University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Cybersecurity

IMAT: A Lightweight IoT Network Intrusion Detection System based on Machine Learning techniques

Supervisor
Prof. Mauro Conti
University of Padova

Co-supervisor
Prof. Nele Mentens
KU Leuven

Master Candidate
Alex Baron

Student ID
2021133

Academic Year
2021-2022

"The isolated man does not develop any intellectual power. It is necessary for him to be immersed in an environment of other men, whose techniques he absorbs during the first twenty years of his life. He may then perhaps do a little research of his own and make a very few discoveries which are passed on to other men. From this point of view the search for new techniques must be regarded as carried out by the human community as a whole, rather than by individuals"

— Alan Turing

Abstract

Internet of Things (IoT) is one of the fast-expanding technologies nowadays, and promises to be revolutionary for the near future. IoT systems are in fact an incredible convenience due to centralized and computerized control of any electronic device. This technology allows various physical devices, home applications, vehicles, appliances, etc., to be interconnected and exposed to the Internet. On the other hand, it entails the fundamental need to protect the network from adversarial and unwanted alterations. To prevent such threats it is necessary to appeal to Intrusion Detection Systems (IDS), which can be used in information environments to monitor identified threats or anomalies. The most recent and efficient IDS applications involve the use of Machine Learning (ML) techniques which can automatically detect and prevent malicious attacks, such as distributed denial-of-service (DDoS), which represents a recurring thread to IoT networks in the last years.
The work presented on this thesis comes with double purpose: build and test different light Machine Learning models which achieve great performance by running on resource-constrained devices; and at the same time we present a novel Network-based Intrusion Detection System based on the latter devices which can automatically detect IoT attack traffic. Our proposed system consists on deploying small low-powered devices to each component of an IoT environment where each device performs Machine Learning based Intrusion Detection at network level. In this work we describe and train different light-ML models which are tested on Raspberry Pis and FPGAs boards. The performance of such classifiers detecting benign and malicious traffic is presented and compared by response time, accuracy, precision, recall, f1-score and ROC-AUC metrics. The aim of this work is to test these machine learning models on recent datasets with the purpose of finding the most performing ones which can be used for intrusion-defense over IoT environments characterized by high flexibility, easy-installation and efficiency. The obtained results are above 0.99% of accuracy for different models and they indicate that the proposed system can bring a remarkable layer of security. We show how Machine Learning applied to small low-cost devices is an efficient and versatile combination characterized by a bright future ahead.

Contents

Abstract v

List of figures ix

List of tables xi

Listing of acronyms xiii

1 Introduction 1

2 Related Works 7
- 2.1 Anomaly-based & Specification-based . . . 7
- 2.2 Signature-based . . . 11
- 2.3 Hybrid Approaches . . . 12

3 Background 15
- 3.1 IoT Security and Intrusion Detection . . . 15
- 3.2 Machine Learning . . . 18
  - 3.2.1 Neural Networks and Convolutional Neural Networks . . . 18
  - 3.2.2 Decision Tree . . . 21
  - 3.2.3 Random Forest . . . 21
  - 3.2.4 AdaBoost . . . 22
  - 3.2.5 Extremely Randomized Trees . . . 23
  - 3.2.6 TabNet . . . 24
- 3.3 Raspberry Pi . . . 25
- 3.4 FPGA . . . 27
  - 3.4.1 FINN . . . 28

4 Proposed Approach 29
- 4.1 Problem Formulation . . . 29
- 4.2 Proposed Framework . . . 30
  - 4.2.1 Traffic Capture . . . 30
  - 4.2.2 Feature Extraction . . . 31
  - 4.2.3 Feature Processing . . . 32
  - 4.2.4 Training and Classification . . . 33
- 4.3 Datasets . . . 33
- 4.4 Feature Analysis . . . 38
- 4.5 Experimental Setup . . . 42
- 4.6 Evaluation Metrics . . . 48

5 Results 51

6 Conclusion 63

References 65

Acknowledgments 71

Listing of figures

3.1 A diagram model of an artificial neuron as proposed by McCulloch and Pitts in 1943. . . . 19
3.2 Example of 1D Convolution Neural Network. . . . 20
3.3 Random Forest's voting outcome example [1]. . . . 22
3.4 TabNet's Encoder-Decoder Architecture [2]. . . . 24
3.5 Raspberry Pi 3B board. . . . 26
3.6 PYNQ-Z2 FPGA board. . . . 28

4.1 Proposed network-based intrusion detection system applied architecture. . . . 31
4.2 Workflow of the proposed framework, starting from the data collection to the classification. 32
4.3 The plot depicts the Feature Importance for a Random Forest. In particular, Feature importances are provided by the fitted attribute related to the Random Forest, obtained with scikit-learn [3]. Feature Importances are computed as the mean and standard deviation of accumulation of the impurity decrease within each tree, which is represented by the y axis. In this model (Random Forest) it is possible to see that the timestamp feature is very important for the correct classification of data. In the x axis the features are represented numerically as they are shown in Table 4.3. . . . 39
4.4 The chart represents feature statistics for normal traffic protocol distribution on TON Dataset. The protocols are TCP, UDP and ICMP. It is possible to notice how evenly TCP and UDP packets are divided in a normal traffic condition. . . . 40
4.5 The chart represents feature statistics for malicious traffic protocol distribution on TON Dataset. It is clear from the images that the protocol represents an important feature since most of the malicious packets captured in TON Dataset are TCP packets. On the other hand, very few malicious packets are ICMP and UDP packets have reduced from 54.3% to 6.4%. . . . 40
4.6 Service distribution of benign packets. The chart depicts a comparison of feature statistics for benign traffic on TON Dataset. It can be seen that most of the service types are DNS and HTTP. . . . 41
4.7 Service feature distribution of malicious packets. The chart depicts a comparison of feature statistics for malicious attack traffic on TON Dataset. It is noticed the majority of *HTTP* and *DNS* packets and the difference in usage between normal and attack purposes. Compared with benign traffic, it is noticeable that there are more HTTP packets and fewer DNS packets. 41
4.8 Comparison of Adam to other optimization algorithms (AdaGrad, RMSProp, SGDNesterov, AdaDelta) training a multilayer perceptron [4]. . . . 45
4.9 Trained model block representation which depicts the different layers of the Quantized Neural Network used during FINN workflow. . . . 47
4.10 The ROC curve is plotted on a graph with the True Positive Rate (or Sensitivity) on the y axis and False Positive Rate (or 1 - Specificity) on the x axis. . . . 48
4.11 Confusion matrix structure. . . . 49

5.1 Comparison of the statistical results of the models on TON Dataset. The chart includes neural network based models. The considered metrics are accuracy, f1-score and AUC-ROC measure. 59
5.2 Comparison of the statistical results of the models on TON Dataset. The considered metrics are accuracy, f1-score and AUC-ROC measure. It is clear by the plot that the tree-based models perform much better than the others. 59
5.3 Comparison of the statistical results of the models on IOT-23 Dataset. The chart includes neural network based models. The considered metrics are accuracy, f1-score and AUC-ROC measure. 60
5.4 Comparison of the statistical results of the tree-based models on IOT-23 Dataset. The considered metrics are accuracy, f1-score and AUC-ROC measure. The chart show the better performance of tree-based models even in this scenario. 60

Listing of tables

2.1 Related works table. 14

3.1 Comparison between the specifications of the Raspberry Pi which was released in 2012 and Raspberry Pi 3, shown in Fig. 3.5 which it was released in 2016. 26

4.1 Statistics of Network Records of TON_IoT dataset [5]. In particular, the attacks collected are Backdoor, Distributed Denial of Service and normal DoS, Injection attacks Man-In-The-Middle, Password stealing attacks, Ransomware, Scanning and Cross-Site-Scripting (XSS). 35

4.2 IoT-23 Dataset scenarios used in our work. FD stands for FileDownload, PortScan is the short for PartOfHorizontalPortScan and HB is the short of HeartBeat. In details, *FileDownload* label indicates that a file is being downloaded to the infected device. *Attack* label refers to some kind of attacks which try to exploit flaws such as telnet login, brute force, command injection etc. 35

4.3 Features chosen, from TON and IoT-23 dataset, which have been used to train the different Machine Learning algorithms. In total, 14 features have been chosen and the table shows the type of each feature. 37

4.4 Key evaluation metrics for the experiments. 50

5.1 Results of neural networks models and TabNet for TON Dataset. 52

5.2 Results of Decision Tree and ensemble models for TON Dataset. The number of estimators is also shown as the training time. 53

5.3 Results of neural networks models and TabNet for IoT-23 Dataset. 55

5.4 Results of Decision Tree and ensemble models for IoT-23 Dataset. The number of estimators is also shown as the training time. 55

5.5 Training times and samples per second of neural networks models performed on the local machine on TON dataset. All the models are tested with the parameters presented in the previous chapter. 57

5.6 Training times and samples per second of tree-based models performed on the local machine on TON dataset. All the tree-based models are tested with the parameters exposed in the previous chapter and 20 estimators. 57

5.7 Training times and samples per second of neural networks models performed on the local machine on IoT-23 dataset. All the models are tested with the parameters presented in the previous chapter. 58

5.8 Training times and samples per second of tree-based models performed on the local machine on IoT-23 dataset. All the tree-based models are tested with the parameters exposed in the previous chapter and 20 estimators. 58

5.9 Results of the quantized neural network for TON and IoT-23 dataset. 61

5.10 Performance metrics obtained on PYNQ-Z2 for both simulation and physical test. 62

Listing of acronyms

AI ..... Artificial Intelligence
CEP ..... Complex Event Processing
CNN ..... Convolutional Neural Network
DL ..... Deep Learning
DNN ..... Deep Neural Network
DoS ..... Denial of Service
DDoS ..... Distributed Denial of Service
DT ..... Decision Tree
EPR ..... Event Processing Repository
FF ..... Flip Flop
FPGA ..... Field-Programmable Gate Array
HIDS ..... Host-based Intrusion Detection System
IC ..... Integrated Circuit
IoT ..... Internet of Things
IDS ..... Intrusion Detection System
LUT ..... LookUp Table
ML ..... Machine Learning
NIDS ..... Network Intrusion Detection System
NN ..... Neural Network
RAM ..... Random Access Memory
RF ..... Random Forest
TCP ..... Transmission Control Protocol
UDP ..... User Datagram Protocol
XSS ..... Cross Site Scripting

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 [6].

IoT is one of the emerging trends of the current decade. According to Cisco, there will be 500 billion devices connected to the internet by the year 2030 [7]. Automobile field sees the number of vehicles connected globally growing over the hundreds of millions. In the context of the IoT, *Things* refers to a wide variety of devices. It could be a simple sensor to a complex could server. Nowadays almost every electronic device can be connected and capable of exchanging data, like smart bulbs, IP cameras, thermostats, appliances, homes and many other. As well as in the high number of interconnected devices, IoT unfolds in several fields, other than homes and automobile, including healthcare, entertainments, industrial appliances, sports, etc.

Deploying IoT applications, as stated in [8], could be a challenging task due to several reasons: *(i)* the general complexity of a distributed computing system; *(ii)* the lack of general guidelines or frameworks that handle low level communication and simplify high level implementation, as the elements that make up an IoT system are hardware/devices, communication protocols, and interfaces [9]; *(iii)* several programming languages are involved,

and *(iv)* the devices are connected to each other over different communication protocols. These problems have led to a quick evolution in terms of introducing IoT programming frameworks that handle the aforementioned challenges.

IoT is an incredibly innovative environment but very sensitive to safety problems, since attacks that jeopardize the functioning of an IoT network can have devastating consequences if not stopped or prevented in time. In the next years, business-to-business applications will probably capture more value than consumer uses, implying IoT as large potential in developing economies. A dynamic industry is also evolving around IoT technology [6], and like the house environments, all of these IoT networks need the best integrity, confidentiality and availability protection considering the risks at stake.

The proliferation of insecure IoT devices has resulted in disastrous events in recent years. In October 2016, the Mirai botnet with hundreds of thousands IoT devices targeted the server of Dyn, company that controls much of the Internet's DNS infrastructure, and brought down sites like Twitter, Netflix, Reddit, CNN and many others [10]. It is then noticebale that the role of protecting the security IoT networks is fundamental and requires the best practice. Although traditional defense techniques which relate on authentication, encryption and access control may be useful in some specific situation where the attacks are known, it emerged that IoT networks need other layer of protection [11]. *Intrusion Detection Systems (IDS)* come to help and the deployment of these mechanisms is extremely significant for the protection of IoT networks. *Intrusion Detection* is a security service that monitors and analyzes system events with the purpose of finding, and providing real-time or near real-time warning of attempts to access system resources in an unauthorized manner [12]. IDSs monitor the state of software and hardware running in a network and it is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified.

Traditional *Network Intrusion Detection Systems (NIDS)* may be less efficient for IoT environments due to constrained resource, limited power and connectivity [11] [13]. Another challenging aspect is that the defender cannot expect that there will be an exact distinction between an attack and a normal use of resources by an authorized user. Rather, we should expect that there will be some sort of overlap, and this could lead to an unwanted situation. *Machine Learning (ML)* tools come to help over these situations where pure heuristic/algorithmic and/or traditional IDS are not enough. Machine Learning and Deep Learning (DL) models are a leading technology nowadays as extremely powerful tools that can fit in several fields and outperform common "old" practices. ML is one of today's most rapidly growing technical fields which has several different applications such as computer vision, speech recognition, natural language processing, robot control, etc. [14]. We will see that *Neural Networks* (NN) and tree-based ensamble models trained to accomplish intrusion detection task over network traffic are capable to reach 99% of accuracy or more in some scenarios. ML is a subfield of Artificial Intelligence (AI)

that consists of various algorithms and techniques characterized by the ability to learn and improve automatically from experience. Nowadays, everything that comes out of the world of technology has an ML component within it, and it is easy to see how this field of study is growing rapidly as it is changing the world by affecting various segments of our lives. Machine Learning is being applied to many aspects of modern society. Intelligent technologies such as IoT and cloud computing work together with ML tools to create "smart" devices to improve the coexist with intelligent humans. Another aspect of the impact of AI is in social media, where it is being applied to increasingly improve feeds and user experience. It starts by collecting user data and then feeding it to ML algorithms that, for example, try to predict the type of posts a user would like to see. Health care is an area where AI plays an important role, as it is used for faster diagnosis of patients. With the help of artificial intelligence, it is possible to predict health problems, genetic problems and prevent diseases. Artificial intelligence is also used in education to personalize the learning experience, business analysis and many other fields.

