

UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Andressa Amaral Cunha

**Machine Learning Architectures for Early Detection and
Classification of Botnet Attack in the Internet of Things**

Belo Horizonte
2023

Andressa Amaral Cunha

**Machine Learning Architectures for Early Detection and
Classification of Botnet Attack in the Internet of Things**

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Antonio Alfredo Ferreira Loureiro

Co-Advisor: João Batista Borges Neto

Belo Horizonte
2023

Cunha, Andressa Amaral

C972q Machine learning architectures for early detection and classification of botnet attack in the internet of things [manuscrito] / Andressa Amaral Cunha— 2023.
122 f. il.; 29 cm.

Orientador: Antonio Alfredo Ferreira Loureiro.
Coorientador: João Batista Borges Neto
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Departamento de Ciência da Computação, Instituto de Ciências Exatas
Referências: f.93-101.

1. Computação – Teses. 2. Internet das Coisas – Teses. 3. Detecção de anomalias (Computação) – Teses. 4. Computação de borda– Teses. I. Loureiro, Antonio Alfredo Ferreira. II. João Batista Borges Neto Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. III. Título.

CDU 519.6*82 (043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

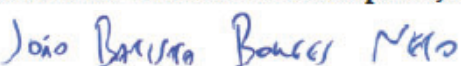
FOLHA DE APROVAÇÃO

Machine Learning Architectures for Early Detection and Classification of
Botnet Attack in the Internet of Things

ANDRESSA AMARAL CUNHA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:


PROF. ANTONIO ALFREDO FERREIRA LOUREIRO - Orientador
Departamento de Ciência da Computação - UFMG


PROF. JOÃO BATISTA BORGES NETO - Coorientador
Departamento de Computação e Tecnologia - UFRN


PROF. HEITOR SOARES RAMOS FILHO
Departamento de Ciência da Computação - UFMG


PROF. LEANDRO APARECIDO VILLAS
Instituto de Computação - UNICAMP

Belo Horizonte, 24 de fevereiro de 2023.

Resumo

A Internet das Coisas (IoT, do inglês *Internet of Things*) é uma área que trata de sistemas pervasivos, conectados por padrões de comunicação como Bluetooth, WiFi e 5G. Para poder prover serviços com baixa latência, a computação de borda emerge como um paradigma que permite realizar o processamento de dados mais próximo da camada de sensores, onde a informação do ambiente é obtida. Aplicações IoT recentes requerem soluções em tempo real, com a segurança das informações e dos dispositivos garantida, prezando pelo correto gerenciamento dos dados e o desenvolvimento de sistemas energeticamente eficientes. Aplicar técnicas de aprendizado de máquina em dispositivos IoT é uma possível solução que pode auxiliar com requisitos como a governança dos dados (segurança, confiabilidade e usabilidade), balanceamento do processamento e a tomada de decisão, como detecção precoce de ataques vindos da rede, reconhecimento de objetos, ou autoaperfeiçoamento dos sistemas IoT. Para auxiliar com esses requisitos, a computação de borda também tem sido muito utilizada com a finalidade de criar sistemas em tempo real e com eficiência energética. O objetivo deste trabalho é avaliar o uso de aprendizado de máquina em ambientes IoT e de computação de borda, com foco na análise do requisito de segurança das aplicações e dispositivos. Para isso, propõe-se dois experimentos empíricos com relação ao requisito de segurança, conduzindo um estudo sobre ataque de botnets em dispositivos IoT e possíveis estratégias de protegê-los contra esses tipos de ameaças.

Palavras-chave: internet das coisas; detecção de ataques; aprendizado de máquina; aprendizado profundo; computação de borda

Abstract

Internet of Things (IoT) is an area that deals with pervasive systems, connected by communication standards such as Bluetooth, WiFi and 5G. To provide low-latency services, edge computing emerges as a paradigm that allows processing data closer to the sensing layer, where information is acquired from environment. Recent IoT applications require real-time solutions, with the security of data and devices guaranteed, focusing on correct data management and the development of energy efficient systems. Applying machine learning techniques to IoT devices is a possible solution that can help with requirements such as data governance (security, reliability and usability), balancing processing and decision making, such as early detection of attacks coming from the network, recognition of objects, or self-improvement of IoT systems. To assist with these requirements, edge computing has also been widely used for the purpose of creating real-time and energy-efficient systems. The goal of this work is to evaluate the use of machine learning in IoT and edge computing environments, focusing on the analysis of the security requirement of applications and devices. For this, two empirical experiments are proposed regarding the security requirement, conducting a study on botnet attacks on IoT devices and possible strategies to protect them against these types of threats.

Keywords: internet of things; attack detection; machine learning; deep learning; edge computing

List of Figures

1.1	Concise representation of consolidated architecture for IoT systems. A complete model is presented in Chapter 3.	14
3.1	Consolidated architecture of Internet of Things systems.	35
3.2	Another modeled architecture, now considering the paradigm of edge computing. Edge, or cloudlet nodes [2.1], are able to communicate with management layer through network tier [2.2], or can directly send response or information to application layer [4], depending on the design and the purpose of the structure.	36
4.1	The structure of a neural network with its nodes, layers and weight values that transform data while it passes through the steps of the network [Zaki and Meira, 2020; Gron, 2019].	49
5.1	Basic architecture of a Variational Autoencoder.	61
5.2	Proposed methodology to execute the attack detection experiments on both datasets.	63
5.3	Confusion matrix of the attack detection on Danmini Doorbell model.	67
5.4	Confusion matrix of the attack detection with N-BaIoT's unique model.	68
5.5	Confusion matrix of the attack detection with Bot-IoT model.	68
5.6	Accuracy results of Danmini Doorbell models on the attack detection task with dynamic feature space.	69
5.7	Accuracy results of unique models on the attack detection task with dynamic feature space.	70
6.1	Basic architecture of the proposed Convolutional Neural Network.	77
6.2	Proposed methodology to execute the attack classification experiments on both datasets.	80
6.3	Confusion matrix of the attack classification on Danmini Doorbell with CNN, KNN and NB models.	83
6.4	Confusion matrix of the attack classification on Bot-IoT with CNN, KNN and NB models.	84
6.5	Accuracy results for Danmini Doorbell on the attack classification task with CNN, KNN and NB models.	85
6.6	Accuracy results for the Bot-IoT dataset on the attack classification task with CNN, KNN and NB models.	85

A.1	Confusion matrix of the attack detection on Ecobee Thermostat model.	102
A.2	Confusion matrix of the attack detection on Ennio Doorbell model.	103
A.3	Confusion matrix of the attack detection on Phillips Baby Monitor model. . .	103
A.4	Confusion matrix of the attack detection on P737 Security Camera model. . .	104
A.5	Confusion matrix of the attack detection on P828 Security Camera model. . .	104
A.6	Confusion matrix of the attack detection on S1002 Security Camera model. . .	105
A.7	Confusion matrix of the attack detection on S1003 Security Camera model. . .	105
A.8	Confusion matrix of the attack detection on Samsung Webcam model.	106
A.9	Accuracy results of Ecobee Thermostat models on the attack detection task with dynamic feature space.	106
A.10	Accuracy results of Ennio Doorbell models on the attack detection task with dynamic feature space.	107
A.11	Accuracy results of Phillips Baby Monitor models on the attack detection task with dynamic feature space.	107
A.12	Accuracy results of Provision 737 Security Camera models on the attack detection task with dynamic feature space.	108
A.13	Accuracy results of Provision 838 Security Camera models on the attack detection task with dynamic feature space.	108
A.14	Accuracy results of SimpleHome 1002 Security Camera models on the attack detection task with dynamic feature space.	109
A.15	Accuracy results of SimpleHome 1003 Security Camera models on the attack detection task with dynamic feature space.	109
A.16	Accuracy results of Samsung Webcam models on the attack detection task with dynamic feature space.	110
B.1	Confusion matrix of the attack classification on Ennio Doorbell with CNN, KNN and NB models.	111
B.2	Confusion matrix of the attack classification on Ecobee Thermostat with CNN, KNN and NB models.	112
B.3	Confusion matrix of the attack classification on Phillips Baby Monitor with CNN, KNN and NB models.	113
B.4	Confusion matrix of the attack classification on P737 Security Camera with CNN, KNN and NB models.	114
B.5	Confusion matrix of the attack classification on P838 Security Camera with CNN, KNN and NB models.	115
B.6	Confusion matrix of the attack classification on S1002 Security Camera with CNN, KNN and NB models.	116
B.7	Confusion matrix of the attack classification on S1003 Security Camera with CNN, KNN and NB models.	117

B.8	Confusion matrix of the attack classification on Samsung Webcam with CNN, KNN and NB models.	118
B.9	Accuracy results for Ecobee Thermostat on the attack classification task with CNN, KNN and NB models.	119
B.10	Accuracy results for Ennio Doorbell on the attack classification task with CNN, KNN and NB models.	119
B.11	Accuracy results for Phillips Baby Monitor on the attack classification task with CNN, KNN and NB models.	120
B.12	Accuracy results for Provision 737 Security Camera on the attack classification task with CNN, KNN and NB models.	120
B.13	Accuracy results for Provision 838 Security Camera on the attack classification task with CNN, KNN and NB models.	121
B.14	Accuracy results for SimpleHome 1002 Security Camera on the attack classification task with CNN, KNN and NB models.	121
B.15	Accuracy results for SimpleHome 1003 Security Camera on the attack classification task with CNN, KNN and NB models.	122
B.16	Accuracy results for Samsung Webcam on the attack classification task with CNN, KNN and NB models.	122

List of Tables

2.1	List of works that provide researches about IoT and edge computing architectures and data management efforts for IoT solutions.	28
2.2	Summarizing main points treated on studied researches about machine learning models applied in IoT and edge solutions.	29
2.3	Summarizing works that treats about security for IoT environments, specifically for botnet attack detection. The table shows the main points addressed in the papers.	31
3.1	Comparison between some microcontrollers (MCUs) and microcomputers applied in IoT systems. The available resources should be carefully considered when designing the desired prototype. Raspberry is the microcomputer with best resources, and, indeed, is employed in many works, but is also the most expensive one.	40
3.2	Description of each particular statistical feature on the N-BaIoT dataset, according to the works of Meidan et al. [2018], Mirsky et al. [2018] and Alqahtani et al. [2020].	45
4.1	Ordinary classification of types of machine learning algorithms, as well as its technical features, pros, cons and example techniques. The given examples are assigned based on the common use in literature.	48
5.1	Hyperparameters optimized for each model.	65
5.2	Precision, recall and f1-Score results of N-BaIoT dataset.	66
5.3	Precision, recall and f1-Score results of Bot-IoT dataset.	67
5.4	Training interval of each evaluated model of N-BaIoT dataset.	71
5.5	Detection, or testing interval, of each evaluated model. The ratio column is calculated by taking the mean value between the experiments and dividing it by the number of rows on the test set. This column shows the average interval that the model would take to detect an attack per incoming data.	72
5.6	Training and testing intervals of Bot-IoT dataset. The ratio column is calculated by taking the mean value between the experiments and dividing it by the number of rows on the test set. This column show the average interval that the model would take to detect an attack per incoming data.	72
5.7	Comparison between the detection metrics achieved in this work and recent results from the literature for the N-BaIoT dataset.	74

