which consists in the use of small board microcontrollers, also called *edge-devices*. Edge ML can be defined as a field of ML which aims to bring ML applications on devices that are cheap, as well as resource and power-constrained. The ultimate goal is to develop and test different lightweight ML models that can fit into resource-constrained devices to be integrated and deployed alongside the components of an IoT network with the aim of guaranteeing network-based protection.

The pipeline of this work starts with the data collection of IoT network traffic. The analyzed traffic is preprocessed after feature extraction which aims to find the best performing features. Both common IoT network traffic and different IoT networks attacks are taken into consideration in the dataset. These attacks mainly include Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS). ML models are trained to classify the network traffic as benign or malicious and they will alert the system if any malicious packet is encountered. Different kinds of classifiers are compared in this work, including *Neural Networks (NNs), Decision Trees (DTs) and Random Forests (RFs)*. ML models are installed on Raspberry Pis and Field-Programmable Gate Arrays (FPGAs), to build a network-based device-independent IDS that can guarantee a remarkable protection to the whole IoT environment.

The main contributions of this work are:

- We develop and carry out a performance assessment on a set of lightweight ML classifiers for IDS applications in IoT environments;
- We investigate the applicability of the TabNet architecture on resource-constrained devices;
- We provide a proof-of-concept implementation of all classifiers based on Raspberry Pi and/or FPGA platforms;
- We evaluate our models on state-of-the-art datasets and report relevant metrics.

This paper is organized as follows. We present different recent and performing IDSs applied to the IoT field in Sect. 2. In Sect. 3 we give an overview about IoT security and IDSs. Furthermore, we describe ML models that are used in this work. In Sect. 4, we present problems and the methodology assumed. Subsequently, the results of the performed experiments are presented with relative discussions in Sect. 5. Sect. 6 concludes the paper and proposes future works.

2 Related Work

In this section, we introduce various ML-based IDSs for IoT security.

There are many publications showcasing various techniques related to anomaly detection in IoT networks. Common techniques include various NNs [6, 10, 13], Conditional Variational Autoencoder (CVAE) [19], Recurrent Neural Network (RNN) [2], RF [15], and Extreme Learning Machine (ELM) [17]. Additionally, some authors introduce more specific approaches. Lee *et al.* [18] analyze the power consumption of sensor nodes in IPv6 to flag malicious activity. Hosseinpour *et al.* [14] base their IDS on an Artificial Immune System (AIS). Nobakht *et al.* [23] analyze device activity in a Software Define Network (SDN). Sarhan *et al.* [28] use a Hierarchical Blockchain-based Federated Learning (HBFL) framework to detect intrusions in a collaborative fashion. We provide an overview of these techniques in Table 1, additionally highlighting whenever implementations are distributed or centralized.

Related anomaly-based work shows considerable promise. However, most systems have been tested on older datasets, even though different attacks on smart and industrial networks are continuously advancing. Most recent IDS systems are centralized or cloud-based, leaving the internal network vulnerable if attackers bypass the single centralized IDS or if the connection between devices and cloud computing fails. Therefore, in this work, we explore distributed solutions.

3 Background

In this section, we briefly review the main tools and components involved in our work. We start by describing the environment security of IoT networks and the features of IDSs. Next, we describe the used ML model architectures. Finally, we introduce FINN, which is used to deploy quantized neural networks on hardware.

Table 1. Overview of relevant anomaly-based IDSs for IoT settings

References	Method	Placement
Lee *et al.* [18]	IDS over 6LowPAN	Distributed
Hosseinpour *et al.* [14]	Lightweight IDS based on AIS	Distributed
Hodo *et al.* [13]	ANN	Centralized
Nobakht *et al.* [23]	Host-based IDS in SDN	-
Lopez-Martin *et al.* [19]	CVAE	-
Kozik *et al.* [17]	IDS based on ELM	Centralized
Doshi *et al.* [6]	NN	Centralized
Ge *et al.* [10]	NN	-
Hasan *et al.* [11]	LR, SVM, DT, RF, ANN	Distributed
Almiani *et al.* [2]	Full-automated IDS based on RNN	-
Hussein *et al.* [15]	RF	-
Sarhan *et al.* [28]	HBFL	-

3.1 Problem Setting

The IoT is growing fast and is widening its fields of application. IoT systems are well characterized for the heterogeneity and diversity of the devices involved. As well as the mixture of devices deployed, several protocols are involved to make IoT networks functional and reliable. Despite the widespread use of IoT networks, they are quite different from a conventional computer network. In particular, IoT systems are constrained in terms of computational capability and complexity. Moreover, due to their heavily distributed and heterogeneous nature, a centralized traditional solution may not be always suitable [5].

It is easy to infer that due to these limitations, most IoT devices are not equipped with efficient defense mechanisms (e.g. memory isolation, address space randomization, encryption and authentication algorithms [34]). Furthermore, another serious threat, due in part to IoT networks' fast and wide proliferation, are botnets that produce DoS and DDoS attacks as explained in Sect. 1.

IDSs aim to identify malware, malicious access or any kind of attack to defend internal networks. They represent one major research problem in cybersecurity and as there are several risks concerning networks there are different systems built to secure an environment from external attacks [31].

Motivated by an increasing number of vulnerabilities, attacks, and information leaks, IoT device manufacturers as well as cloud providers and researchers are working to design security systems and protocols, to explore new vulnerabilities and to seek effective ways to protect data privacy.