With the advancement of technology and the growing of IoT networks, there is a continuous search for highly effective solutions in the field of information security [11]. The purpose of this work is to present a new perspective in the field of Machine Learning regarding the protection of IoT networks, which consists in the use of small board micro controllers, also called *edge-devices*. The ultimate goal is to develop and test different Machine Learning light-models that can fit into small-devices to be integrated and deployed alongside the components of an IoT network with the aim of guarantee network-based protection. Edge ML can be defined as a field of Machine Learning which aims to bring ML applications on devices that are cheap, as well as resource and power-constrained. TinyML [15], instead, can be viewed as a sub-field that focuses on Ultra-Low-Power Micro-controllers. Embedded devices which can perform ML tasks represents a huge scale revolution in system security applications although they come with the tradeoffs of requiring optimization and maximization of hardware, software, data science, and machine learning.

Pipeline

The pipeline of this work starts from the data collection of network traffic of an IoT Network under the assumption that packets passing through these kind of networks are distinct from the ones coming by other internet connected devices. In particular, devices as smartphones, laptops and servers generate a different traffic with respect to IoT components that are more likely to have a repetitive and flat network flow [16]. The analyzed traffic is pre-processed and it follows a features extraction over the data with the aim of tracking the best features to lead the ML models to perform at their best. IoT common network traffic and different attacks to IoT networks are taken into consideration as dataset, mostly Denial-of-Service and Distributed-Denial-of-Service attacks, they will be explained in section IV. Machine Learning 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 kind of classfiers are compared in this work, including *Neural Networks*, *Decision Trees* and *Random Forests*. ML models are installed into Raspberry Pi's and FPGA's, with the aim of build a network-based IDS which is device-independent and can guarantee a remarkable protection to the whole IoT environment. This work leverages on the assumption that the malicious traffic is not always blocked at gateway level, but it can flow device-wise. It is also assumed that an adversary may have access to the internal IoT network. The latter assumptions allow our Network-Based IDS to significantly rise the level of protection of an IoT network.

Contributions

The main contributions of this work are:

- The development and performance evaluation of a series of light machine learning classifiers for IDS applications over IoT environments. The performance is measured in terms of key metrics, i.e., accuracy, precision, recall, area under the receiver operating characteristic curve and samples processed per second. The classifiers are deployed on Raspberry Pi and FPGA board and tested on novel collected datasets. Furthermore the feature importance is analyzed and most influential of them are studied individually.
- IMAT, a Network-based Intrusion Detection System is proposed. It is characterized by a high degree of flexibility which allows it to be distributed easily in IoT environments. The system is based on limited-resources and high flexibility devices as Raspberry Pi and FPGA and it aims to overcome limitations of classic Network-based IDS. Specifically, the challenging integration among different natures of devices and the resource-consumption inhibits the performance of the host where the IDS resides. Furthermore, it is proven how the system adds an additional layer of security in a smart IoT environment.

Thesis Organization

This thesis is organized as follows. The related works composed by different recent and performing intrusion detection systems applied to the IoT field are presented in Chapter 2. In Chapter 3 an overview about Internet of Things security and Intrusion Detection System is exposed. Furthermore, the Machine Learning models that are used in this work are also described. In Chapter 4, the complete pipeline of the proposed work, problems, and assumed methodology is presented. Two datasets are used to conduct the experiments and realize the feature analysis. Architecture and parameters chosen of the different ML models are following explained in the same Chapter as the deployment of such models into the FPGA and Raspberry boards. Subsequently, in Chapter 5, the results of the performed experiments are presented with relative discussions. Chapter 6 concludes the thesis and propose future works.

2

Related Works

Simple threshold-based techniques are prone to incorrectly classifying normal traffic as anomalous traffic and are unable to adapt to the evolving nature of attacks. More sophisticated anomaly detection algorithms, particularly those using machine learning, can help minimize false positives. Such approaches include deep neural networks, which promise to outperform traditional machine learning techniques for sufficiently large datasets.

2.1 Anomaly-based & Specification-based

Anomaly-based IDS works by comparing a current system activity against the normal registered behaviour. Anything that is not conformed to the normal behaviour is considered as malicious activity. This approach is characterized by efficiently detecting new attacks. Specification-based detection has the same goal of anomaly-based detection with the only difference that a human expert should manually contribute to define the rules of each specification.

LH Chang et al. [17] suggested a lightweight distributed IDS based on analysis of node's consumed in IPv6

over Low-Power Wireless PAN (6LowPAN). The sensor nodes with unusual energy consumption are identified as malicious attackers. The results show that the IDS scheme accurately and effectively recognize malicious attacks, in particular with DoS attack with a detection rate of 100%.

Hosseinpour et al. [18] implemented a distributed lightweight real-time IDS based on Artificial Immune System (AIS) which consists on an a combination of three layer: cloud, fog and edge computing. The cloud layer is used to train the detectors. In the fog layer the intrusion alerts are analyzed and in the edge layer the detectors are deployed in physical devices. The system has been evaluated on KDD-Cup99 dataset and SSH Brute Force from ISCX dataset obtaining an accuracy of 98.35%.

Hodo et al. [19] proposed an Artificial Neural Network (ANN) to prevent IoT threat, in particular DoS and DDoS attacks. The multi-level perceptron, a type of supervised ANN, has been trained with 2313 internet packet traces samples. The testing results showed an accuracy of 99.4% with 496 samples. The system obtained good results even though more attacks should be taken into account for future developments.

M Nobakht et al. [20] proposed an anomaly host-based intrusion detection framework. The system is called IoT-IDM and provides a network-level protection for devices in smart home environments. The framework takes benefit of a Software Defined Networking (SDN) architecture with machine learning techniques to detect compromised hosts. The authors deployed the OpenFlow protocol which is a network standard to deploy SDN implementation. IoT-IDM monitors the network activities of smart devices within the home and analyzes whether there is any suspicious activity. The system is characterized by great flexibility on choosing a machine learning algorithm, on the other hand it has the drawback that it can only inspect IoT devices that do not overload the SDN controller.

Lopez-Martin et al. [21] presented an unsupervised anomaly NIDS for IoT based on Conditional Variational AutoEncoder (CVAE). This method is unique due to its ability to carry out feature reconstruction, i.e., it can retrieve missing features information from incomplete training datasets which is the greatest advantage of using VAEs. In the best scenario they obtained 99% of accuracy tested on a refined version of NSL-KDD dataset. ID-CVAE is a suitable option for IoT systems due to the efficiency in computation time, flexibility and accuracy.

Bin Jia et al. [22] implemented an hybrid heterogeneous multiclassifier ensemble to detect DDos attacks and they designed an heuristic detection algorithm based on Singular Value Decomposition (SVD) to build their detection system. They constructed the component classifiers of the model based on Bagging, Random Forest, and K-NN algorithms. The system is tested on KDD CUP 1999 dataset and shows excellent results in terms of accuracy, TNR and precision. On the other hand the used dataset is considerably old.

Kozik et al. [23] deployed a distributed detection system based on Extreme Learning Machine (ELM) classifiers. The scheme uses HPC cluster resources available over the cloud for time-consuming and computationally-expensive classifier training tasks. The idea is based on the fact that edge devices are capable of running pre-built

classification models but they lack in storage and processing capabilities. ELMs obtain better generalization performance compared to other classifiers as the edge computing can be exploited in order to performing traffic analysis with sophisticated models. The performance is evaluated on CTU dataset and in general the system obtained satisfactory precision and accuracy values together with quite low error rates.

Doshi et al. [24] demonstrated in their work that by exploiting network traffic characteristics it is possible to build an efficient detection mechanism for IoT smart environments. The anomaly detection system is based on machine learning models, in particular neural networks, which can automatically detect sources of DDoS attacks. The system involves flow-based intrusion detection which only inspects the packet header and does not analyze the packet payload. All five algorithms used in for the experiments obtained a test accuracy higher than 0.99 from data collected on their own setup.

Ge et al. [25] developed and trained feed-forward neural networks models for binary and multi-class classification. Their goal was to build a model capable of recognizing a set of malicious attacks such as DoS, DDoS, reconnaissance and information theft against IoT devices. Algorithms are trained and tested on a recent realistic network traffic dataset, BoT-IoT. The classifier obtained an accuracy of around 0.99 for binary and multi-class classification, and 0.98 for normal traffic classification.

Hasan et al. [26] tested the performance of different Machine Learning models on the DS2OS traffic traces dataset which is composed by normal traffic and various malicious attacks traffic. The authors used Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF) and Artificial Neural Network (ANN) with the aim of detecting attack traffic and build an anomaly-based IDS. The results showed that RF performs better, in general, with an accuracy of 99.4%.

Almiani et al. [27] presented a full-automated anomaly-based IDS for Fog security against malicious cyber-attacks which is based on a Recurrent Neural Network (RNN). The RNN was trained by an adaptive version of back-propagation algorithm for enhanced prediction capability of the normal and malicious classification. Their model is designed to be implemented for Fog computing security for IoT devices protection. The authors tested the model using a balanced version of the dataset NSL-KDD. Even if the IDS does not obtain extremely high results it is capable of efficiently working in real-time smart environments.

Hussein et al. [28] proposed an anomaly-based IDS based on hybrid feature selection techniques for Random Forest. The model can be used to detect intrusions in IoT environments with high performance and accuracy. The efficient results are based on the fact that the highest-ranked attributes are selected in spite of the less-significant ones which are reduced. This implies a reduction on the execution time and an improvement on the accuracy. The trained classifier is tested on IoTID20 dataset and achieved accuracy approaches 100% for the binary target prediction, 98.7% for category, the accuracy ranges from 78.1% to 95.2% for the sub-category target prediction.

Sarhan et al. [29] implemented a hierarchical blockchain-based federated learning framework to deploy a secure

and privacy-preserved IoT IDS. Their HBFL novel framework aim to a collaborative detection of IoT intrusions adopting a cloud-fog-edge topology. The authors wanted to underline the importance of sharing cyber-threat intelligence between inter-organisational IoT environments to improve security capabilities. ML-based Intrusion Detection framework is based on a hierarchical federated learning architecture and it is tested on KDD-CUP dataset obtaining an average classification accuracy of roughly 0.99% in the best scenario.

Le et al. [30] organized the network into small clusters with a similar number of nodes. Each cluster has a cluster head, which is a node that had direct communication with all the cluster members. A specification-based IDS instance is placed in each cluster head with the aim of monitoring the cluster members by sniffing their communication. In this way the save of resources is consistent. According to their experimentation, the true positive rates are very high and in some cases could reach 100%.

2.2 Signature-based

Signature-based IDSs are based on matching signature to recognize malicious traffic or intrusions. Although this approach is effective at detecting know threats, it struggles to identify new and/or variants of known attacks.

Kasinathan et al. [31] [32] focused their studies into the attacks detection for 6LoWPAN networks. The authors proposed a centralized architecture aimed to detect DoS attack where the probes sniff network traffic and send the data to the main NIDS. When the collected traffic matches an attack signature an alert is launched. This solution overcomes SVELTE [33] limitations since the IDS mechanism does not depend on the network architecture so it cannot be affected by DoS attacks against the IoT network. The proposed framework DEMO is flexible and real-word applicable for most IoT environments.

Jun et al. [34] proposed a signature-based IDS for IoT systems based on Complex Event Processing (CEP). CEP is an emerging technique which efficiently filters and processes real-time events characterized by a good response to large volumes of messages with low latency. The system workflow starts by collecting network traffic and event usage from IoT devices. Data are processed and the system performs security events detection using Event Processing Repository (EPR) and CEP engine. The IDS provides good real-time performance even though it is power-consuming at CPU level.

To address the problem of integrating security objects in IoT devices due to their limited-power capacity and resources, Oh et al. [35] implemented a lightweight security signature-based IDS. The system is based on a novel malicious pattern-matching engine. The main goal of the limited-memory system is to heavily reduce the computational cost of comparison between packet payloads and attack signatures in order to work properly in IoT devices and environments which are resource-constrained.

2.3 Hybrid Approaches

Raza et al. [33] designed SVELTE, the first IoT Intrusion Detection System. SVELTE is a real-time hybrid IDS which aims to offer an efficient tradeoff between storage cost of the signature-based method and the computing cost of the anomaly-based method. The system deploys lightweight IDS modules in resource-constrained nodes meanwhile resource-intensive IDS modules are left to the Border Router (BR). SVELTE was implemented in the Contiki OS and obtained 90% of TPR for a lossy-network configuration and almost 100% for a lossless-network configuration. The downside is that DoS attacks can violate the system if they affect IDS nodes.

Midi et al. [36] implemented Kalis, an hybrid IoT IDS which is able to choose automatically the detection techniques depending on collected network's features. The system is the first approach which does not target a single protocol. Kalis can be placed as a standalone tool on a separate external device. The performance assessment of the system was carried out on a real-world IoT scenario. Kalis obtained a detection rate of 91% with an accuracy of 100%.

Anthi et al. [37] described the initial stages of developing a novel hybrid-based IDS for the IoT called Pulse. The system employs Machine Learning (ML) algorithms and it is capable of successfully identifying network scanning probing and simple forms of Denial of Service (DoS) attacks. Pulse consists of two main phases. The first phase consists on building a real IoT smart-home testbed where the normal activities were monitored for each device connected on the IoT network. In the second phase, malicious activities were involved to these devices creating anomaly network traffic. These phases fed a supervised machine learning technique with proper training data which composed the core of the intrusion detection model. Results showed that the model is better at detecting probing attacks (from 95.5% of precision for Intense/Regular scan to 97.7% for Quick Plus scan) than it is for flood-type attacks (80.8% of precision for SYN Flood and 81% for UDP Flood).

Considerations

As previously stated, the best performing IDS are anomaly-based which are dependent on machine learning techniques and they achieve remarkable results in the area of network-level security for IoT environments. However, most of systems have been tested on rather old datasets even though the advancement of malware such as different attacks on smart and industrial networks is continuously advancing. Furthermore, most recent IDS systems are centralized and placed in a single network node or cloud-based, and this can still pose a security threat if an attacker manages to access the internal network and bypass the single centralized intrusion detection system or the connection between IoT devices and cloud-based system fails.

These reasons paved the way for a search for machine learning models capable of achieving excellent performance with recent datasets, i.e., containing malicious attacks collected in the past few years. And consequently, base an IDS on such models that adds an additional and reliable layer of security at the network level.