5.8	Comparison between the detection metrics achieved in this work and recent results from the literature for the Bot-IoT dataset.	74
6.1	Types of botnet attacks on N-BaIoT dataset and their labels.	78
6.2	Types of botnet attacks on Bot-IoT dataset and their labels.	78
6.3	Hyperparameters optimized for each model.	81
6.4	Precision, Recall and F1-Score results of the evaluated models.	81
6.5	Total training time of each model. The Size field corresponds to the number of rows. The ratio between total training time and the number of rows is not considered in this evaluation because the training step is always done with the whole data.	86
6.6	Classification time of each model. The Size field corresponds to the number of rows, and the Ratio columns indicate the proportion of the time interval spent in relation to the total number of rows.	87
6.7	Comparison between the classification metrics achieved in this work and recent results from the literature for the N-BaIoT dataset.	89
6.8	Comparison between the classification metrics achieved in this work and recent results from the literature for the Bot-IoT dataset.	89

Contents

1	Introduction	13
1.1	Motivation	13
1.2	Problem Statement	15
1.3	Contributions	16
1.4	Organization	17
2	Related Work	19
2.1	IoT and Edge Computing Architectures in Literature	19
2.2	IoT Applications with Machine Learning in Literature	21
2.2.1	Supervised Strategies	22
2.2.2	Unsupervised Strategies	23
2.2.3	Semi-Supervised Strategies	24
2.2.4	Reinforcement Strategies	25
2.3	Security Solutions for IoT Environments	25
2.3.1	Machine Learning Strategies for Security in IoT	26
3	Internet of Things Environments	33
3.1	Architecture of IoT Environments	34
3.1.1	IoT Architecture with Edge Computing	35
3.2	IoT and Edge Applications Requirements	37
3.2.1	Communication	38
3.2.2	Energy Consumption	39
3.2.3	Computational Resources	40
3.2.4	Data Management	41
3.2.5	Security and Privacy	42
3.3	Botnet Attacks on IoT Environments	43
3.3.1	IoT Datasets	44
3.3.1.1	N-BaIoT	44
3.3.1.2	Bot-IoT	45
4	Machine Learning Architectures for Internet of Things	46
4.1	Machine Learning Classes	47
4.2	Applying Machine Learning Techniques in IoT Environments	50
4.2.1	IoT Solutions with Machine Learning	51

4.2.1.1	Applications	51
4.2.1.2	Requirements	52
4.2.2	Edge Computing Solutions with Machine Learning	53
4.2.2.1	Applications	54
4.2.2.2	Requirements	55
4.3	Hyperparameter Optimization for Machine Learning Strategies	56
4.3.1	Hyperparameter Types	57
4.3.2	Bayesian Optimization	57
5	Botnet Attack Detection	59
5.1	Defining the Machine Learning Architecture: Variational Autoencoder	59
5.2	Experiment Setup	62
5.2.1	Input Data	62
5.2.2	Methodology	63
5.3	Results and Discussion	64
5.3.1	F1-Score and Accuracy	65
5.3.2	Training and Testing Intervals	71
5.3.3	Comparison with Related Works	73
6	Botnet Attack Classification	76
6.1	Defining the Machine Learning Architecture: Convolutional Neural Network	76
6.2	Experiment Setup	78
6.2.1	Input Data	79
6.2.2	Methodology	79
6.3	Results and Discussion	80
6.3.1	F1-Score and Accuracy	82
6.3.2	Training and Testing Intervals	86
6.3.3	Comparison with Related Works	87
7	Conclusion	90
7.1	Future Work	91
	Bibliography	93
	Appendix A Botnet Attack Detection Results	102
	Appendix B Botnet Attack Classification Results	111

Chapter 1

Introduction

1.1 Motivation

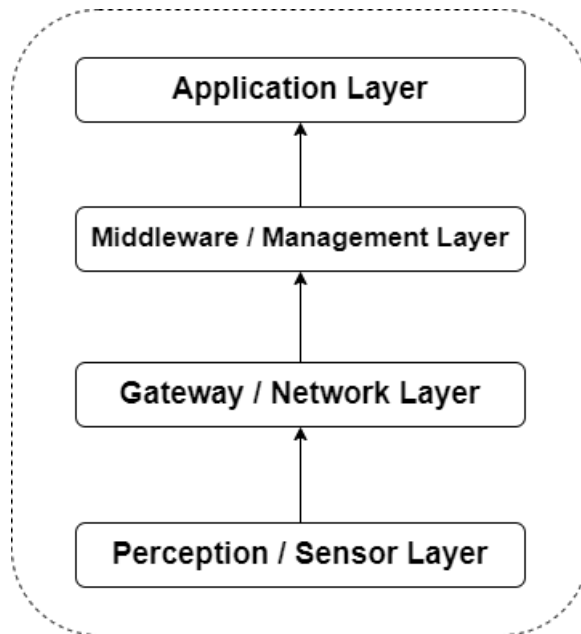
In the past few years, the Internet of Things (IoT) has become one of the most researched topics in the computer science field. [Nikoui et al. \[2021\]](#) noticed that Internet of Things studies increase about 40% annually as the usage and applicability of connected services rise. Smart homes, smart cities, smart healthcare, and transportation are fields in which IoT grants huge advantages and technologies. The expectation is that this trend will increase as IoT applications help society to analyze and predict data, monitor and automate processes and become more and more connected. For instance, IoT and artificial intelligence (AI) solutions are being applied to help doctors and scientists with the recent COVID-19 pandemic, as observed by [Ghimire et al. \[2020\]](#). There are solutions created to help diagnose patients with the disease, predict the disease's spread, and develop new drugs and vaccines. As expected, smart health is one of the areas that most developed in the past five years [[Nikoui et al., 2021](#)].

Despite the significant increase in IoT solutions, security is still a relevant and challenging requirement to guarantee when it comes to IoT systems. That happens due to the heterogeneity of most IoT environments, with vast amounts of transferred data, big attack surface and a significant number of connected devices with diverse protocols and standards [[Aversano et al., 2021](#); [Hussain et al., 2020](#); [Laguduva et al., 2019](#)]. One of the most concerning issues in the security field is the possibility of spreading malicious data across the networks and infecting entire environments in large DDoS attacks, leading to problems like system malfunctions and data stealing [[Aversano et al., 2021](#); [Hoque et al., 2015](#)]. In this context, Botnets are malwares that are widely used to affect IoT applications. They are machines used by hackers to coordinate malicious incursions, which can cause considerable losses to enterprises or even personal data [[Hoque et al., 2015](#); [Kolias et al., 2017](#)]. Some studies have proposed a solution to early detect attacks on incoming information by applying machine learning (ML) strategies to learn data patterns and identify which information is benign and block malware information [[Meidan et al.,](#)

2018; Koroniotis et al., 2019].

Creating IoT applications combined with AI techniques is a current trend. According to recent studies [Aversano et al., 2021; Mahdavinejad et al., 2018], it is possible to achieve remarkable results by combining these technologies. Since there are many fields in which IoT could be applied and many available IoT solutions (smart healthcare, transportation, grid, and mobile communication, to cite some examples), it is necessary to analyze the demanded requirements of the applications and devices to model the best approach to tackle the proposed issues. AI techniques can be used to achieve recent requirements related to IoT applications, such as guaranteeing the security of the systems, limiting energy consumption, choosing nodes and gateways where data should be processed, performing decision-making tasks, and scaling data, which is the topic of many studies [Li et al., 2018; Sun et al., 2019; Capra et al., 2019; Wang et al., 2019; Mahdavinejad et al., 2018].

Figure 1.1: Concise representation of consolidated architecture for IoT systems. A complete model is presented in Chapter 3.



Source: Ma et al. [2013].

Figure 1.1 represents the most consolidated architecture employed to develop IoT applications. It shows the perception layer, which includes the end devices and actuators; the gateway layer, which enables communication and transferring of data to the higher levels; the management layer, where cloud services usually process data; and the application layer, which provides an interface with the user. According to Yahuza et al. [2020] and Xiao et al. [2019], cloud-centered solutions that process data only in the management layer can no longer meet modern IoT requirements. Because of that, the edge computing paradigm emerges as a solution that can improve the latency response of the system, in-

crease data privacy, minimize power consumption and reduce communication constraints of an IoT application [Merenda et al., 2020; Laguduva et al., 2019; Samie et al., 2019; Corneo et al., 2021]. This strategy is a recent and promising paradigm of the Internet of Things, which sets an architecture that allows managing gathered data next to the end devices, providing interesting features for real-time and low-power applications [Merenda et al., 2020].

Although edge devices are especially susceptible to external attacks, they can be relevant to the security and privacy scopes by preventing data from being continuously exposed and sent to the cloud. They can also be used alongside ML techniques to perform early detection and classification of attacks that come through network channels and prevent infectious data from spreading across the network, potentially protecting the IoT environment. Those facts extensively motivate studies in the area.

1.2 Problem Statement

Choosing a solution to solve a problem is not always straightforward. In particular, the Internet of Things is a large field, and its potential is still being discovered and studied. Therefore, it would be necessary to carefully consider the constraints and requirements of building a system to decide the best choices. Edge computing is particularly interesting to a specific class of problems (e.g., real-time, low-power devices [Wang et al., 2018a; Capra et al., 2019]), and developing machine learning solutions also requires an understanding of the architecture and the goals of the system.

As the use of IoT applications rises, many challenges and possible studies appear in the context. IoT systems are known for generating vast amounts of data in short intervals, which should be appropriately processed, analyzed, and stored. Edge computing is a paradigm developed to support integrated IoT systems and improve their performance [Yu et al., 2017; Corneo et al., 2021]. Architectures based on local or cloud servers offer high computational and storage resources but present constraints in communication and energy-efficiency fields. Since many recent IoT applications depend on real-time operations, such as smart healthcare, smart transportation, and smart grid, transmission latency, bandwidth, and power consumption have become bottleneck problems [Yu et al., 2017].

A requirement that is becoming especially important when developing IoT and edge computing solutions is security against malicious attacks [Yahuza et al., 2020]. IoT environments are particularly susceptible to attacks that come from the network, once the majority of devices do not incorporate proper defense software and present a large

attack surface [Alqahtani et al., 2020; Laguduva et al., 2019]. Botnet attacks, which are a particular type of those malwares, can cause infection of the environment, prevent devices from working properly and lead to stealing of data [Hoque et al., 2015; Koliass et al., 2017]. Implementing isolated solutions for all devices is challenging since they are commonly heterogeneous and built with diverse protocols and standards. However, a recent solution being studied to avoid this type of problem is to perform an early attack detection step on incoming data using AI approaches [Aversano et al., 2021; Hussain et al., 2020]. The edge paradigm could also be beneficially applied in this context, preventing malicious data from reaching higher levels of the application and enabling real-time identification of data.

The edge computing architecture proposes executing data management actions on the edge of the systems, in sensors or edge nodes close to end users, as seen in Figure 1.1. This approach intends to avoid huge latency during communication, allow real-time processing, and improve power consumption and privacy, since data does not need to be continuously exchanged [Capra et al., 2019; Merenda et al., 2020]. However, there are some issues to consider: sensor/edge nodes usually have limited computing resources to execute complex algorithms and limited storage capacity. IoT systems are highly heterogeneous, which cause constraints to the management of data and communication between different devices. To mitigate those issues, machine learning techniques are being applied to IoT and edge services to help in processing and governance tasks, including critical improvements to the security field [Wang et al., 2020; Aversano et al., 2021; Hussain et al., 2020; Tuli et al., 2020].

With the theoretical reference gathered during this study, the problem to be analyzed and solved in this work are the botnet incursions that have recently threatened many IoT applications. The research on artificial intelligence shows that ML strategies are suited to tackle IoT systems' typical constraints and provide reliable solutions, especially regarding the security of the environments. Combining those strategies with edge computing is possible and beneficial to enable real-time solutions for the early attack detection task, which is also investigated. With that knowledge, it is possible to design proper architectures to tackle the issues and apply strategies that meet the requirements.