3.2 Machine Learning

In this paper, we investigate the deployment of lightweight ML models on tiny and low-power devices. Specifically, we compare different kinds of NNs, Convolutional Neural Networks (CNNs), Decision Trees (DTs) as well as ensembling

techniques such as RF, AdaBoost, and Extremely Randomized Trees (ET). In addition, we explore Attentive Interpretable Tabular Learning (TabNet) and consider hardware acceleration of the CNNs using the FINN framework. For the sake of completeness, we provide background information on TabNet and FINN.

TabNet. Attentive Interpretable Tabular Learning [3], or TabNet, is a novel high-performance and interpretable canonical deep tabular data learning architecture that uses sequential attention to choose features for reasoning at each decision step, resulting in more efficient learning. TabNet outperforms various tabular learning models on various datasets for classification and regression tasks.

TabNet's architecture is divided in an *encoder* composed of different feature transformers, attentive transformers and feature masking as well as a *decoder* which is composed of feature transformers. TabNet is appropriate for an intrusion detection task with tabular network datasets, as it inputs raw data without any preprocessing. Moreover, its training via gradient descent enables large flexibility. Although the complexity introduced by the attention transformer and the feature transformer in Tabnet's architecture results in challenging deployment for very small micro-controllers, TabNet performs extremely well for medium-sized controllers.

3.3 FINN

FINN is an experimental tool designed by Xilinx Research Lab in order to implement Deep Learning model on FPGA. In particular, FINN builds a streaming architecture where each layer has its own engine and each layer can be executed as soon as the previous layer has generated the data. FINN targets Quantized Neural Networkss (QNNs) trained in PyTorch with the help of Brevitas [24] which is a Python library to create quantized models. Brevitas also comes with a set of tools to manage the quantization properties and the functionality to export the QNNs to FINN. The workflow of FINN starts once a suitable QNN has been trained, tested and exported to an Open Neural Network Exchange (ONNX) representation. FINN will transform the initial QNN into synthesizable High-Level Synthesis (HLS) layers using different transformations. In particular, FINN's pipeline starts by preparing the model to facilitate the tuning of the layers which is based on setting up the graph model correctly and removing floating point operators. The layers are then turned into HLS and grouped in a *Dataflow Partition* which contains HLS layers suitable for acceleration. Once the sythesizable model is completely ready, FINN uses Xilinx's software called Vivado and/or Vitis to generate the final HLS code, bitstream and driver used to deploy the starting model on hardware.

4 Method

In this section we present the architecture, concept, design principles and the pipeline of the conducted experiments of our work. After stating our objective,

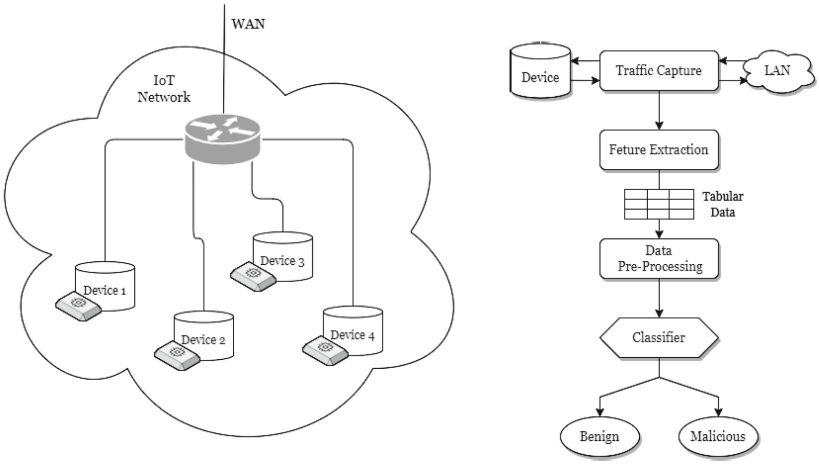


Fig. 1. Proposed network-based IDS applied architecture (a). Workflow of the proposed framework during deployment, starting from the data collection to the classification (b).

we describe the datasets, features and metrics that we use. We describe our metrics in the appendix.

We then show how we train the ML models introduced in Sect. 3 and deploy them on Raspberry Pi and FPGA.

4.1 Research Objective

The main objective of the proposed work is to build different lightweight IDSs, capable of distinguishing between benign and malicious network traffic. We integrate them in individual IoT devices in the Local Area Network (LAN), as shown in Fig. 1a. We assume that any device on the network can exchange data with all the other connected devices and the victim of an attack may be any device which belongs to the LAN. By introducing a high level of flexibility and autonomy in the IDS, we seek to introduce IDSs that can be deployed on various devices. Moreover, the IDSs should not only be accurate, but additionally, they should not influence the overall performance of the IoT devices.

4.2 Proposed Framework

We now give an overview to the computation pipeline performed by the proposed IDS that goes from the traffic capture to the binary classification between benign and malicious traffic as shown in Fig. 1b. We now describe the individual steps.

Traffic Capture. The first step of the workflow consists of real-time and continuous traffic capturing. Raw network data which includes in-going and out-going

packets can be collected and processed with different tools such as Zeek [27] and Wireshark [32]. For testing, in our experiments, we used two different datasets, TON_IoT [22] and IoT-23 [9]. They both provide a large amount of network traffic collected in either the packet capture (pcap) and Comma-Separated Values (CSV) format for TON_IoT or conn.log[1] labelled files for IoT-23.