References	Method	Detection	Placement
LH Chang et al. [17]	IDS over 6LowPAN	Anomaly	Distributed
Hosseinpour et al. [18]	Lightweight IDS based on AIS	Anomaly	Distributed
Hodo et al. [19]	ANN	Anomaly	Centralized
Nobakht et al. [20]	Host-based IDS based on OpenFlow	Anomaly	-
Lopez-Martin et al. [21]	CVAE	Anomaly	-
Bin Jia et al. [22]	Hybrid Heterogeneous Multiclassifier	Anomaly	-
Kozik et al. [23]	IDS based on ELM	Anomaly	Centralized
Doshi et al. [24]	NN	Anomaly	Centralized
Ge et al. [25]	NN	Anomaly	-
Hasan et al. [26]	LR, SVM, DT, RF, ANN	Anomaly	Distributed
Almiani et al. [27]	Automated IDS based on RNN	Anomaly	-
Hussein et al. [28]	Hybrid feature selection for RF	Anomaly	-
Sarhan et al. [29]	HBFL	Anomaly	-
Kasinathan et al. [31] [32]	DEMO	Signature	Centralized
Jun et al .[34]	IDS based on CEP	Signature	Centralized
Oh et al. [35]	Lightweight IDS	Signature	Distributed
Le et al. [30]	Cluster organization	Specification	Hybrid
Raza et al. [33]	SVELTE	Hybrid	Hybrid
Midi et al .[36]	Kalis	Hybrid	Hybrid
Anthi et al. [37]	Pulse	Hybrid	-

Table 2.1: Related works table.

3

Background

In this chapter we present a review of the main tools and components involved in this work. The environment security of IoT networks and the features of intrusion detection systems are described first. The architecture of the machine learning models used is as follows. We also cover techniques and frameworks used to deploy machine learning algorithms into a real-world application.

3.1 IoT Security and Intrusion Detection

The Internet of Things is growing fast and it 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 the IoT networks functional and reliable. Despite the widespread use of IoT networks, they are quite different from a normal network. In particular, IoT systems are constrained in terms of computational capability and complexity, they are heavily distributed and heterogeneous thus, a centralized traditional solution may not be always suitable [11].

The topic of security involves several aspects and features of IoT networks. As the IoT devices keep evolving the need of a human touch in the network is decreasing. This fact implies that different IoT devices communicate with each other autonomously [38]. Many of them can directly control other devices based on the conditions of the environment or any sensible signal they are programmed to respond to. Furthermore, due to many kinds of new devices being released with insufficient safety checks beforehand, it has been found that in 2015 more than 90% of IoT device firmware had security vulnerabilities leading to protocol security problems [39]. On the other hand, one of the most alarming limitation of IoT is the low capacity and constrained conditions of the devices. In fact, several kinds of lightweight devices are involved in an IoT network, especially sensors, which have limited computational power and storage. 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 [38]). Furthermore, another serious threat, due in part to IoT networks' fast and wide proliferation, are DoS or DDoS started by a botnet. We mentioned Mirai botnet in Sect. 1 as it has been a vehicle to perform one of the most powerful DDoS attack in the history of technology [40]. Alternative versions of Mirai have been created and distributed in the last years. It is important to point out the fact that these different versions can still be harmful and can cause serious problems as stated by Zhou et al. [38]. It is clear that if such a malware whose intrusion method is well known can still cause damage then the security practice in IoT devices still lacks consistency.

Intrusion Detection Systems aim to identify malware, malicious access or any kind of attack to defend internal networks. They represent one major research problem in cyber security and as there are several risks concerning networks there are different systems built to secure an environment from external attacks [41]. The aim of IDSs is to provide a wall of defense as they can be used to detect and analyze types of malicious network communications and computer systems usage, whereas conventional firewalls cannot perform this task.

There are several different kinds of Intrusion Detection Systems, and the most common way they can be classified is in three main groups [42] [12]:

- **Network-Based Intrusion Detection (NIDS)**: is the most common solution in the market. Network-based IDSs are deployed in a network node to analyze incoming network traffic that regards multiple host nodes. They are mainly installed into sensors which run in stealth-mode, in order to make it difficult to find them for an attacker [42].
- **Host-Based Intrusion Detection (HIDS)**: works on individual device or computer level. The fact that they monitor a single component makes them more precise in the analysis and they can detect elements that network-based IDSs often miss. Host-Based IDS are more difficult to maintain, as they require host-specific configuration and inhibit the performance of the device. While NIDSs work by inspecting network traffic and derived features, HIDSs can also use other features such as system calls, file access, resource consumption, etc.
- **Application-Based Intrusion Detection**: can be considered as an extension of Host-Based IDS with the only difference of analyzing events at software level. It is prominent the fact that they monitor the real-time actions of physical users even though App-Based IDSs can sometimes be more vulnerable then classic Host-Based IDSs.

Motivated by an increasing number of vulnerabilities, attacks, and information leaks, IoT device manufactures, 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

The machine learning models used and studied in this work will now be explained. They are different kinds of Neural Networks, Decision Trees and Random Forests. As stated above, in general, the best performing and state-of-art models in the Machine Learning and Deep Learning field are high resources-consuming. For the purpose of this work, the research and use of light-ML models is fundamental to efficiently deploy them on cheap and low power consumption devices.

3.2.1 Neural Networks and Convolutional Neural Networks

In 1943, McCulloch and Pitts proposed the first artificial neuron model, also called *perceptron*. This model, shown in Fig. 3.1, is characterized by important features that have paved the way for modern machine learning and deep learning. Specifically, synapses are represented by a set of weighted w_i inputs. The membrane of the cell that collects electrical charge is represented by a summation of input signals and an *activation function* (also called *transfer function*, initially a threshold function) that decides whether the neuron "spikes" for current inputs. The sum of weights and the threshold activation function are exposed below:

$$net_j = \sum_{i=1}^{n} w_{ij} \cdot x_i \tag{3.1}$$

$$o_j = \begin{cases} 1 & if \quad net_j \geq \theta_j \\ 0 & if \quad net_j < \theta_j \,. \end{cases} \tag{3.2}$$

The sum is used as input into an activation function as shown in Fig. 3.1 and the final output o_j is obtained after the activation function, where θ_j is the threshold value.

A single neuron is not particularly useful on its own, it only reacts according to the value of its transfer function. However, by grouping and structuring a collection of single neurons into a proper neural network, it is possible to create a model which can learn and adjust the weights on its own accordingly to produce a desired result or accomplish various tasks.

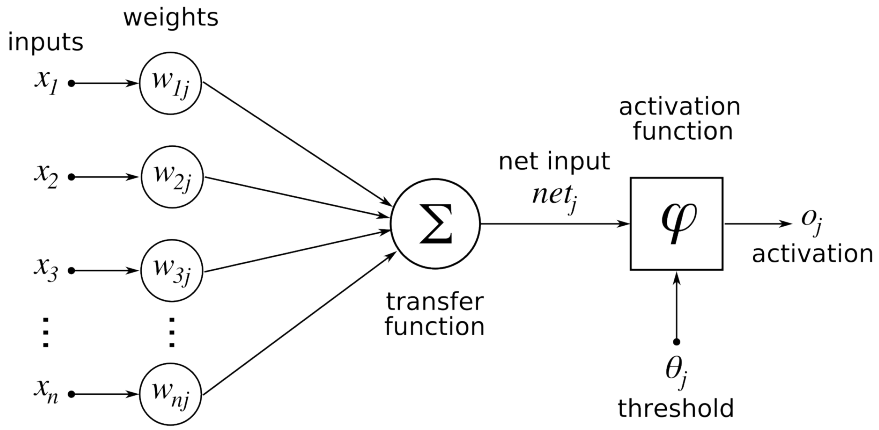

Figure 3.1: A diagram model of an artificial neuron as proposed by McCulloch and Pitts in 1943.

Neural Networks (NN) are networks composed of artificial neurons connected to each other and divided into layers whose structure is inspired by the human brain [43]. Each connection can transmit a signal to other neurons and the output of each neuron is computed by some non-linear function. The connections between nodes are called edges which have a weight that are adjusted in the learning process and they represents the "intensity" of that connection. Typically, the neurons are organized into multiple layers (Deep Neural Network) and neurons of one layer connect only to neurons of the immediately preceding and immediately following layers. Training a Neural Network consists of processing examples, each associated with a label (in the supervised-learning scenario), and is usually conducted by determining the difference between the processed output of the network and a target output (which represents the correct result) and consequently adjusting the weights. Following a training phase, the models identify an underlying pattern between the data. Feed forward neural networks are typically represented by composing together many different functions. In the following example, f^1 is the first layer, f^2 is the second layer, and f^3 is the output layer of the network:

$$f(x) = f^3(f^2(f^1(x))) \,. \tag{3.3}$$

More specifically, a layer is a function of the preceding one. In the general case we can describe a layer as follows:

$$h^i = g^i(W^i \cdot h^{i-1} + b^i) \,. \tag{3.4}$$

W indicates the matrix of the weights, b the bias, i the number of layer and h and g functions. Neural network is an extremely powerful and flexible tool which can be deployed to perform several tasks like classification, regression, prediction, clustering and many other [43]. By the *Universal Approximation Theorem* [44], we know that a large

enough Multilayer Perceptron (MLP), which is a fully connected class of feedforward neural network, will be able to represent any function. We are not guaranteed, however, that the training algorithm will be able to *learn* that function.

Deep Neural Network (DNN) and Deep Convolutional Neural Networks (DCNN, or simply CNN) are part of the sub-field of Machine Learning, called Deep Learning (DL). Deep learning allows complex mathematical models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [45]. This field is making outstanding advances in solving problems which seemed impassable until recently. Convolutional Neural Network are a class of ANN specialized for processing grid-like structures.

CNNs differ from other ANNs for the usage of a mathematical operation called *convolution* in place of general matrix multiplication in at least one of their layers. Inputs in CNNs are tensors with a specific shape that could be one-dimensional, two-dimensional or three-dimensional. After the convolutional layer, the input becomes abstracted to a feature map, also called an activation map. CNNs are mostly applied to visual imagery tasks for the fact that they are specifically designed to process grid-like/pixel data and furthermore, the convolution provides a means for working with samples of different size [44]. The main properties of Convolutional Neural Networks are that it works with inputs of variable size and the convolution is not equivariant to scale change and rotation. *Sparse interactions* and *parameter sharing* (i.e. instead of learning a separate set of parameters for each location, it learns only one set) imply efficiency and much fewer weights compared to fully connected neural networks.

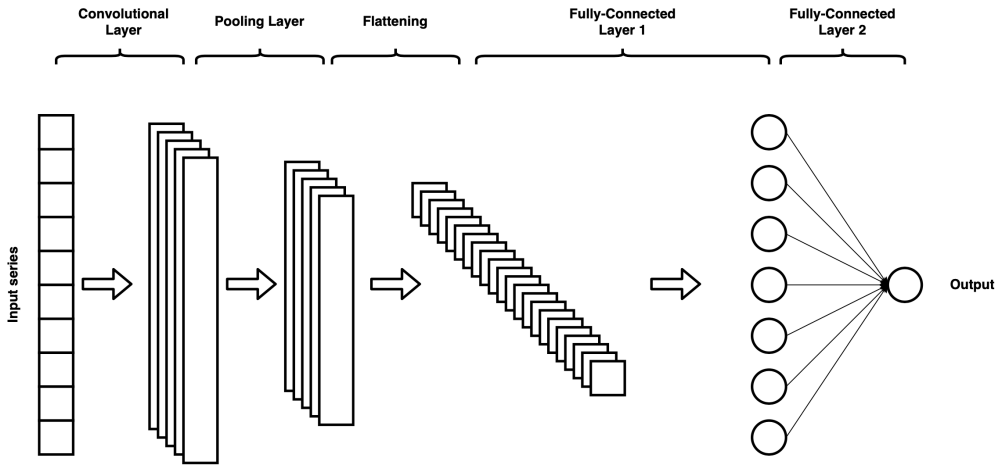

Figure 3.2: Example of 1D Convolution Neural Network.

3.2.2 Decision Tree

Decision Tree (DT) is a non-parametric supervised learning tree-like model that is mostly used for classification and regression problems [3]. The goal of Decision Tree is to create a model that can correctly predict the value of a target by learning simple inferred decision rules. A tree is composed of different nodes, and those nodes are selected (in accordance with a criterion) looking for the optimum split of the features. The most commonly used criterias are *Gini* and *Entropy*:

$$GiniIndex = 1 - \sum_i p_i^2 \tag{3.5}$$

$$Entropy = -\sum_i p_i^2 \cdot log_2(p_i) \tag{3.6}$$

where p_i is the probability of the class i. Gini impurity measures the regularity at which any sample of the dataset will be mislabelled when it is randomly labeled, meanwhile, the Entropy depicts the disorder of the features with the target [46]. In general, the Gini criterion is faster due to its less expensive computation.

The fact that DT is a simple and efficient model makes it suitable on resource constrained device. Although the accuracy obtained by DTs is not, in general, as high as the Random Forest's accuracy, DTs are extremely fast at test time and they can process hundreds of thousands of networks packet features per second. Thus, this model is a good candidate for an intrusion detection application at network level. Furthermore, Decision Trees are incredibly fast on training phase, and this fact opens the training-on-device and incremental-learning possibility on small controllers. Incremental models may be very powerful as they fit to data in a quick and efficient way, which means they can adapt to changes in real time processing [47].

3.2.3 Random Forest

Random Forest (RF) is a supervised machine learning model that, as Decision Tree, is widely used in classification and regression tasks [3]. A Random Forest is based on an ensemble technique called Boostrap Aggregation (or Bagging). The algorithm consists on repeatedly taking (with replacement) a number of random records from the dataset where individual decision trees are constructed for each sample. Each decision tree will generate an output and the final result consists on majority voting or averaging, for classification and regression respectively. Due to the fact that Random Forest is an ensamble method composed of a multitude of individual classifiers it is slower than a single decision tree but better avoids overfitting and in general, is a more stable model.

Ensembles are based on the concept that a multitude of single entities working together for the same purpose will achieve better results than a single model. RF is based on the same idea: several relatively unrelated decision trees are used to accomplish the same task as a committee.

The low correlation between the models of such an algorithm, which is one of its most important features, can be compared to an investment portfolio in the economic sphere [48]. Since different types of stocks and bonds are collected in one portfolio, the uncorrelated models of RFs in the aggregate perform better than the results taken individually. Consequently, even if some trees in the ensamble are wrong, many others obtain correct values and thus protect the final result from individual errors.