1.3 Contributions

Considering the growing importance of IoT technologies combined with machine learning techniques to the continuous development of helpful and performative IoT solutions, as presented in the sections above, the main contributions of this work are:

- Perform a complete research and provide an analysis about the use of machine learning solutions with IoT and edge computing applications to help define the best techniques to be employed based on the requirements and constraints of each application;
- Propose two deep learning architectures to perform detection and classification of botnets attacks, validate their performance by executing experiments with known datasets and evaluate the hypothesis of employing the final models in the edge computing context;
- Review IoT architectures and machine learning usage in IoT and edge computing services in the literature, considering their requirements, constraints, and challenges.

In this study, we review machine learning architectures that can be used in the Internet of Things, and the possible technical features and challenges. Finally, the work presents two empirical experiments that apply deep learning techniques (DL), the Variational Autoencoder (VAE) and the Convolutional Neural Network (CNN), for early attack detection and classification tasks on incoming data, which also tackles the problem of security, a worrying constraint for IoT environments. This work is essential to provide a deep analysis that could help in system design, propose possible solutions to IoT's recent and relevant problems, and instigate more investigations within the area.

1.4 Organization

Chapter 2 reviews studies proposed to support this project, explaining and summarizing the most important solutions.

Chapter 3 introduces the main characteristics of IoT environments, showing the most consolidated structures to design IoT and edge computing systems and the most common constraints and requirements that should be considered when building those applications. This chapter also introduces the object of study of the experiments conducted in the following chapters, the Botnets.

Chapter 4 discusses the possible machine learning classes and a summary of ML architectures that can be employed in IoT and edge computing solutions considering the features and constraints of the designed applications.

Chapter 5 proposes a case study considering the reviewed security concepts for IoT applications. The strategy proposed in this chapter is to detect incoming botnet attacks early in a simulated IoT environment with a DL technique. The study employs two known

IoT datasets, N-BaIoT [Meidan et al., 2018] and Bot-IoT [Koroniotis et al., 2019], which comprise real traffic data of various sensors in operation inside an IoT environment.

Both datasets are also used in Chapter 6 to perform early attack classification of the botnet attacks. This strategy is proposed as an additional step after detecting malicious data. This approach intends to gather more knowledge about the possible types of botnets and prepare efficient and reliable protection routines.

Chapter 7 concludes this work and discusses possible lines of future investigations.

Chapter 2

Related Work

The studies about the Internet of Things are prolific since it is a new and promising area, as well as machine learning models and applications researches. Douglas Adams, the writer of “The Hitchhiker’s Guide to the Galaxy” once said:

“We invented the computers; first, we built them in the size of whole rooms, then they started to fit on our tables, then in our briefcases and then in our pockets. In a little while, they will be as ubiquitous as dust - you will be able to dapple computers everywhere. Little by little, all of our environment will become much more interactive and smart, and we will live in a way that would be very difficult for those who are living in this moment to understand.”
[Adams, 2002].

He gave this speech in April 2000 but did not know how correct he was. Society is entering an era of high technology, small but powerful components, and fast processing. Those features were essential to the rise of the IoT concept and development.

This chapter aims to organize relevant and recent studies of machine learning for IoT and edge devices that will be useful for the proposed work. The first section presents known IoT architectures in the literature (Section 2.1), followed by a summary of recent machine learning solutions applied in IoT and edge computing (Section 2.2). The last section is about security in IoT and edge environments, considering models commonly used to tackle recent issues affecting those devices (Section 2.3).

2.1 IoT and Edge Computing Architectures in Literature

First, it is essential to consider IoT’s architecture models since it helps to comprehend better how the data flows across those systems and how edge devices could utilize it.

The most consolidated architectures consider at least four layers [Ma et al., 2013]: sensing layer, the partition in which data is gathered and represents the bottom border of the system; network layer, where the communication protocols are defined and utilized; management or middleware layer, responsible for data processing, storage, and security; and application layer, which provides the interface to the user. Commonly, the management layer is divided into sub-layers, each acting under a specific responsibility. This model is also well explored in the studies of AlSuwaidan [2019], Sikwela [2012], and Ray [2018].

AlSuwaidan [2019] discusses the concept of Internet of Everything (IoE), a natural expansion of the IoT term considering an environment with connected devices, people, and processes. The project shows a similar architecture as presented in Chapter 3, focusing on the data management layer. The author suggests the split of this tier into three different layers: data source layer, master data management layer, and top-level data management service layer. The first layer treats data gathered from different sources, like services, processes, people, documents, and machines. The next tier is responsible for executing the cleaning, governing, and processing of the data, considering big data and machine learning techniques to deal with the endless amount of information. Finally, in the last layer, the idea is to execute additional steps over the master data, such as security, integrity, and federation. The main issue that the author face is the step of storing and migrating data.

Ray [2018] examines the technical fields of IoT architecture, providing examples, standard protocols, and possible research issues, like handling many devices and information. The author proposes a consolidated Service Description Language (SDL) to process data from different sources, types, and sizes and proposes standardization for IoT systems.

Using centralized cloud storage along with Internet of Things data, management is a consolidated methodology that prevents purchasing (usually expensive) servers and machines, and is called cloud computing. Also, there is fog computing, a method that brings data processing to the model's borders at the network level. Since cloud computing faces the issues of high latency and bandwidth, security, and reliability, executing some data management steps in the lower layers could overcome these problems [Sikwela, 2012]. For such, fog computing allocates those activities in processors located within LAN (Local Area Networks) hardware around the network layer. Another similar approach is the so-called edge computing. Like fog computing, the goal is to execute data processing on the system's edge, but closer to the sensors layer, the lowest in the model. That means the information obtained could be instantly analyzed, eliminating network dependency issues.

Yu et al. [2017] summarize the main characteristics of edge computing, and its advantages and disadvantages. Edge computing arises due to the difficulties cloud computing faces in providing the correct Quality of Service (QoS) and Quality of Experience (QoE) to recent devices. The proposed model is the key to offering service with reduced network bandwidth and latency, essential to systems that demand near real-time responses, like

smart healthcare, transportation, and grid [Tuli et al., 2020; Arachchige et al., 2020]. On the other hand, it is vital to think about the weaknesses of this kind of implementation. Some challenges to consider are integrating various systems, security standardization, and managing computational resources. Table 2.1 summarizes the researched studies about IoT architectures and data management and shows the advantages and solutions proposed in this work with the analyzed papers.

2.2 IoT Applications with Machine Learning in Literature

In order to manage resources, a recent methodology applied in many projects is the use of machine learning techniques within IoT and edge devices. Some studies, that are presented in this section, show that this kind of technique allows secure, energy-efficient, reliable and autonomous systems. The discussed studies also represent important contributions to deal with constraints and requirements of IoT and edge based services. The following section is organized considering the four main subcategories of ML strategies (supervised, unsupervised, semi-supervised and reinforcement algorithms). Table 2.2 shows a list of relevant works in the studied field as well as their main topics.

Proposals such as Cui et al. [2018], Mahdavinejad et al. [2018], Samie et al. [2019] and Wang et al. [2020] provide valuable knowledge about the recent use of machine and deep learning strategies in IoT environments and how those technologies can benefit from each other. It is a trend to also evaluate and analyse efforts with edge computing solutions. Cui et al. [2018], for instance, elucidate the potential of ML solutions for tasks like traffic profiling, device identification and security. They point that the model reliability for different environments, huge traffic volumes, DDoS attacks and security against malicious attacks, heterogeneity of data and processing capacity for ML algorithms are highlights among the recent challenges, specially for edge computing.

Mahdavinejad et al. [2018] also bring considerations about applying ML techniques to treat IoT data, considering the taxonomy of ML algorithms and the characteristics of data in real world scenarios. They analyse works among IoT solutions for smart city with combined cloud, fog and edge environments, highlighting challenges such as preserving data quality and privacy and dealing with huge and heterogeneous information. Samie et al. [2019] and Wang et al. [2020] analyse ML architectures for IoT with a special emphasis to the edge computing context, concluding that cloud-only systems are not fulfilling the needs of recent applications anymore, and researches on the area should be

increasing in the next years, with a boost on deep learning strategies.

2.2.1 Supervised Strategies

Supervised techniques on machine and deep learning enable the algorithm to learn a function that defines how to map the input features to the output labels from a previously labeled training set, which serves as an example for the chosen architecture. Some examples of techniques that can be supervised are: Naive Bayes (NB), Support Vector Machines (SVM), K-Nearest Neighbors (KNN), regression strategies, and Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) on the deep learning scope [Zaki and Meira, 2020; Gron, 2019]. Those techniques are being used by recent studies to tackle governance, efficiency, security and communication issues in the IoT context, which are briefly explored during the following topic.

Gao et al. [2021] introduce the concepts of Federated Learning (FL) and Split Learning (SL), which are distributed machine learning methods that enables training steps without needing to access raw data from end devices, that is, protect clients' private information. The first presented method initializes a global ML model in a remote server and offloads the model to the edge nodes, enabling clients to perform the training phase locally; the latter splits the training of the neural network (in particular, a 2D CNN) into two steps, one performed in the client side, and the other in the server side. Authors also propose and utilize a Raspberry Pi 3 to evaluate a combined version of both techniques, the Splitfed Learning (SFL), by comparing the performances achieved with FL and SL paradigms.

Sun et al. [2019] develop a lightweight CNN for IIoT (Industrial Internet of Things) applications and apply it with the so called transfer learning. The transfer learning idea is to pre-train the neural network with some known biases, and utilizes it as the starting point to train new procedures. This approach allows the development of low cost algorithms, which are suitable for architectures with limited resources. The researchers found out that the implementation of artificial intelligence in edge servers is feasible and provides results with great accuracy.

In the edge computing field, Merenda et al. [2020] study the use of machine learning algorithms within edge devices, bringing some known requirements to this kind of application, like network communication and privacy. They also present techniques to develop AI in edge devices with lower processing costs, introducing concepts like joint computation. This technique implements offloading in the edge, guaranteeing that some layers of the supervised DNN (Deep Neural Network) could be processed on cloud, and others on

the edge node, reducing the resources needed to perform such algorithms. Finally, they conduct an image classification experiment (using the known MNIST set) and its evaluation by applying machine learning in an edge device, built from a NUCLEO-F746ZG board with 135.78 kB of RAM and 668.97 kB of flash memory. For image processing problems, authors summarize that the most employed solutions comprise DNN algorithms, in particular, Convolutional Neural Networks (CNNs).

Another example of supervised learning in literature is given by [Serra et al. \[2021\]](#), which create an experiment to perform evaluations with drones in SAR (Search and Rescue) missions. They develop a fog architecture to offload costly processes of face detection and recognition to the most appropriated cloudlet server available with machine learning algorithms, and test the technique with several supervised classifiers: SVM (Support Vector Machine), LR (Logistic Regression), MLP (Multilayer Perceptron), NB (Naive Bayes) and Dummy (random guesses). The choice of common classifiers over deep learning is because they require less processing resources than the latter. The article shows that MLP and LR are the techniques that deliver the best accuracies, 99.64% and 99.20% respectively. Those results are interesting to demonstrate that different machine learning algorithms, whether they are deep learning or not, could be interesting depending on the purpose of the project.