Feature Extraction. After the network capturing we extract the relevant fields from every network packet. In particular, each field corresponds to a feature. We then aggregate packets in their respective flows and use the resulting features. Feature extraction is a key step for the final efficiency and accuracy of ML models, as they totally depend on the quality and quantity of information provided. Therefore, we try to extract generic packet features from traffic instead of focusing on the characteristics of singular attacks or the specific behavior of an infected IoT device. In particular we focus on header fields in the Internet Protocol (IP) packet, including the size of the packet or IPv4 and TCP/UDP related information. In the next subsection, we will list the features selected for each dataset and the consequent reasons.

Feature Processing. Pre-processing consists of different procedures to make acceptable and optimize the input data to a ML algorithm. Typically, the datasets are composed of different types of data (e.g. integers, floats, doubles, binary, strings, etc.). Non-numeric inputs need to be mapped to numbers or to be converted into one-hot-encoded values, after which all numeric data needs to be normalized to improve data quality and ML performance [30].

Training and Classification. Preprocessed data is then used to fit our models on the training set. Finally, after training our ML models using the preprocessed data, they can be deployed for new traffic. In our work, we have selected different ML models to test in different compositions, i.e. in different architectures. During inference, the classifier takes as input the processed data and outputs whether this network flow belongs to traffic considered malicious or not.

4.3 Datasets and Feature Importance

In this work we used two different datasets, specifically the TON_IoT (Network) Dataset [22] and the IoT-23 Dataset [9]. These are the most recent data collections with malicious and benign IoT network traffic, bringing significant information at the level of malware families that even modern security solutions are unfamiliar with.

TON_IoT comprises a collection of Industry4.0, IoT and Industrial IoT new generation data. At UNSW Canberra, the dataset were collected in a large and realistic network environment. The testbed was designed based on interacting network, IoT devices and systems. The environment is composed of three layers

[1] Obtained by running the Zeek network analyser.

to mimick the implementation of recent realistic IoT networks. Specifically, the *edge layer* involves physical devices, while the *fog layer* and the *cloud layer* determine the computation location. TON_IoT presents malicious scenarios with nine different attack categories launched against vulnerable IoT applications, Operating Systems (OSs) and network systems, as listed in Table 2. Several heterogeneous sources of data are included into the dataset, in particular sensors, OSs and network traffic. In our work we focus on the network traffic records (i.e. traffic flow), which are extracted with Zeek [27]. The TON_IoT dataset comes with the extracted traffic flow in CSV format. The authors also provide a Training and Test collection comprising a smaller portion of the dataset, which we use for our work. Table 2 gives the composition of the network traffic data.

Table 2. Statistics of Network Records of TON_IoT dataset [22]

Labels	All Network Data	Training and Testing
Backdoor	508,116	20,000
DDoS	6,165,008	20,000
DoS	3,375,328	20,000
Injection	452,659	20,000
MIMT	1,052	1,043
Password	1,718,568	20,000
Ransomware	72,805	20,000
Scanning	7,140,161	20,000
XSS	2,108,944	20,000
Normal	796,380	300,000
Total	22,339,021	461,043

The IoT-23 dataset [9] is a recent collection of network traffic from different IoT devices. Data were captured in the Stratosphere Laboratory, AIC group, FEL, CTU University in Czech Republic. The dataset comprises 23 different network captures, also called *sessions*, with 20 malware and 3 benign network traffic captures. During data collection, different malicious scenarios were executed related to a specific malware which performed different actions in a Raspberry Pi. The benign traffic however was collected using three real physical devices. In particular, a Philips HUE smart LED lamp, an Amazon Echo home personal assistant and a Somfy smart door-lock. As the devices were not simulated, the collection is characterized by the real network behaviour. We use different scenarios of IoT-23 as shown in Table 3. For each capture, IoT-23 contains a series of *pcap* files and *conn.log.labeled* files, which are the Zeek *conn.log* files obtained by running Zeek network analyser using the original pcap file. We derive a CSV file containing the chosen features with the related data starting from the *conn.log.labeled* files using the *zeek-cut* tool provided by Zeek.

Table 3. IoT-23 Dataset scenarios used in our work. FD stands for FileDownload, PortScan is the short for PartOfHorizontalPortScan and C&C is short for Command and Conquer. Specifically, the*FileDownload* label indicates that a file is being downloaded to the infected device. The *Attack* label refers to some kinds of attack which try to exploit flaws such as telnet login, brute force, command injection etc.

Labels	1-1	8-1	34-1	35-1	44-1
Benign	469,275	2,181	1,923	8,262,389	211
DDos	-	-	14,394	2,185,302	1
C&C	8	8,222	6,706	81	14
C&C-FD	-	-	-	12	11
PortScan	539,465	-	122	-	-
Attack	-	-	-	3	-
Total	1,008,748	10,403	23,145	10,447,787	237

The IoT-23 scenarios contain different kinds of attacks and consequently a considerable amount of information. From the *IoT23-35-1* scenario we extracted a subset where the number of entries has been reduced to avoid data imbalance, composed of $1,048,484$ benign samples and $895,929$ malicious samples. Furthermore, we tested the *IoT23-44-1* scenario with the aim of observing how the selected ML models can perform with extremely reduced amount of data.