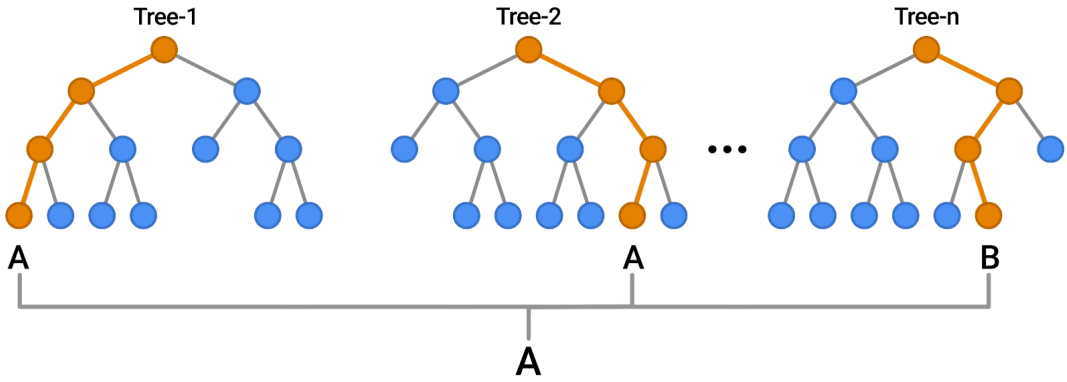

Figure 3.3: Random Forest's voting outcome example [1].

3.2.4 AdaBoost

AdaBoost stands for Adaptive Boosting [3]. It is a boosting-technique that aims at combining multiple weak classifiers to build one strong classifier. AdaBoost is an Ensamble Method, in particular, it belongs to the category of Boosting methods. These algorithms try to build a strong predictive model from the mistakes of several weaker models. It happens that a single classifier may not be very precise with a task, but with multiple weak classifiers that progressively learn from each others' wrongly classified objects, it is possible to build one strong and consistent model. The AdaBoost's pseudo-code Alg. 3.1 is presented below. The final classifier (i.e. the output of the algorithm) is based on a linear combination of the weak classifiers.

Algorithm 3.1 AdaBoost Pseudocode

Require: $(x_1, y_1), \ldots, (x_n, y_n), x_i \in X, y_i \in Y = \{-1, +1\}$
Require: Set of base learners ▷ i.e. weak classifiers
  Initially set uniform example weights $D_1(i) = \frac{1}{m}$
  for $t = 1 \ldots T$ ▷ for each weak classifier
    Train *base_learner* using distribution D_t
    Test *base_learner* on all data
    Set *learner* weight with a weighted error
    Update example weights based on ensemble predictions
  end for
  $Output \leftarrow sign(\sum_i^T \alpha_t \cdot h_t(x))$
  ▷ with $h(x)$ weak hypothesis, and α coefficient

3.2.5 Extremely Randomized Trees

Extremely Randomized Tree (ETC or ET) is an estimator that fits different randomized Decision Trees [3], also called Extra Trees, on various sub-samples of the dataset. It is a tree-induction algorithm for performing supervised classification or regression. The output is based on averaging results to efficiently improve the final accuracy and to help preventing over-fitting. The algorithm proposed by Geurts et. al. [49], for a numerical attribute, selects its cut-point completely at random. At each tree node, the random selection is combined with a random choice of a certain number of attributes among which the best one is chosen. In this way, the method randomly choose a single attribute and cut-point at each node in the extreme case, and hence builds totally randomized trees.

Extremely Randomized Trees, unlike other tree-based ensemble models, apply complete training samples in order to grow the trees and in the meantime minimizing bias and variance. The splitting procedure of ETC depends on three important parameters:

- The number of features selected at each node.
- The minimum training set size for splitting a node.
- The number of trees in the ensemble.

Although extremely random forest and normal random forest perform almost the same, in the case of optimal feature selection ETC is in general computationally faster than RF.

3.2.6 TabNet

Attentive Interpretable Tabular Learning [2], or TabNet, is a novel high-performance and interpretable canonical deep tabular data learning architecture that uses sequential attention to choose which features to reason from at each decision step with the result of more efficient learning. TabNet outperforms different tabular learning models on various datasets for classification and regression tasks.

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

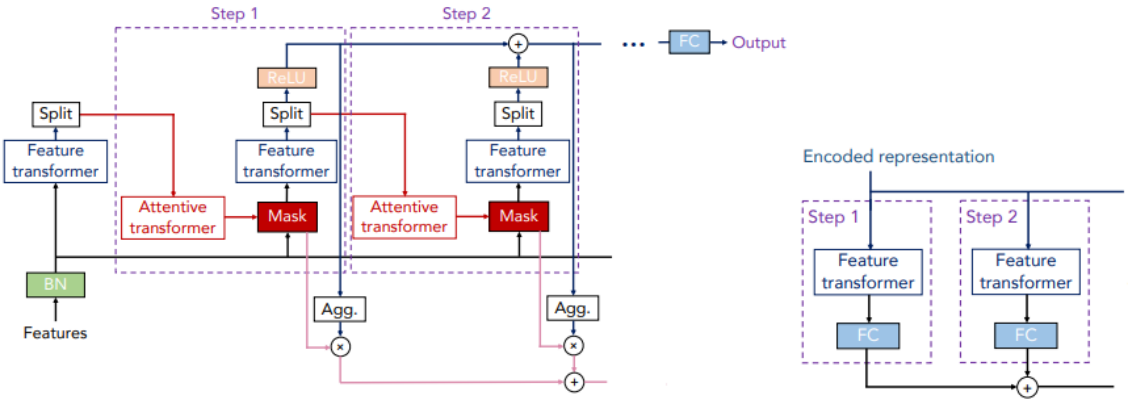

Figure 3.4: TabNet's Encoder-Decoder Architecture [2].

3.3 Raspberry Pi

The Raspberry Pi is a small but powerful single-board computers (SBC) developed by the Raspberry Pi Foundation [50]. Raspberry Pi it is to all intents and purposes a computer and thanks to its reduced power consumption and versatility it has thousands of different applications.

There are three series of Raspberry Pi, they feature Broadcom system on a chip (SoC) with an integrated ARM-compatible CPU and on-chip GPU, while Raspberry Pi Pico has a RP2040 system on-chip with an integrated ARM-compatible CPU. Raspberry's hardware has evolved through several versions that includes different kind of processors, graphic-units, network support and other features. The first generation Raspberry Pi, the Model B, was released in February 2012 and has developed into several families, such as the Raspberry Pi 2, Zero, 3, 4, and Raspberry Pico. The Zero features small size and reduced input/output (I/O) capabilities, while the Pico was the first Raspberry Pi board based on a single micro-controller chip, the RP2040, designed by Raspberry Pi in the UK.

As the Raspberry Pi Foundation says "It's capable of doing everything you'd expect a desktop computer to do [...]. The Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infrared cameras". Raspberry Pi board is an extremely flexible and useful option to test different Machine Learning models in a low-power and limited-resources environment. Raspberry Pi kernel follows behind the main Linux Kernel and the Debian-based OS's running on the Raspberry Pi allow to easily install and run Machine Learning and Deep Learning libraries such as PyTorch [51] and Scikit-learn [3]. In particular, Raspberry Pi 3 Model B is used in this work. A comparison between different Raspberry's generation models is shown in table 3.1.

Specifications	Raspberry Pi Model B	Raspberry Pi 3 Model B
CPU	BCM2835 SoC ARM1176JZF-S	BCM2837 Soc ARM Cortex-A53
CPU Speed	700 MHz	1.2 GHz
CPU Cores	1	4
RAM	512 MB	1024 MB
Power Requirements	5V @ 700 mA	5V @ 2.5A

Table 3.1: Comparison between the specifications of the Raspberry Pi which was released in 2012 and Raspberry Pi 3, shown in Fig. 3.5 which it was released in 2016.

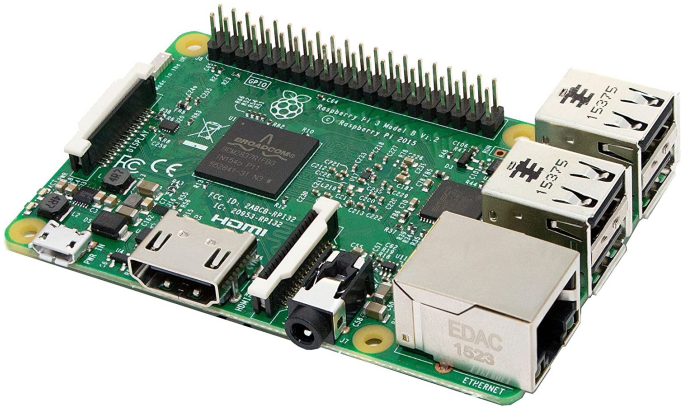

Figure 3.5: Raspberry Pi 3B board.

3.4 FPGA

Field-Programmable Gate Array (FPGA), it is an integrated-circuit (IC) that allows to design custom digital circuits [52]. They can be reprogrammed to desired application or functionality requirements. Because of the FPGAs flexibility, it is possible to implement entire processors using its digital logic and this follows that FPGAs are an ideal fit for many different markets such as aerospace, automotive, industrial and many others. FPGAs contain an array of programmable logic blocks and a hierarchy of links allowing blocks to be wired together. It is possible to configure the individual cells to operate in certain ways and then connect the cells together to form the basis of digital circuits. Logic blocks can be configured to perform complex combinational functions and in most FPGAs, logic blocks also include memory elements, which may be flip-flops or more complete blocks of memory. FPGAs have a remarkable role in embedded system as they integrate extremely well with other hardware devices. Furthermore, as we will see from the results in the next chapters, FPGAs designed to perform intrusion detection in an IoT network are notably fast and capable of processing hundreds of thousands of packets per second. FPGAs provide great flexibility and remarkable performance.

In this work we used a PYNQ-Z2 board which is an FPGA development board. PYNQ is an open-source project designed by Xilinx [52] which makes it easier to use Xilinx platforms providing Jupyter-based framework with Python API. PYNQ enables designers to exploit the benefits of programmable logic and microprocessors to build high performance applications. PYNQ can be used with ZYNQ or other accelerators boards, in particular in the PYNQ-Z2 the XC7Z020-1CLG400C chip is used. Specifically, ZYNQ-7000 SoC [52] devices integrate a single or dual core ARM Cortex-A9 based Processing System (PS) and 28 nm Xilinx Programmable Logic (PL). Furthermore, the PYNQ-Z2 board features 512MB DDR3 of RAM, 1G Ethernet controller, 4 push-buttons, 2 slide switches, 6 LEDs, 2 Pmod ports and a RaspberryPi and an Arduino header. The systems which can be created by the ZYNQ accelerator are characterized by parallel hardware execution, real-time signal processing, low latency control, high bandwidth IO and real-time signal processing.

We are able to implement and run machine learning models, in particular neural networks, into the PYNQ-Z2 board with the help of the framework FINN [53] which is following presented. At the moment, FINN is open-source and publicly available *.

*https://github.com/Xilinx/finn

3.4.1 FINN

FINN is an experimental tool designed by Xilinx Research Lab in order to implement Deep Learning model on FPGAs. 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 Networks (QNNs) trained in PyTorch with the help of Brevitas [54] which is an helpful 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 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 the further processing. 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.

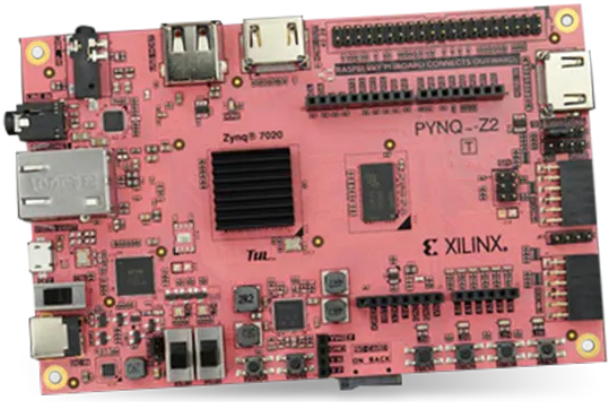

Figure 3.6: PYNQ-Z2 FPGA board.

4

Proposed Approach

In this section the architecture, concept, design principles and the pipeline of the conducted experiments of the proposed work are presented. We first state the objectives as a problem formulation. We describe the network traffic datasets, the importance of the features and the different metrics involved in the tests. Then we show how the chosen Machine Learning models, exposed in Sect. 3, are built and deployed to the Raspberry Pis and to the FPGA in order to perform the Intrusion Detection task. The experiments' results and the related considerations are shown at the end of this section.

4.1 Problem Formulation

The main objective of the proposed work is to build different light Machine Learning models which are trained to correctly classify between benign and malicious network traffic and test them into small and resource-constrained devices. The search for such models is mainly based on the accuracy in performing the intrusion detection task (i.e., detecting malicious packets) and the number of samples the model is able to process per second. This way, we present a set of classifiers trained to recognize different kinds of recent attacks to an IoT network and implement

them into Raspberry Pis and FPGAs.

Consistent research has been conducted on wide appliances for network-security, but these do not scale well to light-weight systems such as Internet of Things environments. In the overall picture, we propose a flexible and efficient Network-Based Intrusion Detection system aimed to efficiently defend an IoT network from malicious attacks. Specifically, different small controllers (i.e. Raspberry Pi's and/or FPGAs) are deployed and integrated to the single IoT devices into the LAN to guarantee protection from malicious attacks. The architecture is shown in Fig. 4.1. Network traffic analysis combined with machine learning techniques is an effective method of identifying abnormal activities in an IoT environment for real-time alerting. 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.

On the other hand, we want to overcome some limitations of Intrusion Detection system. In particular NIDSs may have difficulty processing all packets in a large or busy network and the integration of an IDS system to a single device may be hard to manage as different architectures must cooperate efficiently without any interference. The idea of the proposed work is to introduce a high level of flexibility and autonomy in the IDS to overcome these limitations. The IDS devices involved in the proposed method stand out for their capability of integration between devices of different architectures. Furthermore they are characterized by low-power, low-resource consumption and they can run autonomously. In this way the performance of the IoT devices will not be affected and at the same time they will be defended against malicious attacks.

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. 4.2. The single steps are following described.