2.2.2 Unsupervised Strategies

Differently from the supervised solutions, unsupervised strategies do not rely on labeled training data to fit the algorithms. In fact, the model should learn the function to find patterns among the data by itself, without any kind of human intervention [[Zaki and Meira, 2020](#); [Gron, 2019](#)]. Some examples of algorithms that can be developed in an unsupervised manner are: KNN and SVM, Gaussian Mixture Models (GMM), K-Means that perform clustering of the data, and Autoencoder and Long-Short Term Memory Autoencoder (LSTM) in the deep learning field.

To illustrate the unsupervised techniques, [Janjua et al. \[2019\]](#) choose to apply a two-stage unsupervised machine learning algorithm in their project, which is developed to identify rare-case events occurred in a data stream. The benefit of this approach is that their algorithm can learn from the environment events without needing data labeling, which would be a costly task for the situation. They apply clustering techniques and are able to develop the project fully on the edge.

Similar to [Serra et al. \[2021\]](#)'s work, [Kolomvatsos \[2021\]](#) propose an architecture to deploy real-time edge applications and choose between several possible nodes. On the

other hand, this work utilizes an unsupervised technique with pruning steps to achieve its goal, in particular a clustering algorithm. This choice is made in order to avoid the cost of training process, which occurs with supervised methodologies. Also, the author treats the node selection as a grouping problem, such that clustering techniques should be effective to this design.

[Sharma et al. \[2021\]](#) bring an unsupervised Autoencoder technique to deal with an industrial problem. They use a Federated Learning approach (as explained in [Gao et al. \[2021\]](#)) in order to maintain client's data privacy and deploy Variational Autoencoders (VAEs) combined with K-Means algorithm in the edge to perform clustering methods on data provided by heat sensors. The achieved goal is to evaluate and predict heating points on an industrial furnace.

2.2.3 Semi-Supervised Strategies

The semi-supervised learning is employed when the model learns from a small amount of labeled data, but it is also able to predict outputs for the unlabeled data. Some examples of algorithms that can be implemented in a semi-supervised way are the SVMs, Fuzzy C-Means, Graph-based ML algorithms and Variational Autoencoders and Autoencoders. Regarding those semi-supervised techniques applied in IoT systems, [Ji et al. \[2021\]](#) propose semi-supervised Support Vector Machine (SSVM) algorithms in order to detect crossing intentions from pedestrians to be utilized by autonomous vehicles. Authors study different models of SSVM by applying collected data of pedestrian crossing intentions and vehicle movement databases, achieving a final prediction accuracy of 94.83% to the proposed problem.

[Rathore and Park \[2018\]](#) develop a framework for attack detection in a fog based architecture utilizing the proposed paradigm of semi-supervised Fuzzy C-Means based on Extreme Learning Machine (ELM), called ESFCM. The work brings advances to the security area for edge-based systems and culminates with an accuracy of 86.53%. Semi-supervised learning was chosen in this case to permit dealing with partially labeled data, and the ELM enables a fast performance. Paper also compares the proposed model in relation to traditional machine learning classifiers.

2.2.4 Reinforcement Strategies

The reinforcement learning aims to develop intelligent algorithms that are capable of performing self-supervision by assigning positive values to the desired actions and penalizing the unwanted behaviors until it converges to an optimal solution [Zaki and Meira, 2020; Gron, 2019]. Reinforcement learning techniques are the newer ones among the explored types, but some advances are being achieved in the field, with Q-Learning, Markov decision problems and Deep Q-Networks as examples of algorithms.

An example of reinforcement learning applied in the IoT context is shown by Chen et al. [2019]. The researchers propose a platform capable of learning the network environment of mobile devices and allocating resources with maximum performance. They combine reinforcement learning techniques with DNNs to create a self-supervised ambient. The resource allocation is made through decision tasks, making the reinforcement algorithm the best solution for the desired goal.

Recent studies about self-supervised systems seems to be prolific, such as in Castano et al. [2020], which develop a system to monitor characteristics of sensors. One of the great drawbacks faced within sensors networks is the reliability of the devices. Since IoT solutions depend on the information obtained across the environment to make decisions and actions, the reliability of detectors represents great importance to the flow. The authors, thus, create a self-learning platform to identify reliability of sensors with reinforcement learning (Q-Learning) in a global level and classification (MLP - Multi-layer Perceptron) and clustering methods (KNN - K-Nearest Neighbors) on local level, applied in autonomous driven transportation. In conclusion, it is highlighted the importance of continuous study in this field, since there are challenges regarding performance, pre-filtering and computational cost on edge devices. The experiment achieves quality results, with low absolute error (below 40%) and good generalization (above 70%), which encourages further working.

2.3 Security Solutions for IoT Environments

The last chapters of this work (Chapters 5 and 6) treat about data anomaly recognition in IoT environments. With the continuous growth of IoT solutions in several sectors of society, the concern about security and reliability of those devices also grows. IoT environments are usually composed by many heterogeneous sensors and actuators,

creating huge nets of connected devices through Wi-Fi, Bluetooth, ZigBee and other communication protocols. They are usually involved in continuous exchange of massive data, which contributes to their susceptibility to being attacked by network malwares, as well as the lack of standardization in devices' security protocols and the big attack surface of the environments due to the pervasive deployment of large numbers of elements [Alqahtani et al., 2020; Aversano et al., 2021; Hussain et al., 2020].

In particular, the botnet is a common attacking technique that is currently being used to affect IoT systems. The botnets are networks of infected devices that perform different attacks in unprotected environments. Once infected, the device can be used for malicious activities, such as stealing of data, accessing protected networks, and performing large DDoS (Distributed Denial of Service) attacks to infect more devices [Meidan et al., 2018]. Koliass et al. [2017] present the botnet known as Mirai and its variants, which is usually employed to perform DDoS attacks on targeted servers by propagating through unprotected IoT devices. The Bashlite, also known as Gafgyt, is another botnet that threatens those environments by using brute force techniques to gain access on Linux-based devices [Pa et al., 2016]. Both of the botnets are explored by Meidan et al. [2018] work, which provides a complete dataset with botnet behavior against IoT systems.

In the works of Aversano et al. [2021], Hussain et al. [2020], Zeyu et al. [2020], Yahuza et al. [2020] and Xiao et al. [2019], authors summarize and analyze the impact of recent studies in the IoT security field, highlighting relevant efforts, the most common threats, current trends and future challenges. The use of machine learning techniques to early detect and prevent network attacks arises as an optimistic solution to be applied within IoT and edge environments. The papers of Zeyu et al. [2020], Yahuza et al. [2020] and Xiao et al. [2019] bring a special emphasis to edge computing solutions, pointing security and privacy constraints as relevant enablers for this technology. A summary of recent studies that propose techniques to deal with security issues in IoT environments is provided in Table 2.3.

2.3.1 Machine Learning Strategies for Security in IoT

Considering the attack detection and classification problems for IoT systems, there are also several efforts that successfully apply ML solutions. Meidan et al. [2018] propose the N-BaIoT set and Koroniotis et al. [2019] propose the Bot-IoT set, which are datasets that contain traffic information from IoT networks when receiving benign and malicious data. Meidan et al. [2018] propose a semi-supervised network of Deep Autoencoders to early detect botnet incursions on the N-BaIoT and achieve 100% of True Positive Rate.

[Koroniotis et al. \[2019\]](#), on the other hand, evaluate detection and classification tasks with some ML architectures: a RNN, a LSTM and a SVM. Those datasets are extensively used on literature to study botnet behavior and improve current security solutions

[Alqahtani et al. \[2020\]](#) use a supervised optimized XGBoost classifier combined with a genetic algorithm on the N-BaIoT dataset for the binary classification problem. They also add an extra step of feature selection based on the Fisher score to identify the features that grant most important information for the classifier. They are able to achieve above 99.00% of accuracy and f1-score with this approach, even with the reduction of the feature space. [Nömm and Bahşi \[2018\]](#), on the other hand, choose an unsupervised solution based on SVMs and feature space reduction for the same set. Although unsupervised techniques usually result in reduced evaluation metrics, their solution is able to achieve a mean accuracy value of 96.64% and 92.55% for unbalanced and balanced distributions, respectively. [Borges et al. \[2022\]](#) analyze the number of packets that a device from the N-BaIoT set transmits and perform a multiscale ordinal patterns transformation on the time series data to apply an Isolation Forest algorithm to detect anomalous data. With that approach, they are able to achieve 99.50% of accuracy with a unique model for all of the devices, and 100% when using separated models for each device, while applying a simple ML solution.

[Popoola et al. \[2022\]](#) employ both N-BaIoT and Bot-IoT datasets to study the hyperparameter optimization of DL models and its effects on the final results for detection and multi-classification of attacks with a supervised DNN. Other works that consider the attack classification problem that can be highlighted are [Htwe et al. \[2020\]](#) with the ML CART algorithm applied on the N-BaIoT set; [Tran and Dang \[2022\]](#) with the same dataset and an ANN architecture; and [Parra et al. \[2020\]](#) that propose a LSTM architecture for the N-BaIoT dataset attack detection.

Anomaly detection for the edge computing context is pointed as one of the most important trends by authors like [Xiao et al. \[2019\]](#) and [Zeyu et al. \[2020\]](#), which summarize the state-of-art for security of edge computing environments. This field of study still require further and deeper work, but it is possible to cite [Gopalakrishnan et al. \[2020\]](#) and [Tian et al. \[2020\]](#)'s works, which propose ML strategies for anomaly detection with edge devices. A solution that has also been explored by recent works is to execute offloading techniques on edge nodes to combine the resourceful computing capacities of the cloud and the real-time responses provided by the edge [[Roy et al., 2022](#)]. For instance, some costly algorithms can be ran on the cloud, like the training step of ML and DL architectures, while the prediction phase can be applied on the edge.

The next Chapter starts the discussions on IoT and edge computing environments, bringing information about the most common architectures developed for these scenarios, the requirements needed to enable reliable, fast and performative systems and the most relevant considerations regarding the security field for IoT.

Table 2.1: List of works that provide researches about IoT and edge computing architectures and data management efforts for IoT solutions.

Paper	Main Points of the Work
Ma et al. [2013]	Provide the most consolidated architecture for IoT solutions, which considers sensors layer, network layer, management layer and application layer. Comparing with the current work, it proposes to expand this structure and also consider edge computing solutions.
AlSuwaidan [2019]	Focuses on the data management layer providing an unraveled architecture to deal with governance, privacy, storing and migration of data. ML solutions explored on the present work could be used to deal with the challenges of storing and migration of data pointed out by the author.
Ray [2018]	Provides a survey analysis on the technical fields of IoT architectures with protocols and recent challenges faced on the IoT context, specially considering data treatment. As the current work shows, ML solutions can be successfully used to deal with huge amounts of data and organize data governance.
Sikwela [2012]	Explores the possible architectures for IoT environments and also brings the concept of fog and edge computing.
Yu et al. [2017]	Provides a study specifically about the edge computing architecture context, considering its advantages and constraints to the recent technologies. The proposed work explores a particular issue raised by the authors, the standardization of security routines.

Table 2.2: Summarizing main points treated on studied researches about machine learning models applied in IoT and edge solutions.