From TON_IoT and IoT-23 we extract a subset of features that we use to train the ML models. In particular, we concentrate on features concerning the *connection activity* and the *statistical activity* related to the network and transport layer. Although there are multiple characteristics which can be derived from individual network packets, we have focused on the most consistent information. Specifically, during the traffic analysis it is possible that some protocols are not present in the layers, or the tool used to capture the traffic fails to gather certain non-essential characteristics. We have excluded information regarding DNS, Secure Socket Layer (SSL) as well as Hypertext Transfer Protocol (HTTP) activity, focusing more on addresses, ports, amount of bytes transmitted, amount of packets transmitted, duration of transmission and protocol used. From both TON_IoT and IoT-23 we extract the same 14 features, as shown in Table 4. We additionally use scikit-learn [26] to depict the importance of each feature in an RF in Fig. 2.

4.4 Experimental Setup

We train and validate the chosen ML models on a machine operated with 64-bit Windows 11 Home, and an Intel Core i5-10210U four core CPU having 1.60 GHz base frequency and 4.20 GHz as max turbo frequency. Afterwards, the saved trained models are transferred to Raspberry Pi and FPGA for inference on new input data.

Table 4. Dataset features used in our experiments. Note that some identical features may have different names in the datasets.

TON_IoT			IoT-23		
No	Name	Type	Name	Type	Specifics
1	ts	Time	ts	Time	Timestamp of connection
2	src_port	Number	id.orig_p	Number	TCP/UDP source port
3	dst_port	Number	id.resp_p	Number	TCP/UDP destination port
4	proto	String	proto	String	Protocol
5	service	String	service	String	DNS, HTTP, SSL, DHCP, etc.
6	duration	Number	duration	Number	Flow duration
7	src_bytes	Number	orig_bytes	Number	Source bytes
8	dst_bytes	Number	resp_bytes	Number	Destination bytes
9	conn_state	String	conn_state	String	Connection state
10	missed_bytes	Number	missed_bytes	Number	Number of missing bytes
11	src_pkts	Number	orig_pkts	Number	Number of source packets
12	src_ip_bytes	Number	orig_ip_bytes	Number	Number of source IP header bytes
13	dst_pkts	Number	resp_pkts	Number	Number of destination packets
14	dst_ip_bytes	Number	resp_ip_bytes	Number	Number of IP destination header bytes

Classifiers are implemented in Python 3.8 via several popular ML libraries, especially PyTorch [25] as well as Scikit-learn [26], which is also used to derive the performance and statistical results. Section A lists our model architectures and their corresponding hyperparameters, and additionally comments on the grid search, optimization and regularization used in our experiments.

In order to run the experiments, the trained and saved models are transferred to the Raspberry Pi running Pi OS Lite ready to perform inference. In particular, the board is a Raspberry Pi Model 3B [7] which is equipped with 1.2 GHz BCM2837 Soc ARM Cortex-A53 CPU with 4 cores, 1024 MB of RAM and a power requirement of 5 V at 2.5 A.

We also implement some models on FPGA using the FINN pipeline described in Sect. 3. At the moment, FINN is open-source and publicly available[2]. We tested the QNN Quant-NN on the PYNQ-Z2 FPGA board. The PYNQ-Z2 features 512MB DDR3 of RAM and it is equipped with a 650MHz dual-core Cortex-A9 processor.

5 Results

In this section, we carry out a detailed performance analysis and we discuss the results of the classifiers described in the previous section.

We start by evaluating the results for TON_IoT and IoT-23 by comparing tree-based and NN-based models in two separate groups. Table 5 and Table 6

[2] https://github.com/Xilinx/finn.

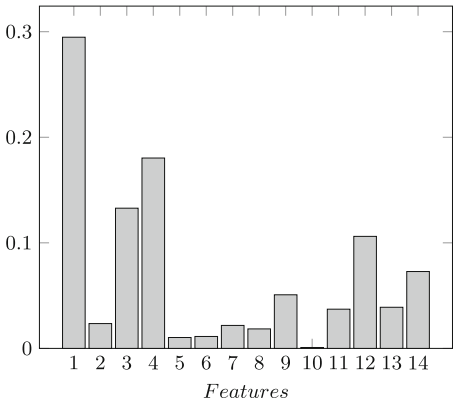


Fig. 2. This histogram depicts the Feature Importance for a Random Forest. In particular, the Feature Importance is provided by the fitted attribute related to the Random Forest, obtained with scikit-learn. In this RF model it is possible to see that the timestamp feature is very important for the correct classification of data. The horizontal axis represents the features numerically in correspondence with Table 4.

show the results obtained in terms of key metrics for the TON_IoT dataset, and similarly Table 7 and Table 8 for IoT-23. Besides detection performance metrics, we also consider the throughput, which is relevant when considering large-volume attacks such as DoS and DDoS. This measure is computed by dividing the total number of test instances by total time taken by a model to classify all the test instances. Furthermore, tree-based models were trained on the Raspberry Pi itself, as they are typically more lightweight than NNs, and their training times were measured. This value is useful for hypothesizing possible on-device-training for such models, directly on deployed devices (i.e. Raspberry Pis), with the goal of improving performance over time and user privacy, and without requiring users to update the device software.

Table 5. Results on TON_IoT Dataset - Neural Networks.