4.2.1 Traffic Capture

The first step of the workflow consists of real-time and always running traffic capturing. Raw network data which includes ingoing and outgoing packets can be collected and processed with different tools such as Zeek [55] and

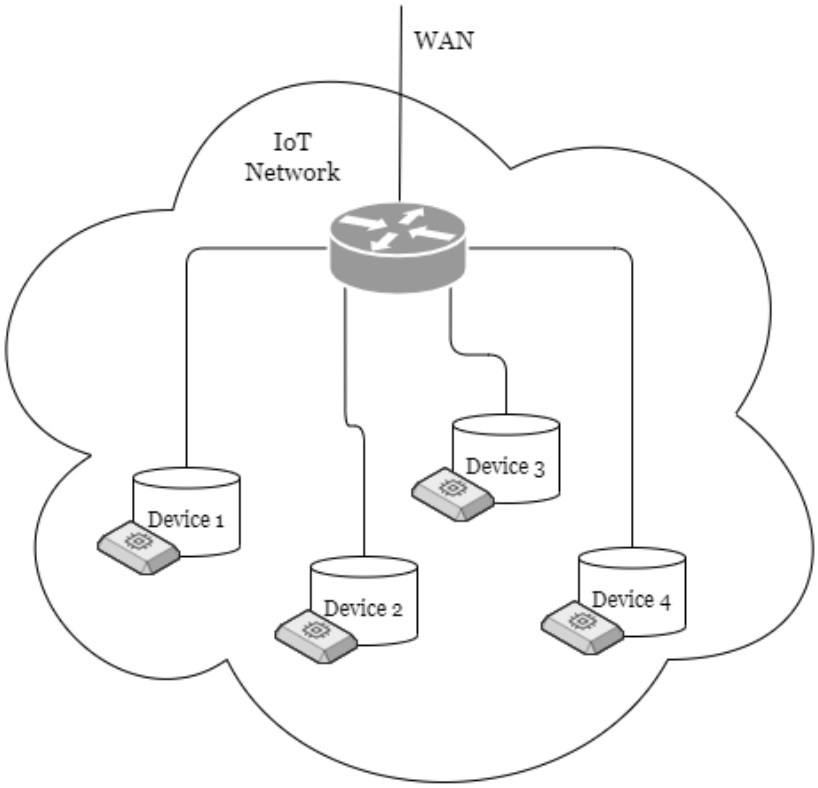

Figure 4.1: Proposed network-based intrusion detection system applied architecture.

Wireshark [56]. With the purpose of testing, in our experiments we used two different datasets, TON [5] and IoT-23 [57]. They both provide huge amount of network traffic collected in packet capture (pcap) formats and CSV files for TON and conn.log.labeled files for IoT-23 which is the Zeek conn.log file obtained by running the Zeek network analyser.

4.2.2 Feature Extraction

After the network capturing we extract the relevant fields from each single network packet. In particular, each field corresponds to a feature. We do not consider aggregations of packets related to devices or time, but we compute features on the single packet.

Feature extraction is a key step for the final efficiency and accuracy of Machine Learning models, as they to-

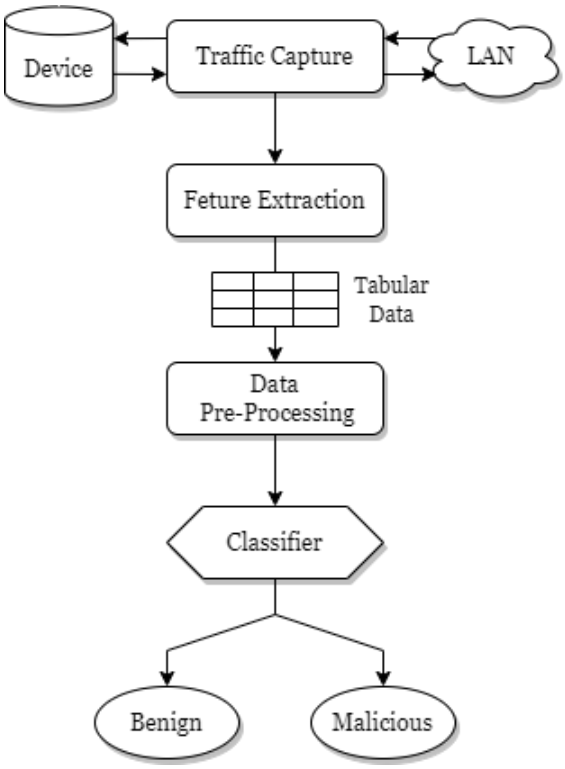

Figure 4.2: Workflow of the proposed framework, starting from the data collection to the classification.

tally depend on the quality and quantity of information provided. It is therefore necessary to search for features characterized by a high degree of entropy, that is, which contain high amounts of information. In the field of Machine Learning, especially during a classification task, the best features are those that contain the key information that characterizes a sample. In other words, these features give a remarkable help to the correct classification of a sample. Therefore, we try to extract generic packets features of the traffic instead of focusing of the characteristic of the single attacks or the specific behavior of an infected IoT device. In particular we concentrate on header fields in the IP packet, which include the size of the packet, IPv4 and TCP/UDP related information. In the next section we will list the features selected for each dataset and the consequent reasons.

4.2.3 Feature Processing

This is a crucial step before feeding the data into an ML model. It is well known that the results and success of a machine learning algorithm are highly dependent on the data provided. Pre-processing consists of different procedures to make acceptable and optimize the input data to a machine learning algorithm. In particular, these

models are discrete entities based on mathematical algorithms and procedures, requiring purely numerical data in order to function.

In general, the datasets are composed of different types of data (e.g. integers, floats, doubles, binary, strings, etc.), the data can often be proportioned between them having large dimensions or being too small. Non-numeric inputs need to be mapped to numbers or to be converted in one-hot-encoded values and normalized in order for the features to contribute equally. Data normalization is one of the pre-processing approaches and it is very important for improving data quality and subsequently the performance of ML algorithms [58]. It consists on either scaling or transforming data with the aim of making an equal contribution of each feature.

We did not use one-hot encoding in our work as it would have dramatically increased the size of the data sets used inhibiting the performance of the resource-limited devices employed. Instead, we mapped non-numeric values to numbers and applied data normalization for pure numeric features.

4.2.4 Training and Classification

The last part of the IDS pipeline consists on the classification of the collected data between benign and malignant traffic. This classification phase is preceded by a training phase of the Machine Learning model used. In this phase we take the processed data from the training network traffic of the Dataset into a model for training. The resulting trained model is deployed as IDS classifier when new traffic comes. In our work we have selected different ML models to test in different compositions, i.e. in different architectures. In the classification phase, the classifier takes as input the processed data and outputs whether this packet belongs to traffic considered malicious or not.

4.3 Datasets

In this work we used two different datasets, i.e. TON_IoT (Network) Dataset [5] and IoT-23 Dataset [57]. We chose these datasets as they are the most recent data collections with malicious and benign IoT network traffic. These datasets bring significant information at the level of malware families that even modern security solutions are unfamiliar with. Both TON_IoT and IoT-23 contain different kind of attacks (such as DoS, DDoS, ransomware, etc.) related to IoT networks and they are extremely useful to validate and test Machine Learning models which accomplish Intrusion Detection tasks.

TON_IoT datasets are a new collections of Industry4.0, Internet of Things and Industrial IoT new generation data. They were collected from large-scale realistic network designed at the Cyber Range and IoT Labs, School of Engineering and Information technology (SEIT), UNSW Canberra. The testbed was designed based on interacting network, IoT devices and systems. The environment is composed by three layers with the aim of mimicing the implementation of recent real world IoT networks. Specifically, the *edge layer* which involves the physical devices, the *fog layer* and the *cloud layer* which consists on the cloud services configured online in the testbed. The malicious scenarios involved nine different attack categories launched against vulnerable IoT applications, operations systems and network systems. The attacks categories are listed in table 4.1. Several heterogeneous sources of data are included into the dataset, in particular belonging to sensors, operation systems and network traffic. In our work we focus on the network traffic records (i.e. packets), which are extracted with Zeek [55]. The TON-IoT dataset comes with the extracted traffic flow in CSV format. The authors also provided a Training and Test collection which consists on a smaller portion of the dataset for ease of use on Machine Learning field application. In our work we used the Training and Testing portion split of TON Dataset. The composition of the network traffic data are shown in table 4.1.

The IoT-23 dataset [57] 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 is composed of 23 different network captures, also called *sessions*, with 20 malware and 3 benign network traffic captures. During the collection of the data, different malicious scenarios were executed related to a specific malware which performed different actions in a Raspberry Pi. On the other hand, the benign traffic 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. The devices were not simulated and this implies that the collection is characterized by the real network behaviour. We used different scenarios of IoT-23 dataset that are shown in the table 4.2. The dataset, for each capture, 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. In out experiments, we derived 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.

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

Table 4.1: Statistics of Network Records of TON_IoT dataset [5]. In particular, the attacks collected are Backdoor, Distributed Denial of Service and normal DoS, Injection attacks Man-In-The-Middle, Password stealing attacks, Ransomware, Scanning and Cross-Site-Scripting (XSS).

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

Table 4.2: IoT-23 Dataset scenarios used in our work. FD stands for FileDownload, PortScan is the short for PartOfHorizontalPortScan and HB is the short of HeartBeat. In details, *FileDownload* label indicates that a file is being downloaded to the infected device. *Attack* label refers to some kind of attacks which try to exploit flaws such as telnet login, brute force, command injection etc.

The IoT23 scenarios contain different kind 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 by 1,048,484 benign samples and 895,929 malicious samples. Furthermore, we tested *IoT23-44-1* scenario with the aim of observing how the selected ML models can perform with extremely reduced amount of data.

From TON and IoT-23 datasets we extracted a subset of features that we used to train the Machine Learning models. In particular, we concentrated the extrapolated information 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, SSL, HTTP activity, focusing more on addresses, ports, amount of bytes transmitted, amount of packets transmitted, duration of transmission and protocol used. From both BOT and IoT-23 dataset we extracted 14 features. The chosen features, shown in Table 4.3, are the same for both datasets even if they appear with different names. It should be noted that even if the features chosen for both TON and IoT-23 are the same, the testbeds are different as are the types of attacks collected in the datasets. Consequently the samples will have different characteristics.

Another determining factor that we have taken into consideration to achieve good performance using devices with limited resources is the amount of features extracted. Each feature, in fact, will have to be subjected to a pre-processing and consequently will use a part of the processing capacity of the device. And the same goes for the memory of the single devices involved.

	TON		IoT-23	
No.	Name	Type	Name	Type
1	ts	Time	ts	Time
2	src_port	Number	id.orig_p	Number
3	dst_port	Number	id.resp_p	Number
4	proto	String	proto	String
5	service	String	service	String
6	duration	Number	duration	Number
7	src_bytes	Number	orig_bytes	Number
8	dst_bytes	Number	resp_bytes	Number
9	conn_state	String	conn_state	String
10	missed_bytes	Number	missed_bytes	Number
11	src_pkts	Number	orig_pkts	Number
12	src_ip_bytes	Number	orig_ip_bytes	Number
13	dst_pkts	Number	resp_pkts	Number
14	dst_ip_bytes	Number	resp_ip_bytes	Number

Table 4.3: Features chosen, from TON and IoT-23 dataset, which have been used to train the different Machine Learning algorithms. In total, 14 features have been chosen and the table shows the type of each feature.

4.4 Feature Analysis

We now evaluate the importance of the single features for the performance of our anomaly detection by exploring and describing their role. We analyze why they are relevant to differentiating benign traffic from malicious traffic. With the following features, our goal is to be able to understand the behavior of the network in an attacking or normal situation.

- **Timestamp**: of connection between flow identifiers. Timestamps are employed in network devices for various purposes for example: logs, measuring delays, monitoring the performance and in addition, they are attached to network packets for analysis purposes. Fig. 4.3 depicts the importance of this feature employed into the Random Forest.
- **Port**: refers the TCP/UDP ports of interest of a packet.
- **Protocol**: normal traffic and malicious traffic have varying protocol distribution. DoS attack traffic is usually characterized by the fact that TCP packets outnumber UDP packets as shown in Fig. 4.4 and Fig. 4.5.
- **Service**: refers to protocols detected dynamically, such as DNS, HTTP, SSL, DHCP, SMB, FTP , IRC etc. From Fig. 4.6 and Fig. 4.7 it is possible to notice that heir distribution can be significant in detecting anomalies.
- **Duration**: is the time of the packet connections, which is calculated by subtracting the time of last packet seen and the time of first packet seen. In general, malicious entries have approximately 75% less connection duration than non-malicious traffic entries.
- **Source and Destination bytes**: bytes which are respectively originated or responded from payload bytes of TCP sequence numbers. The distribution of packet sizes differs significantly between malicious and normal traffic. For instance, malicious packets in TON dataset have in general 15 times the amount of source and destination bytes compared to the normal traffic.
- **Connection State**: specifically describe the state of the connection by indicating if the connection is established, attempted, terminated, rejected, etc.
- **Missed bytes**: is defined as the number of missing bytes in content gaps.
- **Source and Destination packets**: number of packets which is estimated from source or destination systems.
- **Source and Destination IP bytes**: refer to the number of IP bytes which is the total length of IP header field of source or destination systems.

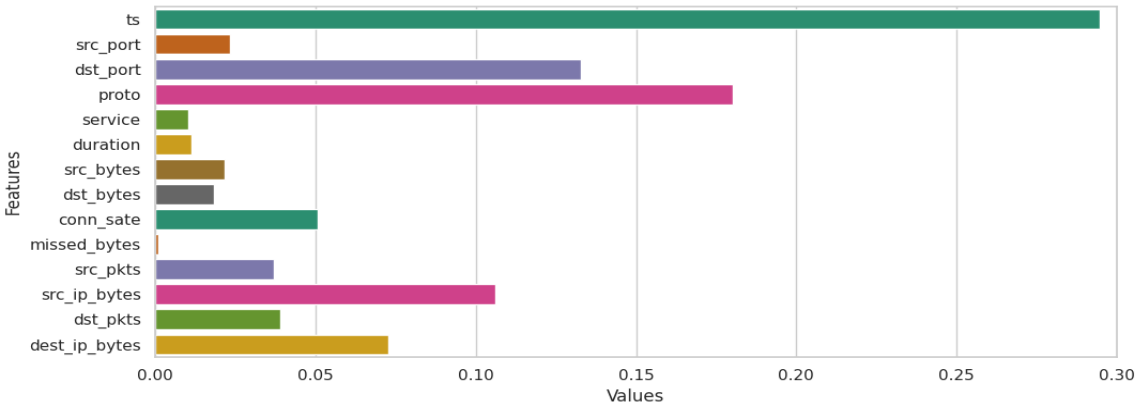

Figure 4.3: The plot depicts the Feature Importance for a Random Forest. In particular, Feature importances are provided by the fitted attribute related to the Random Forest, obtained with scikit-learn [3]. Feature Importances are computed as the mean and standard deviation of accumulation of the impurity decrease within each tree, which is represented by the y axis. In this model (Random Forest) it is possible to see that the timestamp feature is very important for the correct classification of data. In the x axis the features are represented numerically as they are shown in Table 4.3.

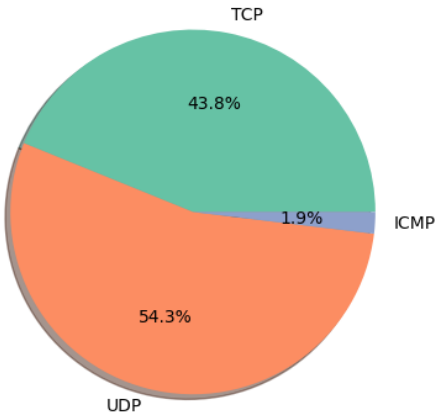

Figure 4.4: The chart represents feature statistics for normal traffic protocol distribution on TON Dataset. The protocols are TCP, UDP and ICMP. It is possible to notice how evenly TCP and UDP packets are divided in a normal traffic condition.