Paper	Main Points of the Work
Cui et al. [2018]	Survey about the application of machine learning techniques in IoT applications with recent examples on traffic profiling, device identification, security and edge computing scope.
Mahdavinejad et al. [2018]	A study about machine learning taxonomy and its possible applications in IoT with an use case emphasizing a smart city traffic application.
Samie et al. [2019]	Authors identify that cloud-centric solutions are not able to meet IoT requirements anymore. They propose and research inserting ML solutions into the IoT world with edge computing paradigms to improve recent technologies.
Wang et al. [2020]	A research about implementation of DL techniques to edge computing applications considering practical implementations, enabling technologies, challenges and future trends for pervasive and intelligent applications.
Merenda et al. [2020]	A study about machine learning models for edge computing devices and characteristics of this type of implementation with experimentation.
Gao et al. [2021]	Implementation of Federated and Split Learning techniques for offloading processes with CNN. It also proposes a blended offloading framework, the Splitfed Learning.
Sun et al. [2019]	Authors implement a lightweight Convolutional Neural Network with transfer learning technique applied on industrial IoT.
Corte et al. [2020]	Evaluation of different supervised classifiers to the task of tree detection with drones.
Janjua et al. [2019]	Application of an unsupervised technique in edge device to identify rare-case events on data stream with clustering solution.
[Kolomvatsos, 2021]	Proposing clustering technique to perform tasks management within edge devices.
[Chen et al., 2019]	Task management work developed with deep neural networks as well as reinforcement learning techniques.
[Serra et al., 2021]	Fog-based architecture with offloading processes to be employed in SAR missions. It evaluates the best classification technique among supervised options.

Sharma et al. [2021]	Proposing of unsupervised technique with Federated Learning approach to perform predictions on industrial IoT. The method was employed to guarantee privacy of client's data.
Castano et al. [2020]	Self-aware and self-learning model to detect reliability of employed sensors on IoT network with reinforcement learning blended with MLP and KNN.
Ji et al. [2021]	Proposing semi-supervised method (SSVM) to detect crossing intentions from pedestrians. The framework should be used in the context of autonomous vehicles.
Rathore and Park [2018]	Applying semi-supervised technique (Fuzzy C-Means) to improve security of system by detecting attacks on network. Also compares outcomes with state-of-art machine learning models.

Table 2.3: Summarizing works that treats about security for IoT environments, specifically for botnet attack detection. The table shows the main points addressed in the papers.

Paper	Main Points of the Work
Aversano et al. [2021]	<p>Authors review 69 articles about IoT security with DL solutions considering the chosen architectures, the security aspects and studied datasets. In the end, they are able to highlight gaps and future challenges of the field.</p>
Hussain et al. [2020]	<p>A comprehensive survey about application of ML strategies to IoT network nodes in order to improve security and privacy of the applications, considering that regular cryptography techniques are insufficient to secure recent IoT solutions.</p>
Zeyu et al. [2020]	<p>Focus on reviewing security solutions for edge computing nodes considering the main threats that are employed to attack those types of systems.</p>
Yahuza et al. [2020]	<p>Provide a survey focusing on studies about security and privacy for edge computing applications, considering the taxonomy of possible attacks, popular defense mechanisms and challenges to be dealt with in the future years.</p>
Xiao et al. [2019]	<p>Authors point that security and privacy constraints narrow the acceptance of edge computing applications as reliable solutions and provide a review of requirements and relevant researches, future challenges and important issues to address on further edge computing studies.</p>
Meidan et al. [2018]	<p>Authors create the N-BaIoT dataset containing botnet and benign information for IoT devices. It contributes with real world data that can be used to simulate and develop efficient models that could be used to improve IoT security. Authors also propose and evaluate a classification model.</p>
Koroniotis et al. [2019]	<p>Like Meidan et al. [2018], authors propose a novel dataset to study the effects of machine learning to the security task and evaluate it with three ML architectures.</p>
Alqahtani et al. [2020]	<p>Propose a XGBoost classifier combined with a genetic algorithm to detect attacks on N-BaIoT dataset with feature reduction.</p>
Nömm and Bahşi [2018]	<p>Create a SVM-based model to solve the attack detection issue considering feature reduction, balancing of the data and time evaluation.</p>

Popoola et al. [2022]	Studying the effects of the hyperparameter optimization for N-BaIoT and Bot-IoT datasets for attack detection and multi-classification, evaluating not only the metrics but also the time spent on training and testing phases.
Htwe et al. [2020]	To the multi-classification task, authors propose a pure ML architecture called CART comparing its results with the Naive Bayes algorithm.
Tran and Dang [2022]	Authors create an ANN strategy to perform multi-classification of botnet attacks and compare its results with other ML techniques.
Parra et al. [2020]	It is developed a Deep CNN to identify phishing detection on network data and a LSTM to perform botnet multi-classification per attack.
Borges et al. [2022]	Propose a novel approach that converts time series data with a multiscale ordinal transformation and apply an Isolation Forest algorithm to detect malign data based on its temporal behavior.
Tian et al. [2020]	Provide a distributed deep learning strategy to detect anomalous URL requests on edge computing applications.
Gopalakrishnan et al. [2020]	Develop a Deep Belief Network (DBN) with LSTM offloading to detect anomaly tendencies in the edge computing context, achieving metric values around 94.00%.
Roy et al. [2022]	Create an intrusion detection mechanism based on machine learning architectures with two layers: a fog and a cloud layer. Authors can choose whether the algorithms run on the fog or the cloud, enabling real time predictions and preventing from losing performance. They achieve great results with this approach and prove that the energy consumption is reduced with the fog-cloud architecture.

Chapter 3

Internet of Things Environments

The Internet of Things is a relatively recent field of study. The term was firstly utilized in 1999 by Kevin Ashton while developing an ubiquitous integrated system with RFID (Radio Frequency Identification) sensors. Since then, many technologies emerged to compose the paradigm of Internet of Things and many studies were conducted to propose standardization of the models, data management, security and privacy solutions [Yu et al., 2017] [Ray, 2018]. In order to develop propositions to those challenges, it is recommended to understand the flow of data in IoT systems and the most consolidated architectures to those models. For example, during data management studies it is relevant to acknowledge in which layer the data should be gathered, mounted or modified.

The main concept of Internet of Things is to promote an interconnected environment of physical devices, focusing on gathering and transmitting data and information [Ma et al., 2013]. The core unit of an IoT network is the sensor, a device capable of responding to external stimuli and generating signals that can be converted and interpreted. Sensors, embedded systems and communication devices are widely used in industry and electronic fields to build automatic responses based on known behaviours and measurements [Sun et al., 2019]. Recently, those units are being employed more and more in daily contexts: RF sensors and NFC (Near Field Communication) protocol are widely used in public transport access and cash transactions [Lazaro et al., 2018]; temperature, humidity, pressure and other sensors are applied in cities, airports and universities to obtain or study environmental conditions; speak sensors are employed in smartphones and smart hubs to understand voice requests from users, and so on. All of these data can be processed and transmitted across internet or communication protocols in order to gather information about the analysed context and generate intelligence or demanded actions.

In this chapter, it is initially explained the classic IoT architecture with which IoT environments are usually constructed in Section 3.1, also bringing an architecture option to be used with edge computing devices, considering the particular characteristics of edge technologies. Section 3.2 treats about the most common requirements related to IoT and edge devices, or, in other words, the constraints and characteristics that should be considered when developing solutions for those types of environments. The last section presents the concept of botnets, that are widely used to threaten IoT environments, and

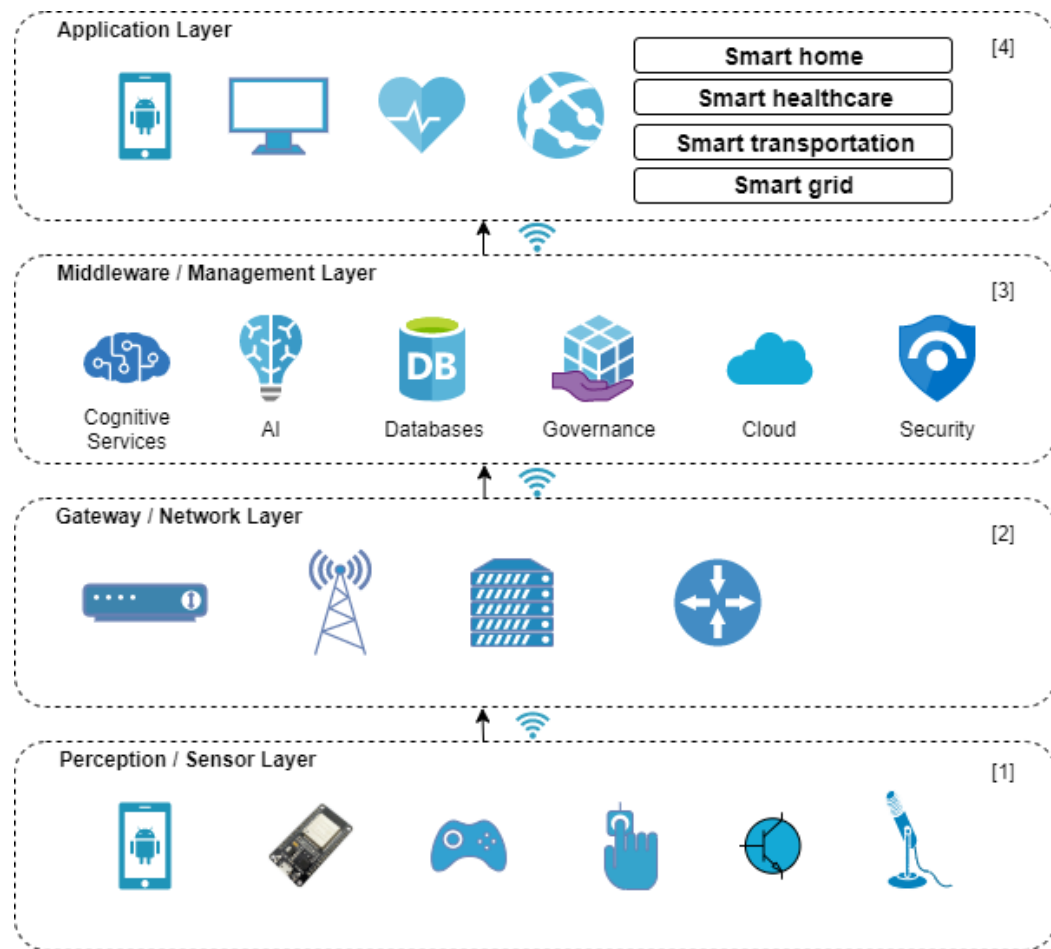
datasets that allow prolific studying in the field (Section 3.3).

3.1 Architecture of IoT Environments

With the statements proposed by authors in [Ma et al. \[2013\]](#), [AlSuwaidan \[2019\]](#), [Sikwela \[2012\]](#) and [Ray \[2018\]](#), it is possible to verify that the most consolidated IoT structure considers a base organization as represented in Figure 3.1. The characteristics of each layer, their features and the possible interactions between them are described in the following list:

- [1] The first layer, looking from bottom to top, is the perception or sensors layer, where all data is first gathered through many different devices. Data are one of the most important inputs on the globalized digital world, since they enable decision-making actions, analysis based on real information and development of intelligence [[Sikwela, 2012](#)].
- [2] Once data are gathered, it is required to send them to a suitable server where the information could be correctly processed and analysed. The transmission of data is executed across the gateway or network layer, where the necessary protocols are implemented in order to transfer them to higher layers or another devices (machine-to-machine connection). Some known types of networks employed in IoT context are LANs (Local Area Network) or WANs (World Are Network) and its associated protocols, like Wi-Fi, Ethernet or 3G/4G/5G, Bluetooth, LoraWAN, 6LowPAN, Z-Wave and ZigBee [[Merenda et al., 2020](#)].
- [3] Data should be then transferred to a remote resourceful server, usually the cloud. This area is called middleware or management layer. Here, data can be managed with different techniques, like machine learning or statistical algorithms, in order to provide intelligence to its users; they should be correctly stored to keep a history of the information and it is also necessary to develop security and privacy actions to protect their structure and the information contained in data. There are also efforts to study the construction and correct management of data, since they are gathered by different sensors, with varied protocols and definitions [[Sikwela, 2012](#)] [[Ray, 2018](#)].
- [4] Finally, with data properly analysed, it is possible to release them to users on the application layer, where they can be utilized to take decisions and execute actions.