Models	*Acc*	*P*	*R*	F_1	ROC-AUC	*T*
NN small	0.9893	0.9789	0.9904	0.9846	0.9895	2,336
NN	0.9889	0.9826	0.9858	0.9842	0.9882	1,810
CNN small	0.9473	0.9530	0.8937	0.9224	0.9349	2,398
CNN	0.9815	0.9921	0.9545	0.9729	0.9752	2,088
TabNet	0.9868	0.9744	0.9882	0.9813	0.9871	743

Table 5 shows the value of all relevant metrics concerning NNs on the TON_IoT dataset. It is evident from these results that CNNs do not perform as well as the other NNs and TabNet, which achieve almost the same accuracy (98.9%, 98.2% and 98.6% respectively). However, Table 6 shows that tree-based

models perform considerably better than NNs using that dataset. We observe that the DT outperforms other classifiers in terms of execution speed and overall performance, as it obtains an almost perfect classification accuracy (99.99%) and F1-score (99.99%). Tree-based ensemble models perform very similarly, with all results hovering around 99.9%.

Table 6. Results on TON_IoT Dataset - DT & Ensamble Models.

Models	Estimators	*Acc*	*P*	*R*	F_1	ROC-AUC	*T*	Training Time [s]
DT	-	0.9999	0.9998	0.9998	0.9999	0.9998	2,311,528	15.87
RF	20	0.9998	0.9997	0.9998	0.9998	0.9998	122,221	116.81
RF	100	0.9998	0.9997	0.9998	0.9997	0.9998	18,247	573.40
AdaBoost	20	0.9989	0.9986	0.9985	0.9985	0.9988	41,893	110.25
AdaBoost	100	0.9997	0.9996	0.9995	0.9996	0.9996	7,940	561.32
ET	20	0.9998	0.9997	0.9998	0.9997	0.9998	51,809	103.66
ET	80	0.9998	0.9997	0.9998	0.9997	0.9998	19,312	396.65

Furthermore, training and classification speeds vary according to the number of estimators in the various ensembles. The DT also has the best performance for those metrics. In fact, this model is extremely suitable for classifying network traffic with the selected pre-processing. RF, Adaboost, and ET perform slightly worse however, with a noticeable difference in the sample processing speed. Regarding TON_IoT then, the DT clearly prevails, while both CNNs get the worst results. Lastly, we observe that the 1D convolution applied in the context of the TabNet does not obtain the same results as the feed-forward NNs and tree-based models, which seem to be better suited for this classification task.

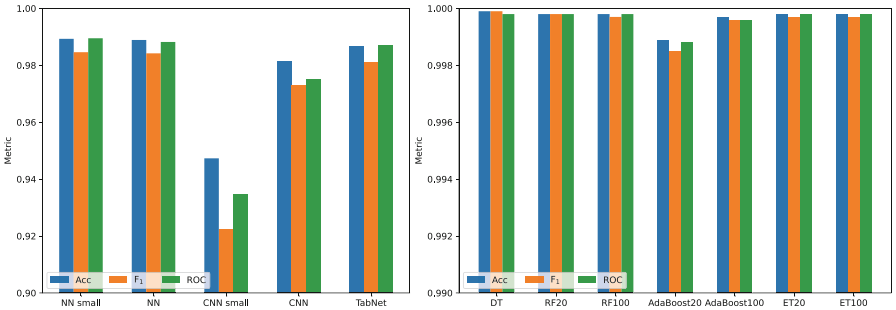
Table 7. Results on IoT-23 Dataset - Neural Networks.

Models	*Acc*	*P*	*R*	F_1	ROC-AUC	*T*
NN small	0.9977	0.9954	0.9998	0.9976	0.9977	1,568
NN	0.9984	0.9969	0.9997	0.9983	0.9984	1,250
CNN small	0.9954	0.9941	0.9963	0.9952	0.9953	1,524
CNN	0.9961	0.9955	0.9964	0.9960	0.9961	1,063
TabNet	0.9957	0.9973	0.9939	0.9956	0.9957	258

Table 8. Results on IoT-23 Dataset - DT & Ensamble Models.

Models	Estimators	*Acc*	*P*	*R*	F_1	ROC-AUC	*T*	Training Time [s]
DT	-	0.9999	0.9999	1.0	0.9999	0.9999	153,991	164.57
RF	20	0.9999	0.9999	0.9999	0.9999	0.9999	72,101	698.52
AdaBoost	20	0.9967	0.9954	0.9979	0.9966	0.9967	36,102	665.36
ET	20	0.9997	0.9994	0.9999	0.9997	0.9997	67,681	540.91

Regarding IoT-23, similar to the TON_IoT results, the largest fully connected NN performs best among the NN-based models with an accuracy of 99.84% as shown in Table 7. Table 8 presents the tree-based models, with the DT and RF as the best performing models. They obtain near-perfect results with an accuracy, precision, recall, F1-score and AUC of at least 99.99%. Our results suggest that the traffic flows in IoT-23 are more easily classified than the traffic captured by TON_IoT. Similar to TON_IoT, the smallest CNN appears to perform worst, while the DT and ensemble models again obtain the best results. Figure 3 and Fig. 4 graphically show the results obtained for both TON_IoT and IoT-23 with regards to the accuracy, F1-score and AUC metrics. These results suggest that the encoded belief over hypothesis carried by the NN characteristic is less effective in this scenario than the piece-wise constant approximation view introduced by tree-like models where decision rules are inferred from the data features.