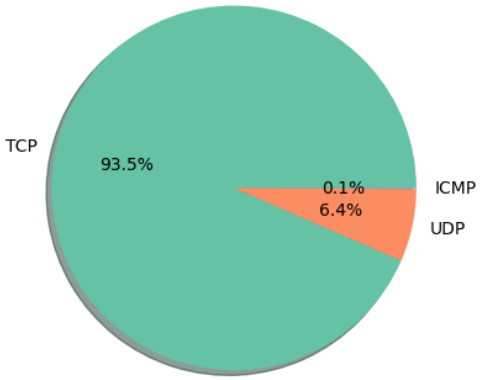

Figure 4.5: The chart represents feature statistics for malicious traffic protocol distribution on TON Dataset. It is clear from the images that the protocol represents an important feature since most of the malicious packets captured in TON Dataset are TCP packets. On the other hand, very few malicious packets are ICMP and UDP packets have reduced from 54.3% to 6.4%.

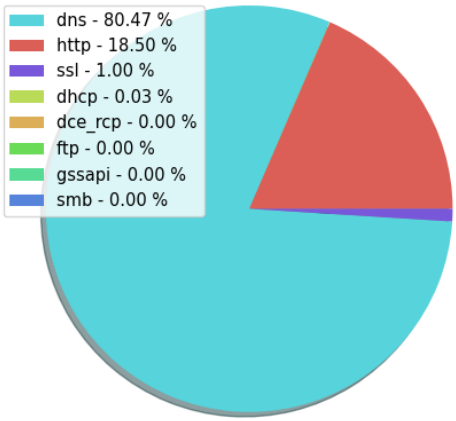

Figure 4.6: Service distribution of benign packets. The chart depicts a comparison of feature statistics for benign traffic on TON Dataset. It can be seen that most of the service types are DNS and HTTP.

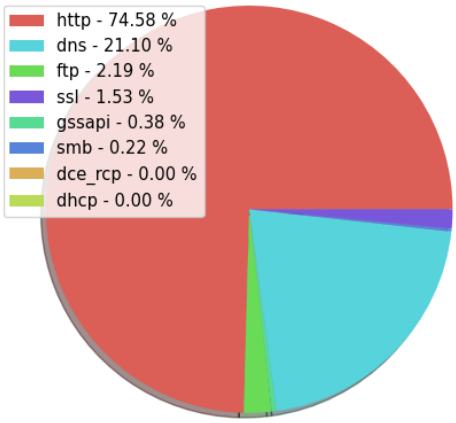

Figure 4.7: Service feature distribution of malicious packets. The chart depicts a comparison of feature statistics for malicious attack traffic on TON Dataset. It is noticed the majority of *HTTP* and *DNS* packets and the difference in usage between normal and attack purposes. Compared with benign traffic, it is noticeable that there are more HTTP packets and fewer DNS packets.

4.5 Experimental Setup

The training and validation of the chosen Machine Learning models has been carried out on a machine operated on 64-bit Ubuntu 18 and equipped with Intel Core i5-10210U four core CPU having 1.60 GHz base frequency and 4.20GHz as max turbo frequency. Afterwards, the saved trained models are transferred to Raspberry and FPGA ready to perform inference to new input data.

Classifiers are implemented in Python programming language (version 3.8) via several popular machine learning libraries, especially PyTorch [51] and Scikit-learn [3], which is also used to derive the performance and statistical results. The model architectures used and their parameters are listed below.

Neural Networks Models

The description of the NNs built and used follows. To give a better understanding of the behaviour of such models we experimented with the difference between a small NN and a bigger size NN, both for the complete feed-forward network and convolutional network.

- **NN1** (small_NN): fully connected NN 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 neural network composed by one hidden layers of 64 neurons. Input layer has 14 neurons and the output layer 2.
- **CNN1** (small_CNN): it is composed by 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 by 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 by 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.

Tree-based Models

- **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 *gini* as function to measure the quality of a split.
- **AdaBoost**: the number of estimators chosen are 100, with a learning rate of 0.1 and *SAMME.R* as a boosting algorithm [59]. 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**: the width of the decision prediction layer is set to 32 as the width of the attention embedding for each mask, they should be set as the same value according to the authors [2]. We set 4 as number of steps in the architecture, then the number of independent Gated Linear Units layers at each step is set to 3 as the number of shared Gated Linear Units.

Grid Search

For Decision Trees, RF, AdaBoost, Extremely Randomized Trees we used *GridSearchCV* to find the best hyperparameters. Specifically, Grid Search is a model hyperparameter optimization technique which exhaustively considers all parameter combinations to find the best performing one. 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 leverage the decision of the final hyperparameters between bandwidth and accuracy.

Activation Function and Loss Function

An *activation function* (or transfer function), as stated in Chapter 3 is used to determine the output of a node in a neural network according to a mapping (E.g.: the resulting values are between -1 and 1.). On the other hand, the function used to evaluate a candidate solution is referred to as the *objective function*. In optimization problems, as with neural networks, we seek to minimize the error, meaning that the algorithm is searching for a solution with the lowest amount of loss. The objective function (also called cost function or loss function) is a method of evaluating the performance of an algorithm: if the predictions are wrong, the loss function will output a higher

number while if the model makes correct predictions the loss function will output a low value.

In this work *ReLU* (Rectified Linear Unit) is used as activation function and *Cross Entropy Loss* is used as objective function for neural networks. Cross-entropy measures the difference between two probability distributions for a given random variable or set of events and it is widely used as a loss function when optimizing classification models.

ReLU is defined as : $f(x) = x^{+} max(0, x)$. Cross-entropy for binary classification is defined by Equation 4.1. For multiclass classification, cross-entropy is defined by Equation 4.2. Where y is the binary indicator if the class label i is the correct classification for the observation o. M is the number of classes and p is the predicted probability observation o is of class i.

$$-(y \cdot log(p) + (1y)log(1p)) \tag{4.1}$$

$$-\sum_{i=1}^{M} y_{o,i} \cdot log(p_{o,i}) \tag{4.2}$$

Optimization function

An *optimization function* is used to change the characteristics of neural networks such as weights and learning rate in order to reduce the loss and increase the general accuracy of the model. In our work we used Adam (Adaptive Moment Estimation) optimizer [4] for all the neural networks and for TabNet. Adam uses the momentum of first and second order and it works by reaching the minimum in a slower way but with more precision. The learning rate set for TabNet is 0.02 and 0.01 for all the neural networks. Adam realizes the benefits of both *AdaGrad* and *RMSProp* so its efficiency is remarkable when working with large problem involving several parameters. Adam's features are notable for low memory requirements, computational efficiency, ease of implementation, and adaptability to problems with sparse gradients. Intuitively, Adam is a combination of the *gradient descent with momentum* and *RMSProp* algorithm. In Figure 4.8 it is possible to see a comparison of Adam to other optimization algorithms.

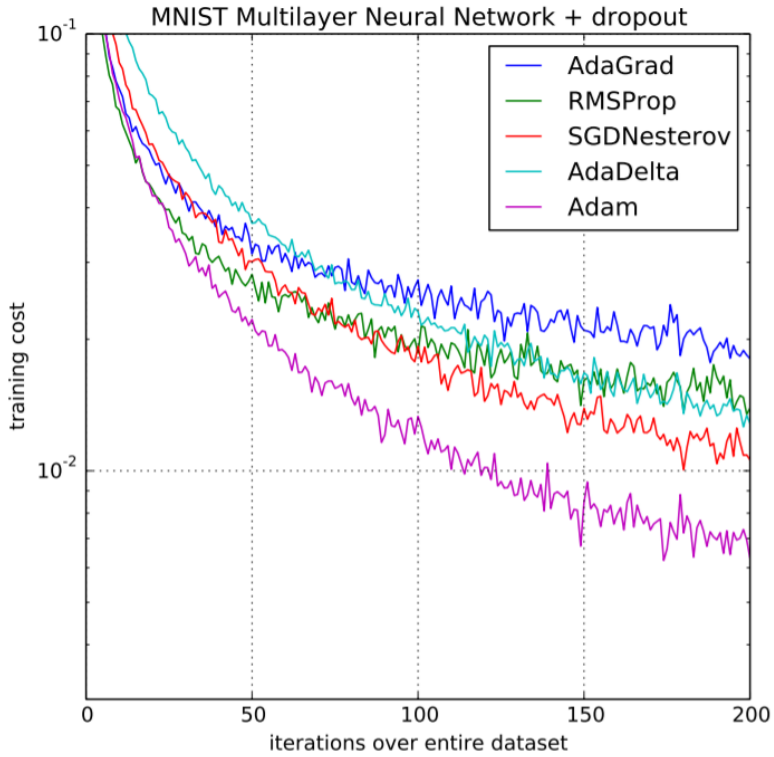

Figure 4.8: Comparison of Adam to other optimization algorithms (AdaGrad, RMSProp, SGDNesterov, AdaDelta) training a multilayer perceptron [4].

Regularization

At first, the two fully-connected neural networks and the two convolutional neural networks have been tested with different regularisation techniques including Dropout, L1, L2 regularization. Regularization basically adds a penalty as model complexity increases and this is how L1 and L2 work. When computing the new weight vectors, L1 regularization penalizes the absolute value of the weight and this implies that extreme values, both positive and negative, are penalized. L1 Regularization induces sparsity by driving weights to zero and consequently removing the corresponding features from the model [60]. L2 Regularization, also called weight decay, penalizes the square of the weight. L2 Regularization has a smaller impact on less extreme weights than on those with high values. It can be interpreted intuitively as a reduction of the weights by a certain percentage each time. Dropout on the other hand, is a regularization method that approximates training a large number of neural networks with different architectures. In particular, during the training phase, some number of layer outputs are randomly ignored (dropped).

Although regularisation is proven effective for preventing overfitting, the classification accuracy of our models did not further improve. Overfitting may occur when large neural networks are trained on relatively small data sets, resulting in the model learning the statistical noise of the training set, leading to poor performance when the model is evaluated on the test set (i.e., on new samples). The neural networks are trained on dataset of millions of samples and they are able to generalize very well on them, this implies that the overfitting problem doesn't influence our models on this task.

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 [50] 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 5V at 2.5A.

On the other hand, to run our chosen models into an FPGA the pipeline follows a different way. As previously stated in Chapter 3, with the help of the framework FINN [53] we are able to run neural networks into an FPGA. We tested the quantized neural network (previously called Quant-NN) on the PYNQ-Z2 FPGA board. Fig. 4.9 shows how FINN interprets the quantized neural network during its workflow. PYNQ is an open-source project designed by Xilinx [52] which makes it easier to use Xilinx platforms providing Jupyter-based framework with

Python API. PYNQ-Z2 features 512MB DDR3 of RAM and it is equipped with a 650MHz dual-core Cortex-A9 processor.

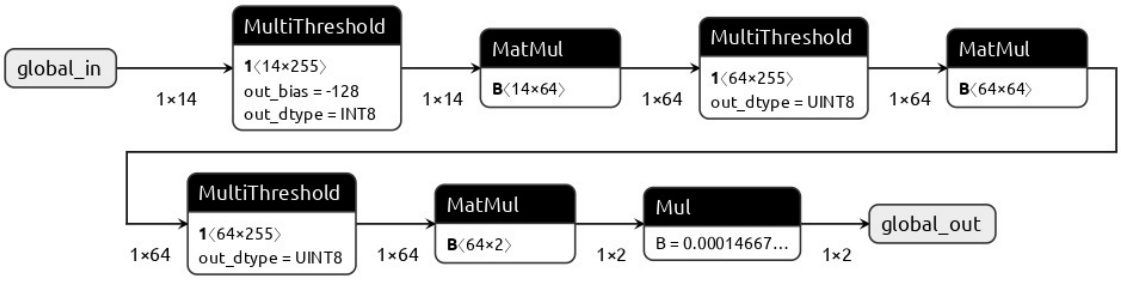

Figure 4.9: Trained model block representation which depicts the different layers of the Quantized Neural Network used during FINN workflow.

4.6 Evaluation Metrics

The chosen ML models are evaluated with different metrics. Accuracy is defined as the total number of correctly classified instances over the total number of instances in the dataset. The most descriptive metrics are accuracy, precision, recall and f1-score. Specifically, the precision represents the ability of a classifier not to label as positive a sample that is negative. It is a good measure to state when the costs of false positive is high. Recall (or True Positive Rate - TPR), is intuitively the ability of the classifier to recognize all the positive samples. When there is an high cost associated with False Negative the recall should be the metric to use to select the best model as in this task the False Positives represents false alarms, while False Negatives are missed attacks. The F1-score, on the other hand, can be interpreted as a harmonic mean of the precision and recall. F1-score is a function of Precision and Recall which they give a equal relative contribution and it is used to seek a balance between the precision and recall metrics. Another important performance measurement consists on the plot of TPR against the FPR, which depicts the Receiver Operating Characteristics (ROC) curve. ROC is a probability curve and the Area Under the Curve (AUC) identifies the degree of separability which tells how good is the model on distinguishing between different classes. The ROC curve is plotted with TPR against the FPR and an example of the plot can be seen in Figure 4.10. We also evaluate our models under how many samples they can process per second.

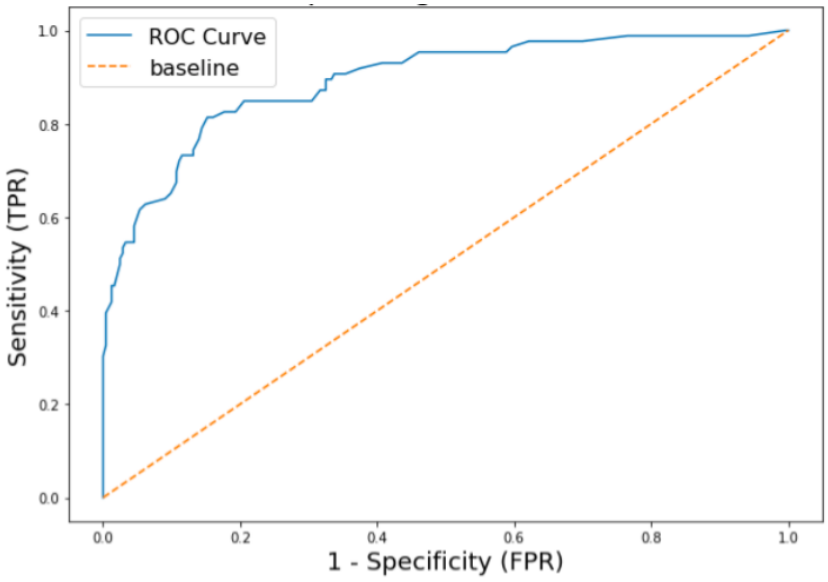

Figure 4.10: The ROC curve is plotted on a graph with the True Positive Rate (or Sensitivity) on the y axis and False Positive Rate (or 1 - Specificity) on the x axis.