Figure 3.1: Consolidated architecture of Internet of Things systems.



Source: [Ma et al. \[2013\]](#).

As an example of the flow, consider a temperature sensor, whose metrics can be used to define if an air conditioner should be turned on or not. The signals would be gathered by the sensor, sent through internet to a cloud server for processing and storing and the result would trigger a connected air conditioner if the determined requirements were reached. This flow works, but it tends to spend some time during the communication with cloud server, raising the latency of the system and its response time.

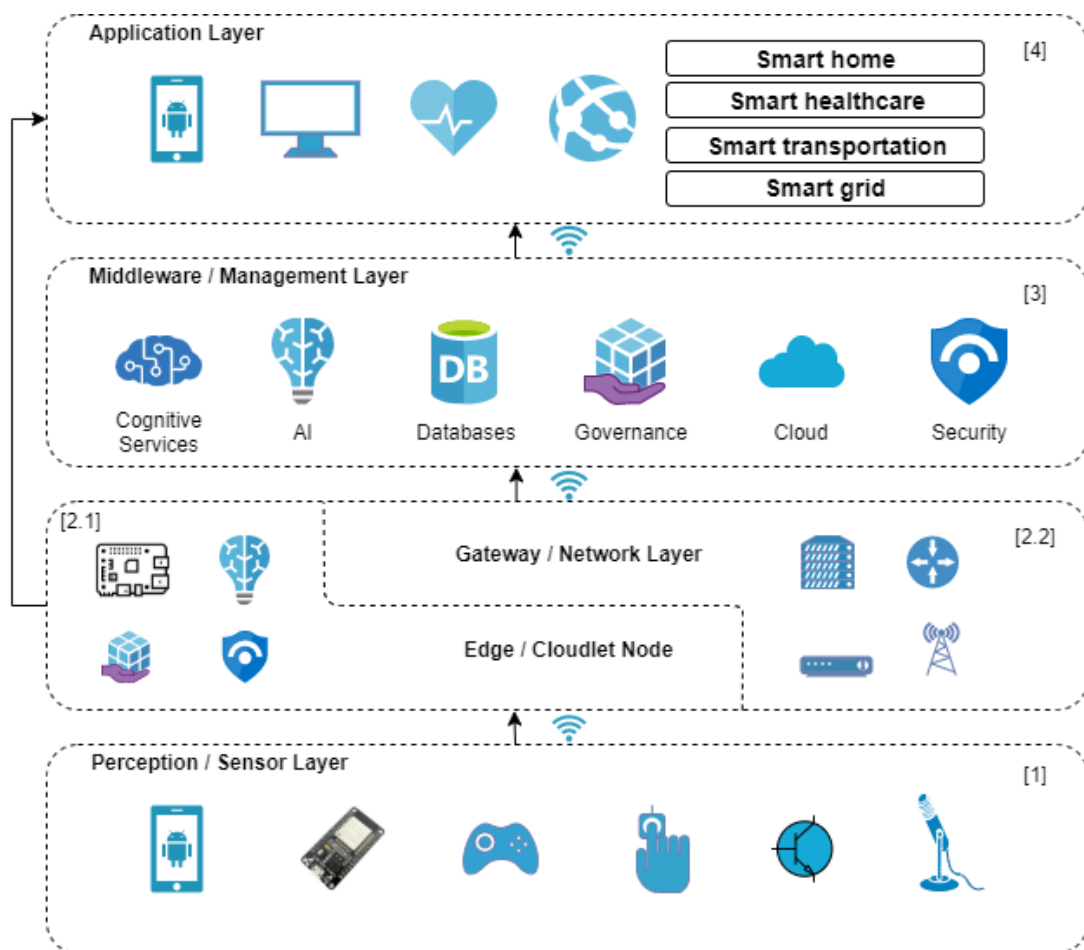
3.1.1 IoT Architecture with Edge Computing

The edge computing paradigm could offer a solution for the latency/communication drawback presented on previous section, by processing the obtained signals in a middle-node closer to the sensor. This node is called cloudlet or edge node, as represented by layer number [2.1] in Figure 3.2. The edge node is composed by any device that

could provide modeling, processing and management of data [Ceselli et al., 2017] [Wang et al., 2019], such as Raspberries, Arduinos, ESPs (from Espressif) or GPUs (Graphic Processing Units). Since the cloudlet node is closer to the sensors node, or perception layer, the time spent to exchange information between the layers is much lower, granting real-time responses.

In the given example on Section 3.1, the temperature sensor could send data to be processed by an Arduino on edge node, and the microcontroller would instantly return whether the air conditioner should be triggered or not. Also, it could send the weather information to the cloud server only for storing, since edge-cloud hybrid architectures are useful to maintain history, governance of data and provide more processing power to the applications [Merenda et al., 2020] [Yu et al., 2017].

Figure 3.2: Another modeled architecture, now considering the paradigm of edge computing. Edge, or cloudlet nodes [2.1], are able to communicate with management layer [2.2], or can directly send response or information to application layer [4], depending on the design and the purpose of the structure.



Source: Ceselli et al. [2017].

By inserting the edge layer to provide data management closer to the end-user, some benefits and drawbacks can be enumerated. Since data do not need to be sent to

cloud servers to be processed anymore, communication and response latencies are tremendously reduced, enabling real-time data analysis and improving the QoS of the application [Yahuza et al., 2020]. Techniques can be also used alongside edge implementation to balance the energy drain of cloudlet nodes and enable low-power consumption applications [Ray, 2018]. The architecture also boosts the growth of mobile IoT-edge systems (called Mobile Edge Computing, MEC), specially with the rise of 5G networks, by improving communication between services and end-users and providing high scalability [Xiao et al., 2019].

As possible drawbacks that may arise with this architecture, edge computing applications usually require algorithms to run quickly in order to enable real-time systems and may need routines that require less computational power. This happens because a cloudlet node could be implemented both with resourceful GPUs (closer to the network layer [2.2]) or with resourceless microcontrollers (closer to the devices layer [1]) [Xiao et al., 2019]. Also, by bringing data processing and management steps to the edge, data becomes more susceptible to external attacks due to the large surface of end-devices and their lack of protection routines, which turns security and privacy requirements important bottleneck problems to enhance edge computing applications [Yahuza et al., 2020; Xiao et al., 2019; Aversano et al., 2021].

3.2 IoT and Edge Applications Requirements

With the constant development of IoT solutions to the connected society, we have everyday even more modern applications based on seamless communication between a huge variety of devices that connect and exchange great amounts of information in a quick way. Because of that, it is important to always develop new technologies and improve existent ones to meet the needs of recent applications. Current IoT environments require stable communication channels, proper data storing and processing, management of energy consumption, real-time data analysis, security and privacy of the data in order to enable reliable, scalable and secure applications [Nikoui et al., 2021; Yu et al., 2017].

Recent studies point that cloud-centric solutions are no longer able to meet all modern IoT applications requirements. That way, the edge computing is a recent and promising paradigm in the IoT world to confront its constraints. The architecture is interesting to enable real-time services by reducing the overall latency on communication processes and promote management of data and security tasks on the edge [Yu et al., 2017] [Merenda et al., 2020]. Nevertheless, it is important to be aware of the requirements and challenges regard the designed IoT applications and the edge devices themselves,

considering hardware and software limitations, in order to build the best possible solutions. Some of these important requirements are discussed below in the chapter.

3.2.1 Communication

A great part of IoT applications require tasks to be executed in real-time scenarios, such as transportation, healthcare, smart home and object detection [Anandhalli and Baligar, 2018; Wang et al., 2018b; Devi and Kalaivani, 2020]. Thus, developers should consider network properties and available communication protocols to achieve the desired performance, avoiding waste of energy caused by exchanging huge amounts of data through the network.

Some used protocols for IoT applications are:

- Bluetooth, in particular Bluetooth Low Energy (BLE), which is a known wireless technology that enables transferring data with a good range, minimum energy consumption and a good transfer rate of 2 MB/s [Merenda et al., 2020];
- IEEE 802.11 pattern comprises wireless network solutions, which, today, can attend low-power and long-range mechanisms with high speed [Merenda et al., 2020];
- IEEE 802.15.3 is the ultrawideband (UWB) radio technology that utilizes more than 500 MHz of frequency to provide fast, short-range and high-bandwidth communication in a limited area. It is useful to enable local and real-time applications [Capra et al., 2019];
- ZigBee and Z-Wave are known communication patterns based on IEEE 802.15.4 standard suitable to low-power, low-rate devices and wireless sensors communication, widely applied on smart home solutions [Merenda et al., 2020];
- LoRA networks are low-power wide-range solutions (LPWAN) suitable for tasks that demand communication between devices remotely located [Khutsoane et al., 2017], like in mining and agriculture fields;
- 5G is a recent pattern of wireless communication that promises to revolutionize the area, providing reliability and latency improvement as well as energy consumption reduction [Capra et al., 2019];
- NFC, or Near-field Communication, is a set of protocols based on radio-frequency identification technology (RFID) that enable low-power and fast communication between devices in a short range [Lazaro et al., 2018];

Solutions like compressing the data, performing ML model offloading, implementing distributed algorithms and tasks management of algorithms combined with a proper communication protocol should help to achieve applications that enable real-time responses [Gao et al., 2021; Alonso et al., 2020; Castano et al., 2020].

For communication constraints, Corneo et al. [2021] point out that edge computing strategies can improve network latency up to 30% in relation to cloud-centric systems. It is hard to define an optimal interval for latency response in IoT solutions because it highly depends on the purpose of the application. Chen et al. [2017], for instance, define with an empirical experiment that a latency response of 300 ms to 500 ms is ideal to perform real-time recognition task on video frames using edge computing, which can be used as a basis to define an acceptable latency for IoT-edge applications.

3.2.2 Energy Consumption

Another important limitation that may appear in IoT applications and employed devices is the energy consumption. IoT systems are usually required to keep working indefinitely, always collecting data and performing actions in response, and it is not always possible to be connected to appropriate energy sources, like in drones and cameras. Cloud servers are unlimited in that aspect, but that does not always apply to the end-devices.

Decision making and data management tasks with AI algorithms normally require high processing, which can cause energy drain, and employed hardwares for IoT solutions can be small and powered with low capacity batteries. Thus, the system should be designed in a way to maintain low-power consumption and guarantee the correct operation of the application. Edge computing implementation can help in this scenario because it prevents the transmission of data to the central clouds, which is an energy-hungry operation [Capra et al., 2019]. Edge devices, however, may also suffer from battery limitation.

Offloading machine learning techniques help to balance energy drain from nodes by supporting parallelism on algorithm, like in Ray [2018], where authors use a fog-cloud architecture that enables the energy consumption reduction. It is also possible to design duty cycling, a paradigm which prevents the device of being in function all the time. The peripheral is put in sleep mode and wakes up only when it is necessary to perform a task [Sun et al., 2019] [Kolomvatsos, 2021]. Most CPUs (Central Processing Units) also have a deep-sleep mode, similar to duty cycling, but the wake up signal is controlled by an internal clock. Other methodologies can be used on physical layer, such as clock gating, which stops the clock on portions of the hardware where no task should be performed at

the time; and power gating, by disconnecting supply voltage from the system when it is not required [Capra et al., 2019].