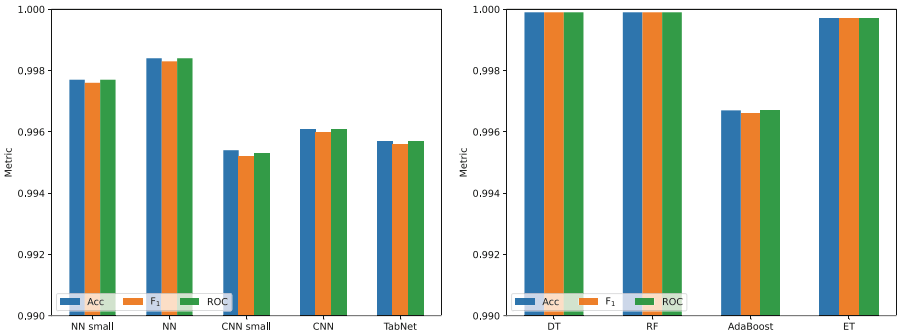


(a) Neural Networks (b) Tree models

Fig. 3. Comparison of the statistical results of the models on TON_IoT. The left chart includes neural network based models and the right chart the tree-based algorithms. The considered metrics are accuracy, f1-score and AUC-ROC measure. It is clear by the plots that the tree-based models perform much better than the others.

All results up to this point refer to the experiments carried out on the Raspberry Pi 3. Next, the results of the tests carried out by implementing a QNN in the FPGA are exhibited.

Table 9 shows the results of the QNNs trained locally and then implemented on the PYNQ-Z2 FPGA board. As can be seen from Table 10, which exhibits the percentage utilization of the physical components of the FPGA board and the measured values of latency, bandwidth, frequency, and power consumption, this implementation is characterized by extraordinarily fast processing of input data. The difference between the bandwidth measured during the simulation and the bandwidth measured directly on the device can be attributed to overhead in sample transfers. We note that the utilization of LUT, FF and BRAM is relatively low, and combined with a clock frequency of 100 MHz we achieve an extremely low latency. This translates to a generally good accuracy for an 8-bit quantized model, which is 97.47% and 99.81% for TON_IoT and IoT-23, respectively.



(a) Neural Networks (b) Tree models

Fig. 4. Comparison of the statistical results of the models on IoT-23 Dataset. The left chart includes NN-based models meanwhile the right chart shows the tree-based algorithms. The considered metrics are accuracy, F1-score and AUC-ROC measure. The charts show the better performance of tree-based models even in this scenario.

Table 9. Results of QNN first tested on PC.

Models	Accuracy	Precision	Recall	F1-score	ROC-AUC
Quant-NN (TON_IoT)	0.9747	0.9798	0.9468	0.9630	0.9682
Quant-NN (IoT-23)	0.9981	0.9964	0.9998	0.9981	0.9981

Overall, the DT seems to outperforms other classifiers for all investigated metrics, but with RF and ET showing similar results, for both TON_IoT and IoT-23. We can firmly state that as tree-based models obtain the best results, they are the best candidates for the IDS system. These results can likely be attributed to the tabular nature of the network traffic alongside the pre-processing applied to the chosen features which favors tree-based algorithms. Furthermore the throughput of these models is significantly higher than the other classifiers considered. The experimental tests demonstrate how the chosen lightweight classifiers obtain excellent result at distinguishing network traffic as benign or malicious within a wide group of different attacks, and how they can be easily integrated into resource-limited devices such as Raspberry Pis and FPGAs. Real-time efficiency of an IDS hardly depends on the dataset used in the training phase. Therefore, to ensure the best security, it is necessary to use a dataset that contains traffic patterns related to recent types of malicious attacks. The datasets we used, TON_IoT and IoT-23, are suitable choices for this purpose and the presented results demonstrate that the chosen models show promising performance.

Table 10. Results on FPGA PYNQ-Z2 using performance Metrics.

Resource	Utilization	Available	Utilization (%)
LUT	17,030	53,200	32.01%
LUTRAM	56	17,400	0.32%
FF	6,090	106,400	5.72%
BRAM	5.5	140	3.93%
T (simulation) $[samples/s]$	1,438,848		
Latency $[\mu s]$	2.56		
Frequency $[MHz]$	100		
Power est. $[W]$	0.503		
T (PYNQ) $[samples/s]$	717,529		

6 Conclusions and Future Work

In this paper, a study on anomaly-based intrusion detection systems suitable for securing IoT networks against malicious attacks is conducted. We document the performance assessment of different ML classification algorithms, including neural networks, decision tree, random forests, adaboost, extremely randomized trees and TabNet. All the classifiers are benchmarked on recent IoT datasets containing different kinds of attacks, namely TON_IoT and IOT-23. The optimal parameters of the models are obtained using a grid search algorithm combined with a study on the feature importance. The performance of all the classifiers has been measured in terms of accuracy, precision, recall and area under the receiver operating characteristic curve. Moreover, the models are evaluated from the perspective of processing speed.

The experiments are carried out on a Raspberry Pi 3B and a PYNQ-Z2 FPGA board, and the results show that the DT, RF, and ET models show the best trade-off between prominent metrics and processing time. We show how these models achieve excellent results for classifying network traffic processed as tabular data. They are therefore a suitable choice for building IoT specific IDSs. We demonstrate that the combination of Raspberry Pi or FPGA with ML classifiers is capable of achieving excellent results in terms of detection accuracy and response time. As our models are lightweight, the additional overhead on edge devices can be limited. This, alongside the high flexibility of our models, allows for hybrid and distributed deployment of additional security in a resource-constrained IoT setting. The proposed system is trustworthy since even if a centralized IDS or cloud computing fails, the internal security of individual devices is not affected. This property proves useful in industrial IoT, where the consequences of a failure to protect the individual device can have serious consequences. In addition, the versatile design allows ordinary model updates via on-device training, in order to adapt to emerging attacks which implies constant advancement of the network security status.