The functions of the key metrics presented above are shown in Tab. 4.4 where TP, TN, FN and FP are following explained and they are graphically represented in the confusion matrix in Figure 4.11.

- *True-Positive (TP)*: number of samples that are in the normal class in the dataset and are correctly predicted in the benign class.
- *True-Negative (TN)*: number of samples that are in the malicious class in the dataset and are correctly predicted in the malicious class.
- *False-Negative (FN)*: number of samples that are in the normal class in the dataset and are incorrectly predicted in the malicious class.
- *False-Positive(FP)*: number of samples that are in the malicious class in the dataset and are incorrectly predicted in the normal class.

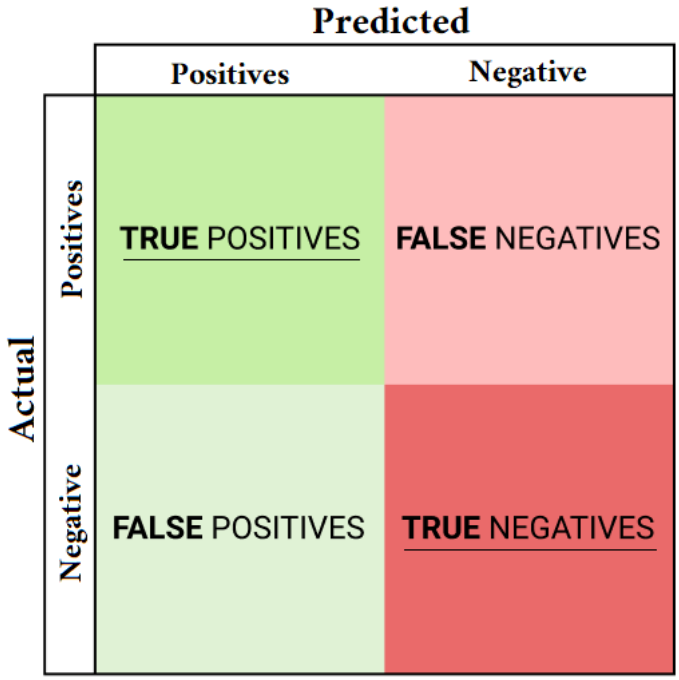

Figure 4.11: Confusion matrix structure.

On the other hand, the evaluation metrics for the FPGA board assessment are mainly related to the electronic components. Specifically we are interested in the resource utilization of the following components:

- *LookUp Table*
- *LUT-RAM*

Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall (TPR)	$\frac{TP}{TP+FN}$
False Positive Rate (FPR)	$\frac{FP}{FP+TN}$
F-1 Score	$\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$

Table 4.4: Key evaluation metrics for the experiments.

- *Flip-Flop*
- *Block RAM*
- *IO*
- *Global Buffer*

LookUp Table (LUT) consists of a block of SRAM and it is basically a table which determines what will be the output for any given input. LUT in resource utilization refers to all the LUTs used in the design. The LUT in the SLICEM utilized in the design used as distributed RAM/ROM or shift register is referred as *LUTRAM* [61]. While the logic is created in the LUTs Flip-flops are used to create registers, which store data. *BUFG* is the global buffer and *BRAM* stands for Block Random Access Memory which are used for storing large amounts of data inside of your FPGA. *IO* refers to Input/Output capacity.

The comparison of different Machine Learning models has always been a challenging task [62]. An ML algorithm may obtain very good performance over one dataset whereas may show poor results over another similar dataset and the reasons for this can be many. Therefore, it turned out to be difficult to find a solution in some cases, especially when using deep learning models with a very complex architecture. However, in general, this fact mostly depends on the type of the features and the characteristics of the machine learning algorithms. Consequently, it is hard to decide with confidence which model is best for a given problem, mostly due to the fact that in this case we add the lightness of the model as a determining factor. We therefore use several datasets for a better confidence and we take into account the fact that all models can be easily interchanged in the final intrusion detection system.

5

Results

In this section, we carry out a detailed performance analysis and we discuss the results of Neural Network and tree-based ensemble classifiers described in the previous paragraph. Benchmarking of all the models is performed on TON and IOT23 datasets. The results obtained are compared and analyzed with the aim of assessing which are the most suitable classifiers for IDS system in IoT environment and document models' behaviour with recent IoT traffic datasets. The Machine Learning models are trained to recognize malicious packets in a supervised learning way. The main reason for choosing supervised learning is that the traffic characteristics can be effectively used to train ML models for further predictions even if the other ML approach like unsupervised learning can also be used to perform a similar task like clustering. Besides accuracy, which is one of the most significant metrics, an important requirement for machine learning models in real world information security applications is to have an extremely low false positive rate, as we do not want the normal operation of an IoT network to be interrupted/alarmed by false positives.

We start by evaluating the results for the TON and IOT23 datasets by comparing tree-based and neural network-based models in two separate groups. Table 5.1 and 5.2 show the results obtained in terms of key metrics for the TON dataset and similarly Table 5.3 and 5.4 for IOT-23. Among the various evaluation metrics, the time it takes

a classifier to process samples per second was also measured. This measure is useful for determining a model's ability to smoothly handle large amounts of incoming data, which is useful for quickly recognizing attacks such as DoS and DDoS. The 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 in the Raspberry Pi itself, as being lighter than neural networks, 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 Pi's), with the goal of improving performance over time and user privacy, and without requiring users to update the device software. The following results refer to the experiments carried out on the Raspberry Pi 3B. Next, the results of the tests carried out by implementing a neural network in the FPGA are exhibited.

Results Table 1: TON Dataset - Neural Networks

Models	Accuracy	Precision	Recall	F1-score	ROC-AUC score	Samples/Second
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 5.1: Results of neural networks models and TabNet for TON Dataset.

Results Table 2: TON Dataset - DT & Ensamble Models

Models	N° Est.	Acc.	Prec.	Recall	F1	ROC-AUC	Samples/s	Train [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
AB	20	0.9989	0.9986	0.9985	0.9985	0.9988	41,893	110.25
AB	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

Table 5.2: Results of Decision Tree and ensemble models for TON Dataset. The number of estimators is also shown as the training time.

Table 5.1 shows the value of all relevant metrics concerning neural networks models previously described on TON dataset. *CNN_small* and *NN_small* refer to a "smaller" version of *CNN* and *NN* respectively. As mentioned in the previous chapter, we ran several tests with different sized networks to highlight the performance difference in the Raspberry, which is a resource-limited device. It is possible to notice from the results tables, how the performance changes according to the size of the neural network. In particular, more than the accuracy and precision, the number of samples processed per second is the parameter with the greatest oscillation; except for *CNN_small* which achieves rather below average results even if it is slightly faster in processing compared to other neural networks. This value is of particular interest to us since, having to deal with devices with limited resources, if you use a model that is too heavy, it would not be able to process all the incoming traffic, having to discard the packets.

It is evident from these results that CNNs do not perform as well as the two feed-forward NN and TabNet, which achieve almost the same accuracy (98.9%, 98.2% and 98.6% respectively). *NN_small* obtains the best accuracy, recall, F1-score and ROC-AUC score among the neural networks models. This is probably due to the fact that the complexity introduced by a greater number of neurons is not essential for obtaining better results with this kind of features. The worst performance in terms of key metrics is obtained by *small_CNN*, even if it achieves the highest value of samples processed per second.

On the other hand, it is evidenced by Table 5.2 that tree-based models perform enormously better than neural networks using this dataset on classifying malicious samples. Furthermore, it is observed that Decision Tree outperforms other classifiers in terms of execution speed and overall performance. DT obtained almost perfect classification accuracy (99.99%) and F1-score (99.99%) and at the same time it is capable of processing roughly 2.3 millions of samples per second which is almost 20 times more than Random Forest, that on his side, obtains excellent results, and 1000 times faster than the fastest neural network tested (i.e.: *CNN_small*). Tree-based ensemble models perform very similarly, except for AdaBoost which is slightly worse with respect to the others.

Furthermore, a comparison is made between the different ensembles in terms of number of estimators to show the difference in classification speed and training speed. DT also had the best performance for these latter metrics. In fact, this model is extremely suitable for classifying network traffic with the pre-processing used. Random Forest, Adaboost, and Extra Trees, on the other hand, perform slightly worse, and it is possible to see a noticeable difference in the processing speed of the samples. Regarding TON dataset, the model that prevails in each field is Decision Tree while both CNNs get the worst results. It is thus inferred that 1D convolution applied in this context of tabular network traffic does not obtain the same results as feed-forward NN and tree-based models which are entirely suitable for this task.

Results Table 3: IOT-23 Dataset - Neural Networks

Models	Accuracy	Precision	Recall	F1-score	ROC-AUC score	Samples/Second
NN small	0.9977	0.9954	0.9998	0.9976	0.9977	1,568
NN	0.9984	0.9969	9.9997	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 5.3: Results of neural networks models and TabNet for IoT-23 Dataset.

Results Table 4: IOT-23 Dataset - DT & Ensamble Models

Models	N° Est.	Acc.	Prec.	Recall	F1	ROC-AUC	Samples/s	Train [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
AB	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

Table 5.4: Results of Decision Tree and ensemble models for IoT-23 Dataset. The number of estimators is also shown as the training time.

Regarding the IoT-23 dataset, similar to the TON results, the largest Neural Network is the best performing among the neural network-based models with an accuracy of 99.84% as shown in Table 5.3. This models achieves the best F1-score and ROC-AUC score among the neural networks. In general, on IoT all the models obtain great results and this fact implies that the scenarios selected and used to test the different classifiers, the malicious packets are more easily recognizable than those of TON, even if the difference does not correspond to a large order of magnitude. The results of neural networks in Table 5.3 and Table 5.5, except for TabNet, are conducted with 10% of the training set. That is 298, 695 samples for time reasons, since these models are rather slow compared to tree-based models and the amount of training samples is sufficient for those models to perform as if they were trained with the entire training set.

Table 5.4 presents the results of tree-based models. Decision Tree and Random Forest are the best performing models, obtaining results which are practically perfect with an accuracy, precision, recall, F1-score and AUC of at least 99.99%. It thus translates that the packets in the IOT23 dataset are more easily classified than the traffic captured by TON, even for tree-based classifiers. Even in this scenario the least performing model is the smaller CNN while DT and the ensamble models obtained the the best results.

With Table 5.5 and Table 5.6 we want to give a term of comparison with the training time and the testing time on the Raspberry. The Raspberry Pi 3B has a CPU while the PC where we conducted these tests is equipped with a CPU.

Results Table 5: training times and testing times on the PC on TON

Models	CNN_small	CNN	NN_small	NN	TabNet
Training Time [s]	160	203	160	438	310
Samples/s	43,721	38,727	44,097	33,781	42,561

Table 5.5: Training times and samples per second of neural networks models performed on the local machine on TON dataset. All the models are tested with the parameters presented in the previous chapter.

Results Table 6: training times and testing times on the PC for TON

Models	DT	RF	AdaBoost	ET
Training Time [s]	1.1	6.3	7	4.3
Samples/s	17,196,522	823,905	586,933	691,297

Table 5.6: Training times and samples per second of tree-based models performed on the local machine on TON dataset. All the tree-based models are tested with the parameters exposed in the previous chapter and 20 estimators.

Results Table 7: training times and testing times on the PC for IoT-23

Models	CNN_small	CNN	NN_small	NN	TabNet
Training Time [s]	130	166	136	401	1,798
Samples/s	45,493	42,202	47,037	29,836	159,326

Table 5.7: Training times and samples per second of neural networks models performed on the local machine on IoT-23 dataset. All the models are tested with the parameters presented in the previous chapter.

Results Table 8: training times and testing times on the PC for IoT-23

Models	DT	RF	AdaBoost	ET
Training Time [s]	8.8	88	67	32
Samples/s	38,436,350	3,269,013	1,793,039	2,394,340

Table 5.8: Training times and samples per second of tree-based models performed on the local machine on IoT-23 dataset. All the tree-based models are tested with the parameters exposed in the previous chapter and 20 estimators.

Subsequently, Fig. 5.1 and Fig. 5.2 graphically show the results obtained for both TON dataset under accuracy, F1-score and AUC metrics. Likewise Fig. 5.3 and Fig. 5.4 for IoT-23 dataset. The results suggest that the encoded belief over hypothesis carried by neural network depth 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.

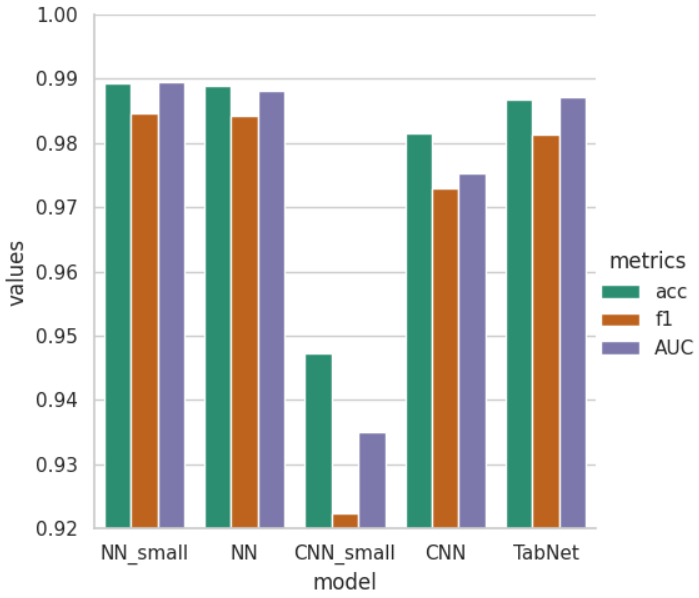

Figure 5.1: Comparison of the statistical results of the models on TON Dataset. The chart includes neural network based models. The considered metrics are accuracy, f1-score and AUC-ROC measure.