This energy consumption requirement is also intrinsic to the previous communication topic (Section 3.2.1). By using low-power short-range protocols such as NFC and BLE it is possible to reduce the overall energy consumption and guarantee longevity of the devices as shown by Lazaro et al. [2018]. Authors, in this case, provide a study of low-power and energy harvesting sensors with the NFC technology as an enabler for green IoT applications.

3.2.3 Computational Resources

The computational resource is another important constraint to be considered when using machine learning architectures in IoT applications, since those algorithms are normally costly and require high processing. Choosing some ML algorithm should consider available resources of employed hardware and the possibility of applying methods to reduce consumption of tasks.

Table 3.1: Comparison between some microcontrollers (MCUs) and microcomputers applied in IoT systems. The available resources should be carefully considered when designing the desired prototype. Raspberry is the microcomputer with best resources, and, indeed, is employed in many works, but is also the most expensive one.

Device	RAM	Flash Memory	Processor	Clock
Arduino Due	96 kB	512 kB	ARM dual-core 32 bits	84 MHz
ESP32-D2WD	520 kB	2 MB	Xtensa dual-core 32 bits	240 MHz
Raspberry Pi 4	4-8 GB	-	Cortex-A72 quad-core	1.5 GHz
Raspberry Pi 3	1 GB	-	Cortex-A53 quad-core	1.4 GHz
SparkFun Apollo3 Blue	384 kB	1 MB	ARM dual-core 32 bits	48 MHz
Adafruit EdgeBadge	192 kB	512 kB	ATSAMD51J19	120 MHz

Some recommended devices to be used on IoT solutions, specially for edge computing applications, are Arduino series, Espressif controllers (ESP32 series), Raspberry Pi, SparkFun EdgeBoard and Adafruit EdgeBadge [Merenda et al., 2020], but most of them have scarce resources, such as memory and processing power. Wang et al. [2018a],

however, points the possibility to employ resourceful Graphics Processing Unit (GPUs), as computational nodes to perform data analysis, specially in recent mobile applications. Table 3.1 compares the characteristics of cited hardwares, intending to analyze overall resources available when creating an IoT prototype.

As presented in Chapter 2, allying edge-cloud services could enable deploying more powerful algorithms to the edge, since virtual machines and cloud servers usually are resourceful [Gao et al., 2021; Yu et al., 2017]. Also, Lin et al. [2020] and Sun et al. [2019] propose developing tiny and distributed architectures of employed neural networks, with less layers, in order to reduce the required processing from AI architectures. Another solution that has been explored in this scenario is to run virtualized or containerized operating systems (OS) to emulate the required hardware resources and create high performance applications [Capra et al., 2019; Wang et al., 2018a].

3.2.4 Data Management

A constraint usually associated with IoT systems is the management of collected data. When peripherals detects external signals, they are not necessarily (and probably not) encoded as the way the software needs it to be. Usually, data should be modified to fit the required patterns and steps should be take to ensure its security, privacy, accuracy, reliability, availability and usability [Sikwela, 2012]. The applications should also be able to gather intelligence from the data, discover patterns and enable actions in an automatized way. For many of those requirements, ML and DL architectures are being used more and more with exciting results [Cui et al., 2018; Mahdavinejad et al., 2018].

Specifically in edge architectures, transformation and governance of data should be performed directly on the edge node. For instance, in Alonso et al. [2020] authors design the system with a transformation step that converts 3D point cloud data to 2D representation to perform the learning technique, and return 2D to 3D information after the segmentation is done. Thus, ML algorithms could also help in treatment of data. As seen in Castano et al. [2020], edge devices highly depend on reliability of its peripherals, which shows that data governance is an important constraint to be considered, comprising integrity, quality, usability and consistency of the acquired information.

3.2.5 Security and Privacy

Security is a particular criteria that is extremely important to consider, specially when treating with sensible data, like health information, or in smart home scope, where devices deal with owners' private data. Recent IoT researches still need improves on the security field, due to the lack of standardization of protection routines on those systems and the huge attack surface of the environments [Alqahtani et al., 2020; Laguduva et al., 2019]. Xiao et al. [2019] identify that poor security and privacy are the most outstanding constraints that prevent more IoT and edge systems from being deployed.

Considering the underlying limitation of IoT devices, it is important to ensure features such as secure communication, security and privacy of data, identity management and malware detection. Some relevant challenges within the security and privacy requirement for IoT are: authentication and control of access within the applications; prevent distributed and coordinated attacks (DoS and DDoS); anomaly and attack detection on network traffic; enable malware analysis to gather knowledge about the current trends and build protected systems [Hussain et al., 2020; Aversano et al., 2021].

Edge devices normally compose environments with minimum protection, once they lay in the border of the system [Laguduva et al., 2019]. As an advantage, usage of edge nodes prevents data to be continuously exchanged across the web, decreasing chance of remote attacks. But a secure system should also be thought and implemented on node's architecture. A current trend shows many works that propose building security and privacy systems with machine learning algorithms to be deployed on the edge, such as Kozika et al. [2018], Yahuza et al. [2020], Xiao et al. [2019] and Zeyu et al. [2020]. As seen, tasks management methods enable developing of learning processes without directly dealing with client's data, ensuring privacy of information [Gao et al., 2021] [Sharma et al., 2021].

Aversano et al. [2021] also summarize papers that propose the application of deep learning strategies to the emerging field of IoT with the promise to accurately identify threats and potential risks and apply classification techniques to help on keeping the systems confidential and incorruptible. Those techniques are being widely researched to help with the constraints of privacy preservation, detection of vulnerabilities, protection of intellectual property, authentication and authorization and anomaly and attack detection. The ML strategies proposed to tackle the studied requirements are usually applied on the network layer (edge) or the data management layer. Among the researched works, there is also a need to improve the search for algorithms with good performance and that enable real-time operations, works well documented, with suitable datasets and well interpreted results.

Regarding the anomaly and attack detection field, which is the main focus of

this work, efforts are done to early detect abnormal data on systems, which can cause malfunctioning of a whole environment, and spread of malware information. The last type comprises botnet data, which are bots created to attack systems and flood the network with spam, infect other resources and steal data. These are better explained on the further Section 3.3. According to [Alsoufi et al. \[2021\]](#), the most used DL strategies to deal with the attack detection task currently are the CNNs, the Autoencoders and the LSTMs.

3.3 Botnet Attacks on IoT Environments

Botnets are networks of computers that are assembled to perform continuous attacks on several devices and create a net of infected machines. Those attacks are usually coordinated by the attackers, known as botmasters, as huge DDoS incursions, which explore vulnerabilities of the systems to invade and gain control of targeted devices. Attackers can, then, use compromised hosts to send infected packets to other victims, flood server's networks, steal information, gain prohibited access and prevent systems to proper work [[Hoque et al., 2015](#)].

With the recent grow of IoT solutions and its growing importance on global society, those environments became one of the most targeted systems by the botmasters. Despite its importance, IoT environments are also very susceptible to malicious attacks as explained on Section 3.2.5 and can lead to the disrupting of important autonomous mechanisms, such as in smart industries (machines, processes, enterprise computers), smart homes (door locks, personal computers and information) and smart healthcare (surveillance cameras, personal monitors).

[Kolias et al. \[2017\]](#) and [Pa et al. \[2016\]](#) present some recent known botnets such as Mirai, Gafgyt (or Bashlite), Hajime and BrickerBot. With the knowledge on how those malwares work, it is possible to respond with efficient mechanisms to protect the environments and prevent them to be infected by malicious information. In order to do that, many authors have studied the botnets and created datasets that simulate real botnet incursions to help on researching and developing new solutions for the problem. Some of them are the N-BaIoT [[Meidan et al., 2018](#)], the Bot-IoT [[Koroniotis et al., 2019](#)], the NSL-KDD [[Tavallaee et al., 2009](#)] and the CTU-13 [[Garcia et al., 2014](#)] datasets. The botnet datasets employed on the present study are presented on the following section.

3.3.1 IoT Datasets

This section discusses some botnet datasets developed for the Internet of Things context that are used on the case study chapters of this work (Chapters 5 and 6). The datasets gather information of different devices placed on monitored IoT environments that are affected by malware data. The data can be used to study the behavior of botnet information within the environments and develop automated solutions to early detect and prevent infection.

3.3.1.1 N-BaIoT

The N-BaIoT dataset is created by [Meidan et al. \[2018\]](#) following the methodology provided by [Mirsky et al. \[2018\]](#) to study the behavior of botnet attacks on the IoT context. The database contains statistical information of some features on the traffic channel of nine different IoT devices: Danmini and Ennio doorbells, two Provision and two SimpleHome cameras, a Samsung webcam and a Phillips baby monitor. The devices were monitored when operating under normal conditions (benign traffic) and when receiving malicious information. In total, 115 features were extracted from the channels, with different values of acquisition time: data gathered with 1 min, 10 s, 1.5 s, 500 ms and 100 ms (L0.01, L0.1, L1, L3, e L5, respectively). The possible types of features are explained in the Table 3.2. For instance, a possible feature of the dataset would be **HH_L5_mean**, which represents the statistical feature **mean** obtained with **100 ms** of acquisition time to the **Channel** traffic data (between two hosts).

The methodology followed by the authors is summarized by the following steps:

- Collect the traffic packages generated by the observed device and create a raw binary with this information. The obtaining process is conducted with five different time intervals: 100 ms, 500 ms, 1.5 s, 10 s and 1 min
- Determine some information obtained with the binary file, like the package size, arrival time, network addresses and others
- Extract from the file 115 statistical features (mean, variance, etc.) related to the traffic information for each of the described time intervals

Table 3.2: Description of each particular statistical feature on the N-BaIoT dataset, according to the works of Meidan et al. [2018], Mirsky et al. [2018] and Alqahtani et al. [2020].

Feature Type	Acronym	Details
Host-IP	H	Traffic data when it comes from a specific IP address.
Host-MAC&IP	MI	Traffic data when coming from specific IP and MAC addresses.
Channel	HH	Traffic is sent between the packet's source and destination IP.
Socket	HpHp	Traffic is sent between the packet's source and destination TCP/UDP sockets.
Network jitter	HH_jit	A subtype of the Channel category, which considers delays between outgoing traffic packets.

3.3.1.2 Bot-IoT

The BoT-IoT [Koroniotis et al., 2019] is another dataset employed on many studies that deal with the attack detection and classification problems [Popoola et al., 2022]. To create the database, authors assemble a testbed environment to simulate operational IoT services on five devices: a weather station, a smart fridge, a smart thermostat and motion-activated lights. The devices are connected to a web server in order to gather the traffic information when receiving both benign and malicious data. Authors were able to gather 29 relevant features of the channels, which were used to generate 14 more features, totalizing 43 features. With this technique, authors intend to improve the predictability for the classifiers.

The next Chapter introduces the concepts of machine and deep learning for IoT environments, considering the advantages those strategies can bring to tackle the recent problems and requirements that were described in this Chapter, the modern solutions that are being researched and current trends.