For future work, we plan on further investigating various models for lightweight implementation on low-powered devices. Additionally, we should explore how our system can be integrated in a real-work networking environment. For this, we plan on integrating our system alongside existing IDS solutions. Furthermore, leveraging incremental learning to extend our heuristics could facilitate more robust online intrusion detection.

A Model Architectures

- **Neural Network Models**:
  To better understand the behaviour of NNs we experiment with the difference between a small NN and a bigger NN, both for the multilayer perceptron (MLP) using only fully connected layers as well as the CNN.
  - *NN1* (small_NN): fully connected MLP composed of 3 hidden layers, with 64, 256, 64 neurons respectively. There are 14 input features and the output layer has 2 neurons (binary classification). ReLU is used as activation function.
  - *NN2*: smaller fully connected NN composed of one hidden layer of 64 neurons. Input layer has 14 neurons and the output layer 2.
  - *CNN1* (small_CNN): it is composed of 1 convolutional layer characterized by 1 input channel and 16 output channels, kernel size of 2 and stride of 1 and a fully connected layer with 96 input neurons and 2 output neurons with ReLU as activation function.
  - *CNN2*: more advanced CNN composed of 2 convolutional layers and 2 fully connected layers. The first convolutional layer has 1 input channel, 8 output channels, kernel size of 3 and stride of 1. The second convolutional layer has 8 input and 16 output channels, kernel of 3 and stride 1. The fully connect layers have 64 and 2 neurons respectively. ReLU is used as activation function.
  - *Quantized-NN*: is composed of 3 layers. The hidden layer is composed of 64 neurons and the quantization is done with 8 bits using ReLU as activation function. This model is needed for the deployment on the FPGA.
- **Decision Tree (DT)**: for the performance assessment the criterion is set to *entropy*, and the maximum depth of the tree is set to 24, according to the best parameters of the grid search.
- **Random Forest (RF)**: the parameters are 100 estimators, i.e. the number of trees included in the ensemble. The maximum depth of trees is set to 26 and the *gini* impurity is chosen as function to measure the quality of a split.
- **AdaBoost**: We use 100 estimators, with a learning rate of 0.1 and *SAMME.R* as a boosting algorithm [12]. The base estimator is a Decision Tree initialized with a maximum depth of 1.
- **Extremely Random Forest (ET)**: according to the grid search we set the number of estimators equals to 80, *gini* as a criterion and no maximum depth.

– **TabNet**: We set the width of the decision prediction layer and the attention embedding for each mask to 32, in accordance to the authors' instructions [3]. We use 4 as steps in the architecture, with 3 independent Gated Linear Units layers at each step is.

Grid Search: for Decision Trees, RF, AdaBoost, Extremely Randomized Trees we used *GridSearchCV* to find the best hyperparameters.

Tree-based models have the ability to process a large number of samples per second, so we can search for the highest performing hyper parameters (which generally increase model heaviness) even at the cost of sacrificing some classification speed in exchange for higher accuracy. As a result, we prioritize the accuracy over the bandwidth in the determination of the final hyperparameter values.

Optimization Function: on our work we used Adam (Adaptive Moment Estimation) optimizer [16] for all the NNs and for TabNet. The learning rate set for TabNet is 0.02 and 0.01 for all the NNs.

Regularization: we first test the two fully-connected NNs and the two convolutional NNs with different regularisation techniques including Dropout, L1, L2 regularization. Although regularisation is proven effective for preventing overfitting, the classification accuracy of our models did not further improve. The NNs are trained on dataset of millions of samples and they are able to generalize very well on them, this implies our model models do not suffer from overfitting.

B Metrics

We evaluate our models with conventional ML metrics. Given some classification problem, the true positives (TP) are attack samples that are classified as attack, the false positives (FP) are normal samples that are classified as attack, the true negatives (TN) are normal samples that are classified as normal and the false negatives (FN) are attack samples that are classified as normal. The Accuracy $Acc = \frac{TP+TN}{TP+TN+FP+FN}$ then is the proportion of correction classifications. The Recall $R = \frac{TP}{TP+FP}$ or R monitors how well the classifier can detect attacks. The Precision $P = \frac{TP}{TP+FP}$ or P monitors how well a model can correctly identify attacks, and $F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$ is the harmonic mean of P and R. Finally, the False Positive Rate or $FPR = \frac{FP}{FP+TN}$ monitors the number of false alarms. We also consider the Area under the Curve of the Receiver Operating Characteristics (AUC-ROC), mapping the R in function of the FPR.

Additionally, we define the throughput T of models as the number of samples they are able to process per second. Specifically for FPGA hardware, we also monitor the resource consumption by considering the Lookup Table (LUT), Flip-Flop (FF) and Block RAM (BRAM) utilization.