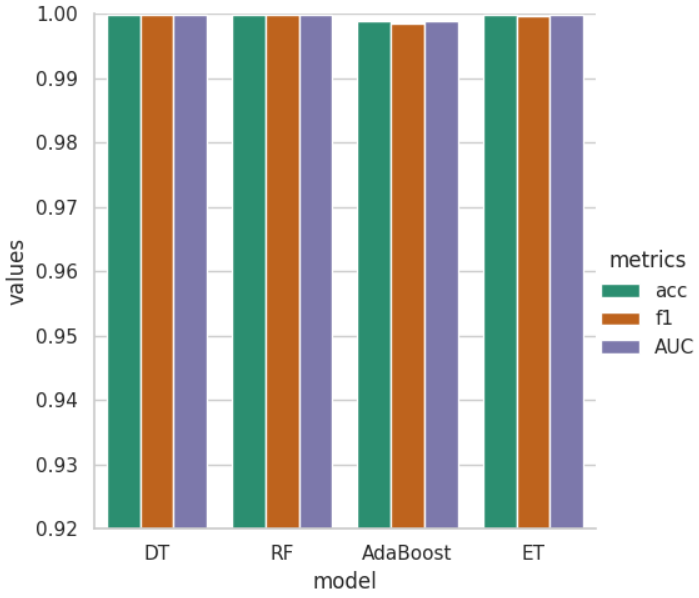

Figure 5.2: Comparison of the statistical results of the models on TON Dataset. The considered metrics are accuracy, f1-score and AUC-ROC measure. It is clear by the plot that the tree-based models perform much better than the others.

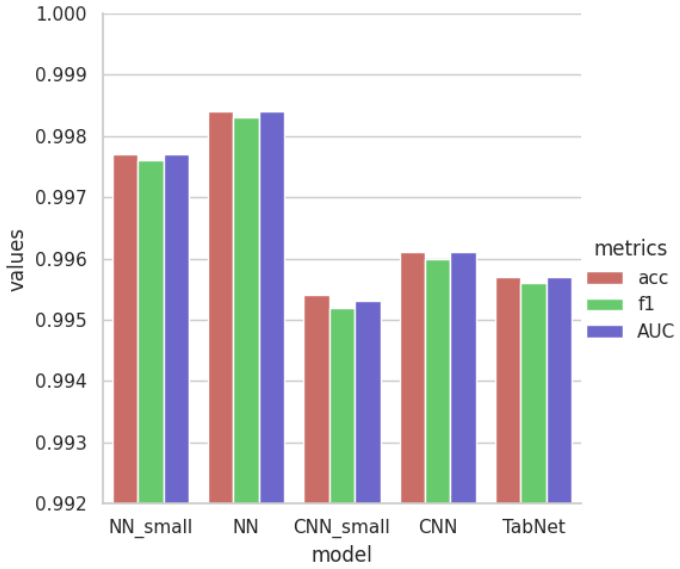

Figure 5.3: Comparison of the statistical results of the models on IOT-23 Dataset. The chart includes neural network based models. The considered metrics are accuracy, f1-score and AUC-ROC measure.

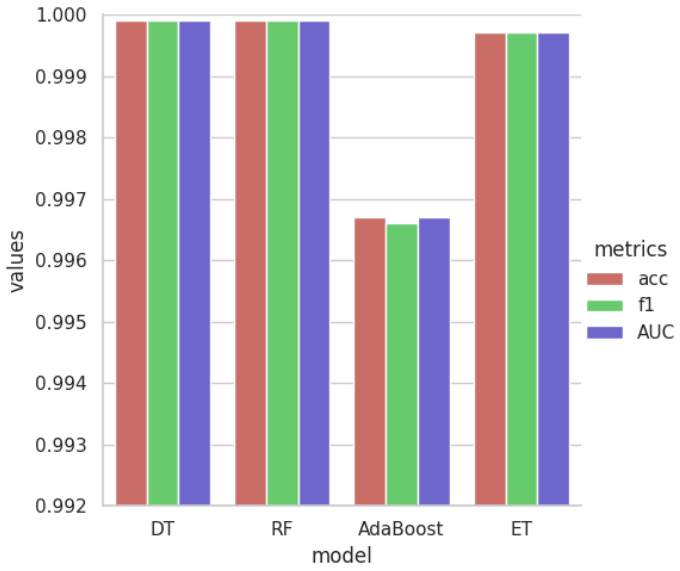

Figure 5.4: Comparison of the statistical results of the tree-based models on IOT-23 Dataset. The considered metrics are accuracy, f1-score and AUC-ROC measure. The chart show the better performance of tree-based models even in this scenario.

Table 5.9 shows the results of the Quantized Neural Networks trained locally and then implemented on the PYNQ-Z2 FPGA board. As can be seen from Table 5.10, 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 device 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 turns out to be less probably due to software overhead. It is possible to notice that the utilization of LUT, FF and BRAM is quite low, and combined with a clock frequency of 100 MHz achieves extremely low latency. This translates in a generally good accuracy for an 8-bit quantized model, which is 97.47% and 99.81% for TON and IOT23, respectively.

Results Table 9: Quantized Neural Network first tested on PC

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

Table 5.9: Results of the quantized neural network for TON and IoT-23 dataset.

It is observed that DT outperforms other classifiers in terms of any key metrics, as well as RF and ET, for both TON and IOT23. We can firmly state that tree-based models obtain the best results as they are they are the best candidates for the IDS system. These results are probably due to the tabular nature of the network traffic alongside the pre-processing applied to the chosen features which favors tree-based algorithms. Furthermore the average number of samples per second processed by these models is significantly higher than the other classifiers considered. The experimental tests demonstrate how the chosen lightweight classifiers obtained 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 Pi 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 and IOT23, are suitable choices for this purpose and it can be seen from the exposed results that the chosen models show promising performance.

Results Table 10: FPGA PYNQ-Z2 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%
IO	46	125	36.80%
BUFG	1	32	3.13%
Bandwidth (simulation) [*samples*/*s*]		1438,848	
Latency [μs]		2.56	
Frequency [MHz]		100	
Power est. [W]		0.503	
Bandwidth (PYNQ) [*samples*/*s*]		717,529	

Table 5.10: Performance metrics obtained on PYNQ-Z2 for both simulation and physical test.

6

Conclusion

In this work, a study on anomaly-based intrusion detection system suitable for securing IoT environment against malicious attacks is carried out. We documented the performance assessment of different machine learning classification algorithms including neural networks, decision tree, random forests, adaboost, extremely randomized trees and TabNet. All the classifiers are bench-marked on recent IoT datasets containing different kind of attacks, TON and IOT-23. The optimal parameters of the models are obtained using a grid search algorithm combined with a study on the features 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 in a Raspberry Pi 3B and a PYNQ-Z2 FPGA board and from the performance results it is concluded that DT, RF, and ET show the best trade-of between prominent metrics and processing time. We showed how these models achieve extremely good results on classifying network traffic processed as tabular data. They are therefore a suitable choice for building IoT specific IDS. Based on these assessments, we propose a network-based intrusion detection system that consists of attaching such light-weight board to the single components of an IoT network. We demonstrate that the combination of RasPI or FPGA with ML classifiers

is capable of achieving great results in terms of detection accuracy and response time. By deploying such IDS, the performance of the single IoT devices is not affected, and due to the high flexibility the IDS devices can be hybridly-placed in the IoT environment introducing an additional layer of security against malicious attacks. 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 more 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.

As future work the IDS design can be implemented using more complex ML models by testing its capabilities in even lighter low-powered devices. Moreover, the evaluation of such IDS with different datasets and the extension to Incremental Learning could open the door to the IDSs of the future.

References

[1] S. B. Mathieu Guillame-Bert and J. P. Josh Gordon, "Introducing tensorflow decision forests." [Online]. Available: https://blog.tensorflow.org/2021/05/introducing-tensorflow-decision-forests.html

[2] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," *CoRR*, vol. abs/1908.07442, 2019. [Online]. Available: http://arxiv.org/abs/1908.07442

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[5] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets," 05 2021.

[6] J. Manyika, M. Chui, P. Bisson, R. D. Jonathan Woetzel, J. Bughin, and D. Aharon, "Unlocking the potential of the internet of things," 2015. [Online]. Available: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world

[7] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "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*, vol. 8, pp. 23 022–23 040, 2020.

[8] M. Ammar, G. Russello, and B. Crispo, "Internet of things: A survey on the security of iot frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212617302934

[9] M. binti Mohamad Noor and W. H. Hassan, "Current research on internet of things (iot) security: A survey," *Computer Networks*, vol. 148, pp. 283–294, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618307035

[10] N. Woolf, "Ddos attack that disrupted internet was largest of its kind in history, experts say," 2016. [Online]. Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

[11] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[12] L. B. William Stallings, *Computer Security: Principles and Practice*. Pearson, 2014.

[13] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[14] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aaa8415

[15] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-low-power Microcontrollers*. O'Reilly, 2020. [Online]. Available: https://books.google.be/books?id=sB3mxQEACAAJ

[16] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," 2017. [Online]. Available: https://arxiv.org/abs/1705.06805

[17] T.-H. Lee, C.-H. Wen, L.-H. Chang, H.-S. Chiang, and M.-C. Hsieh, "A lightweight intrusion detection scheme based on energy consumption analysis in 6lowpan," in *Advanced technologies, embedded and multimedia for human-centric computing*. Springer, 2014, pp. 1205–1213.

[18] F. Hosseinpour, P. Vahdani Amoli, J. Plosila, T. Hämäläinen, and H. Tenhunen, "An intrusion detection system for fog computing and iot based logistic systems using a smart data approach," *International Journal of Digital Content Technology and its Applications*, vol. 10, no. 5, 2016.

[19] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of iot networks using artificial neural network intrusion detection system," in *2016 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2016, pp. 1–6.

[20] M. Nobakht, V. Sivaraman, and R. Boreli, "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)*. IEEE, 2016, pp. 147–156.

[21] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot," *Sensors*, vol. 17, no. 9, p. 1967, 2017.

[22] B. Jia, X. Huang, R. Liu, and Y. Ma, "A ddos attack detection method based on hybrid heterogeneous multiclassifier ensemble learning," *Journal of Electrical and Computer Engineering*, vol. 2017, 2017.

[23] R. Kozik, M. Choraś, M. Ficco, and F. Palmieri, "A scalable distributed machine learning approach for attack detection in edge computing environments," *Journal of Parallel and Distributed Computing*, vol. 119, pp. 18–26, 2018.

[24] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 29–35.

[25] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep learning-based intrusion detection for iot networks," in *2019 IEEE 24th pacific rim international symposium on dependable computing (PRDC)*. IEEE, 2019, pp. 256–25 609.

[26] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in iot sensors in iot sites using machine learning approaches," *Internet of Things*, vol. 7, p. 100059, 2019.

[27] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for iot intrusion detection system," *Simulation Modelling Practice and Theory*, vol. 101, p. 102031, 2020.

[28] A. Y. Hussein, P. Falcarin, and A. T. Sadiq, "Iot intrusion detection using modified random forest based on double feature selection methods," in *International Conference on Emerging Technology Trends in Internet of Things and Computing*. Springer, 2022, pp. 61–78.

[29] M. Sarhan, W. W. Lo, S. Layeghy, and M. Portmann, "Hbfl: A hierarchical blockchain-based federated learning framework for a collaborative iot intrusion detection," *arXiv preprint arXiv:2204.04254*, 2022.

[30] A. Le, J. Loo, K. K. Chai, and M. Aiash, "A specification-based ids for detecting attacks on rpl-based network topology," *Information*, vol. 7, no. 2, p. 25, 2016.

[31] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. A. Spirito, "An ids framework for internet of things empowered by 6lowpan," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 1337–1340.

[32] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits, "Denial-of-service detection in 6lowpan based internet of things," in *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*. IEEE, 2013, pp. 600–607.

[33] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[34] C. Jun and C. Chi, "Design of complex event-processing ids in internet of things," in *2014 sixth international conference on measuring technology and mechatronics automation*. IEEE, 2014, pp. 226–229.

[35] D. Oh, D. Kim, and W. W. Ro, "A malicious pattern detection engine for embedded security systems in the internet of things," *Sensors*, vol. 14, no. 12, pp. 24 188–24 211, 2014.

[36] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, "Kalis—a system for knowledge-driven adaptable intrusion detection for the internet of things," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 656–666.

[37] E. Anthi, L. Williams, and P. Burnap, "Pulse: An adaptive intrusion detection for the internet of things," in *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, 2018, pp. 1–4.

[38] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet of things Journal*, vol. 6, no. 2, pp. 1606–1616, 2018.

[39] H. Liu, C. Li, X. Jin, J. Li, Y. Zhang, and D. Gu, "Smart solution, poor protection: An empirical study of security and privacy issues in developing and deploying smart home devices," in *Proceedings of the 2017 Workshop on Internet of Things security and privacy*, 2017, pp. 13–18.

[40] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[41] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *expert systems with applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[42] R. G. Bace, P. Mell *et al.*, "Intrusion detection systems," 2001.

[43] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844018332067

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[45] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[46] P. Aznar, "Decision trees: Gini vs entropy - quantdare," Dec 2020. [Online]. Available: https://quantdare.com/decision-trees-gini-vs-entropy/

[47] Mathworks, "Incremental learning overview," 2022. [Online]. Available: https://www.mathworks.com/help/stats/incremental-learning-overview.html

[48] T. Yiu, "Understanding random forest," 2019. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[49] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[50] R. P. Foundation, "Raspberry pi," 2022. [Online]. Available: https://www.raspberrypi.com

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[52] I. Advanced Micro Devices, "Amd xilinx," 2022. [Online]. Available: https://www.xilinx.com

[53] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, and K. Vissers, "Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, pp. 1–23, 2018.

[54] A. Pappalardo, "Xilinx/brevitas," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.3333552

[55] T. Z. Project, "The zeek network security monitor," https://zeek.org/, 2020.

[56] Wireshark, "Wireshark," https://www.wireshark.org/.

[57] S. Garcia, A. Parmisano, and M. J. Erquiaga, "Iot-23: A labeled dataset with malicious and benign iot network traffic," Jan. 2020, more details here https://www.stratosphereips.org/datasets-iot23. [Online]. Available: https://doi.org/10.5281/zenodo.4743746

[58] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105524, 2020.

[59] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[60] R. Khandelwal, "L1 and l2 regularization," 2018. [Online]. Available: https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2

[61] F. www.fpgakey.com, "Fpga wiki encyclopedia," accessed: 2022-06-15. [Online]. Available: https://www.fpgakey.com/wiki/

[62] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

Acknowledgments

This thesis work is dedicated to my parents who have always been a constant source of support, love and encouragement during the challenges of my career and my life. You allowed me to study and made my achievements possible. I am truly thankful for having you in my life and for everything you have done for me. This work is also dedicated to my relatives and friends, who have always believed in me and whose good example has taught me to work hard for the things I aspire to achieve. Special thanks go to my supervisors and the ES&S research team at the University of KULeuven, I think we did a great job together.