References

1. RFC 4949: Internet security glossary, version 2 (2007)
2. Almiani, M., AbuGhazleh, A., Al-Rahayfeh, A., Atiewi, S., Razaque, A.: Deep recurrent neural network for IoT intrusion detection system. Simul. Model. Pract. Theory **101**, 102031 (2020)
3. Arik, S.Ö., Pfister, T.: Tabnet: attentive interpretable tabular learning. CoRR **abs/1908.07442** (2019). http://arxiv.org/abs/1908.07442
4. Baron, A.: IMAT: a lightweight IoT network intrusion detection system based on machine learning techniques. Master's thesis, University of Padova (2022)
5. Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., Faruki, P.: Network intrusion detection for IoT security based on learning techniques. IEEE Commun. Surv. Tutor. **21**(3), 2671–2701 (2019)
6. Doshi, R., Apthorpe, N., Feamster, N.: Machine learning DDoS detection for consumer internet of things devices. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 29–35. IEEE (2018)
7. Raspberry Pi Foundation: Raspberry pi (2022). https://www.raspberrypi.com
8. Gamage, S., Samarabandu, J.: Deep learning methods in network intrusion detection: a survey and an objective comparison. J. Netw. Comput. Appl. **169**, 1–21 (2020). https://doi.org/10.1016/j.jnca.2020.102767
9. Garcia, S., Parmisano, A., Erquiaga, M.J.: IoT-23: a labeled dataset with malicious and benign IoT network traffic (2020). https://doi.org/10.5281/zenodo.4743746. https://www.stratosphereips.org/datasets-iot23
10. Ge, M., Fu, X., Syed, N., Baig, Z., Teo, G., Robles-Kelly, A.: Deep learning-based intrusion detection for IoT networks. In: 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), pp. 256–25609. IEEE (2019)
11. Hasan, M., Islam, M.M., Zarif, M.I.I., Hashem, M.: Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. Internet Things **7**, 100059 (2019)
12. Hastie, T., Rosset, S., Zhu, J., Zou, H.: Multi-class adaboost. Stat. Interface **2**(3), 349–360 (2009)
13. Hodo, E., et al.: Threat analysis of IoT networks using artificial neural network intrusion detection system. In: 2016 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6. IEEE (2016)
14. Hosseinpour, F., Vahdani Amoli, P., Plosila, J., Hämäläinen, T., Tenhunen, H.: An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach. Int. J. Digit. Content Technol. Appl. **10**(5) (2016)
15. Hussein, A.Y., Falcarin, P., Sadiq, A.T.: IoT intrusion detection using modified random forest based on double feature selection methods. In: Liatsis, P., Hussain, A., Mostafa, S.A., Al-Jumeily, D. (eds.) TIOTC 2021. CCIS, vol. 1548, pp. 61–78. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-97255-4_5
16. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Kozik, R., Choraś, M., Ficco, M., Palmieri, F.: A scalable distributed machine learning approach for attack detection in edge computing environments. J. Parallel Distrib. Comput. **119**, 18–26 (2018)
18. Lee, T.-H., Wen, C.-H., Chang, L.-H., Chiang, H.-S., Hsieh, M.-C.: A lightweight intrusion detection scheme based on energy consumption analysis in 6LowPAN. In: Huang, Y.-M., Chao, H.-C., Deng, D.-J., Park, J.J.J.H. (eds.) Advanced Technologies, Embedded and Multimedia for Human-centric Computing. LNEE, vol.

260, pp. 1205–1213. Springer, Dordrecht (2014). https://doi.org/10.1007/978-94-007-7262-5_137
19. Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., Lloret, J.: Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT. Sensors **17**(9), 1967 (2017)
20. Manyika, J., Chui, M., Bisson, P., Jonathan Woetzel, R.D., Bughin, J., Aharon, D.: Unlocking the potential of the internet of things (2015). https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world
21. Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E.S.: A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun. Surv. Tutor. **21**(1), 686–728 (2018). https://doi.org/10.1109/COMST.2018.2847722
22. Moustafa, N.: A new distributed architecture for evaluating AI-based security systems at the edge: network ton_iot datasets (2021)
23. Nobakht, M., Sivaraman, V., Boreli, R.: A host-based intrusion detection and mitigation framework for smart home IoT using openflow. In: 2016 11th International Conference on Availability, Reliability and Security (ARES), pp. 147–156. IEEE (2016)
24. Pappalardo, A.: Xilinx/brevitas (2021). https://doi.org/10.5281/zenodo.3333552
25. Paszke, A., et al.: Pytorch: an imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
26. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
27. Project, T.Z.: The zeek network security monitor (2020). https://zeek.org/
28. Sarhan, M., Lo, W.W., Layeghy, S., Portmann, M.: HBFL: a hierarchical blockchain-based federated learning framework for a collaborative IoT intrusion detection. arXiv preprint arXiv:2204.04254 (2022)
29. Shafique, K., Khawaja, B.A., Sabir, F., Qazi, S., Mustaqim, M.: Internet of things (IoT) for next-generation smart systems: a review of current challenges, future trends and prospects for emerging 5G-IoT scenarios. IEEE Access **8**, 23022–23040 (2020). https://doi.org/10.1109/ACCESS.2020.2970118
30. Singh, D., Singh, B.: Investigating the impact of data normalization on classification performance. Appl. Soft Comput. **97**, 105524 (2020)
31. Tsai, C.F., Hsu, Y.F., Lin, C.Y., Lin, W.Y.: Intrusion detection by machine learning: a review. Expert Syst. Appl. **36**(10), 11994–12000 (2009)
32. Wireshark: Wireshark. https://www.wireshark.org/
33. Woolf, N.: DDoS attack that disrupted internet was largest of its kind in history, experts say (2016). https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet
34. Zhou, W., Jia, Y., Peng, A., Zhang, Y., Liu, P.: The effect of IoT new features on security and privacy: new threats, existing solutions, and challenges yet to be solved. IEEE Internet Things J. **6**(2), 1606–1616 (2018)