Chapter 4

Machine Learning Architectures for Internet of Things

Machine learning comprehends a series of techniques that allows data extraction, analysis and prediction. It is a specific area of artificial intelligence, which undertakes to understand and mimetize the human behavior with machines. It also comprises the deep learning area, responsible for generating algorithms that are able to learn with amounts of data similarly to the human brain. It is dispensable to say how important those technologies are to progress of society. Also, they are hugely applicable in IoT solutions, creating devices that are aware of the data received and can make decisions based on them. For example, [Devi and Kalaivani \[2020\]](#) develop an IoT-based platform with machine learning, capable of predicting cardiovascular diseases according to patient's ECG; [Sun et al. \[2019\]](#) show a case study in which a deep learning algorithm (DL-based transfer learning) is employed for image processing in industrial procedures; [Li et al. \[2018\]](#) conduct a research about deep learning architectures applied in edge computing; there are also surveys about the implementation of machine learning in IoT's power systems [[Farhoumandi et al., 2020](#)] and video processing [[Anandhalli and Baligar, 2018](#)], to cite some.

Although edge computing implementation helps with network's latency and performance, there are some limitations to be considered. The edge devices usually have reduced memory and computational capabilities, since they can be low-power mechanisms [[Yu et al., 2017](#)]. The develop of machine learning techniques should be done carefully, considering optimization and artifices to decrease algorithm's computational cost, like pruning, reducing the number of parameters, lowering precision and perform offloading techniques [[Merenda et al., 2020](#)] [[Gao et al., 2021](#)].

Section 4.1 discusses the different classes of the machine learning area, dividing the algorithm types based on the supervision level of the possible architectures. Section 4.2 discusses recent uses of machine learning techniques in IoT environments, including edge computing systems. Finally, Section 4.3 treats about the hyperparameter optimization, an important step to be done when creating ML models. In particular, it is also presented the automated optimization applied on the experiments conducted in this work, the Bayesian

Optimization.

4.1 Machine Learning Classes

Machine learning architectures usually are separated in three main classes: supervised, unsupervised and reinforcement algorithms. In supervised techniques the machine receives both the inputs and outputs of the data it should analyze and maps a function from the input to the output. Basically, it learns the characteristics of data by comparison with the information provided on labeled data. On the other hand, only inputs are provided to unsupervised learning. It means that the machine should create its own model for the inserted function, learning more about the data by itself. To the reinforcement learning, the goal is putting the machine in a decision making situation, usually in a complex environment, in order to maximize the number of hits or rewards (whether the decision is right). Additionally, semi-supervised learning combines both supervised and unsupervised characteristics, providing some classified labels as input, but guaranteeing that the model can be generated in an unsupervised manner [Cui et al., 2018; Samie et al., 2019].

Supervised learning is simpler to implement and its results are accurate; the unsupervised algorithms are complex to develop and the accuracy is lower, but they do not depend on labeled data to learn; reinforcement techniques, still, grant high accuracy, but are difficult to code and are commonly used in the context of complex environments. Choosing between these types normally depends on the goal of the application. Table 4.1 shows a summary of the taxonomy of machine learning classes, whether they comprise deep learning techniques or not, as well as its technical features, efforts and possible applications.

It is also possible to summarize some common types of problems in the machine learning. Classification problems should have its data classified, by predicting the label that each input should be assigned to. Regression techniques are used to predict the value of an input based on variable predictors. Clustering strategies models groups of variables based on the similarity and proximity between them. Association techniques can be used to detect interesting relationships hidden in large datasets. Finally, generative architectures are able to reproduce the input and create whole new outputs that have similarities with the given input [Zaki and Meira, 2020; Samie et al., 2019; Mahdavinejad et al., 2018].

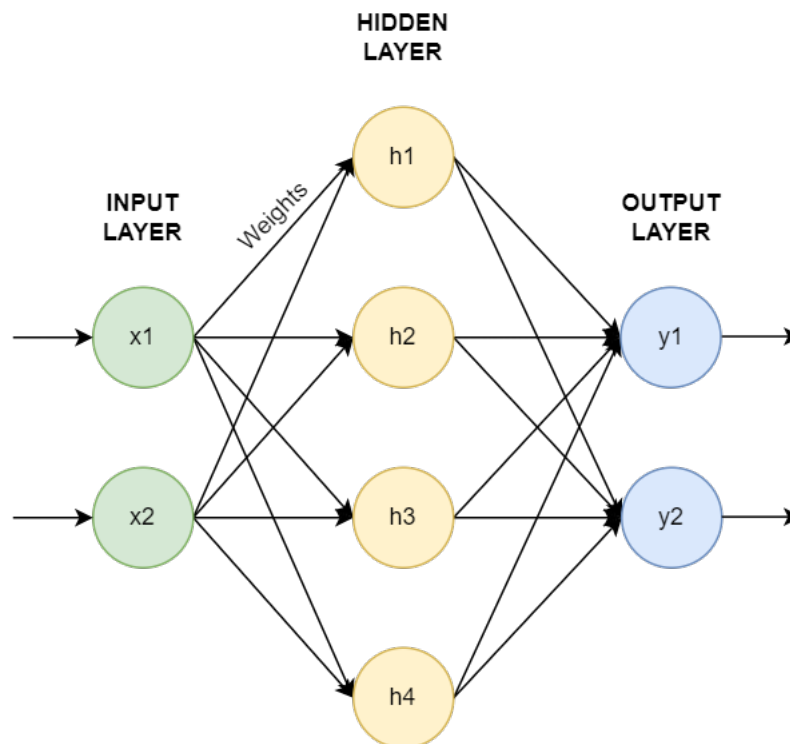
Table 4.1: Ordinary classification of types of machine learning algorithms, as well as its technical features, pros, cons and example techniques. The given examples are assigned based on the common use in literature.

Class	Technical Features	Pros (+) and Cons (-)	Techniques
Supervised Learning	Inputs and outputs are provided, so the algorithm is able to learn by mapping.	<ul style="list-style-type: none"> + It is easy to implement + Memory cost is usually low - Requires labeled datasets 	<ul style="list-style-type: none"> • Random Forests • Linear Regression • CNN
Unsupervised Learning	Unlabeled data is provided and the machine learns by analyzing only the input dataset.	<ul style="list-style-type: none"> - Requires more computational resources - Accuracy is usually lower + Doesn't need pre-generated labeled data 	<ul style="list-style-type: none"> • K-Means • GMM • Isolation Forest
Reinforcement Learning	The algorithm attempts to find an optimal solution with trial and error techniques.	<ul style="list-style-type: none"> + Highly applicable in robots and games, on tasks that depends on decision making + It guarantees high accuracy - Harder to implement 	<ul style="list-style-type: none"> • Markov Decision Problems • Q-Learning
Semi-Supervised Learning	With this method, the algorithm receives labeled and unlabeled data to execute the training.	<ul style="list-style-type: none"> + Useful when analyzing the raw data is difficult + It can be done with small quantities of labeled data - Lower accuracy 	<ul style="list-style-type: none"> • GAN • Fuzzy C-Means • VAE

Deep learning is a particular subfield of machine learning, since its algorithms intend to learn by analyzing data, but utilize the so called artificial neural networks with plenty layers to generate behaviour in a human-like manner. An artificial neuronal

network is a graph-based data structure often separated in layers, in which the information flows and it is constantly updated and analyzed. Each node of the graph can be viewed as a neuron, the biological structure that sends information as an electrochemical signal (synapse) through the network. The process of learning in our brain occurs because the emitted signals can be either excitatory or inhibitory, causing the neurons to adjust the synaptic strengths according to each case. If the synapse is excitatory, the information is more likely to be transmitted; on the other hand, inhibitory signals generates less potential to be broadcasted [Zaki and Meira, 2020].

Figure 4.1: The structure of a neural network with its nodes, layers and weight values that transform data while it passes through the steps of the network [Zaki and Meira, 2020; Gron, 2019].



Source: Elaborated by the author

Like neurons, the nodes of the artificial networks have weights, which mimetize the so called synaptic strength. They act like processing structures, receiving external (or input) signals as a weighted summation and generating outputs by applying an activation function over the input. The persisted outputs are similarly analyzed on the subsequent layers of the net until its end. In supervised structures, the network also computes the deviation between these outputs and the true labeled response, returning an error that causes to change the weight of the node. Large errors generate huge modifications of the weight and vice-versa. Still, in unsupervised learning, weights are grouped by similarity via clustering methodologies [Zaki and Meira, 2020]. Figure 4.1 shows the representation

of a neural network, with the input layer, the hidden layer, where the learning operations are performed, and the output layer that represents the final results.

Supervised and semi-supervised neural networks are helpful to approach classification and regression problems, or situations that require a labeled input data to perform the training. Unsupervised nets could be used to generate new models for an input or determine certain features (smooth variable function) from time series functions as well as clustering situations. Reinforcement strategies are interesting to develop self-supervised solutions that should perceive and interpret environment information to take actions. Despite the different classes, the aim of a neuronal network is to minimize the error and return a model that fits and solve the proposed problem. With the generated model, it would be possible to run it with any inputs from the designed proposition and receive an answer. Whether the answer would be correct or incorrect depends on the accuracy of the model, which determines the percentage of hits over the test set.

In machine learning area, there are two main drawbacks that could occur on developed models: overfitting happens when the model is trained to solve only one kind of problem and the generated outcomes have great variance; underfitting is the analog problem, which occurs when model is poorly trained and generates biased predictions. Variance refers to the variability of the model when different sets are applied, and bias is related to the generation of wrong answers during training or testing processes [Gron, 2019].

4.2 Applying Machine Learning Techniques in IoT Environments

As explained in the previous section, the use of a machine learning technique depends highly on the purpose of the application and the type of the problem (e.g., clustering, classification, regression and so on). Some characteristics of the different classes are highlighted to help on defining the best approaches to some well known cases. In this section, the goal is to discuss the application of the exposed techniques within IoT architectures, including in edge computing devices, considering its challenges and potential improvements. This section also provides relevant examples of use of combined IoT-machine learning applications to tackle recent problems and constraints of the technologies in order to enable reliable, secure and efficient systems.

4.2.1 IoT Solutions with Machine Learning

IoT environments are usually restricted systems. One of their relevant characteristics is the heterogeneity of the peripherals that compose them, which have their own communication, privacy and hardware standards. They are ensembled in highly scalable environments, with a massive number of connected devices. These devices require inter-connectivity with local or global information and close proximity communication in an ultra-reliable and low-latency way. Because of energy consumption, IoT machines are usually low-power devices, with constrained capabilities [Hussain et al., 2020; Capra et al., 2019].

Considering the presented architecture on Chapter 3, however, many IoT solutions today apply their routines on the data management layer, that can be deployed into cloud resourceful environments, leaving the low-power devices to perform simpler algorithms. It is also proven that ML algorithms can help to achieve systems common requirements, such as node selection, reduction of energy consumption, management of communication and security [Aversano et al., 2021; Hussain et al., 2020; Laguduva et al., 2019]. Because of that, applying machine learning strategies on IoT environments is increasingly becoming a popular solution. Some of the areas that take advantage of machine learning and IoT enabled technologies are listed below.

4.2.1.1 Applications

Personal healthcare is one of the most growing applications in an IoT perspective. Some works show that by using IoT devices to monitor patients combined with deep learning strategies it became possible to early-diagnose diseases and help on medical differentiation [Bolhasani et al., 2021; Ghimire et al., 2020]. Pradhan and Chawla [2020] summarize several works considering mostly supervised and unsupervised architectures that blend the machine learning solutions to IoT environments in order to enable lung cancer detection. Tuli et al. [2020] develop a supervised novel framework with a DNN architecture to monitor patients with heart conditions, considering the need to achieve low-latency responses for low-power devices and real-time systems. Authors employ fog computing paradigm in this case to tackle the requirement.

On the smart cities scope, IoT environments can be used to solve many community issues that are related to pollution, traffic, energy distribution, public transportation, public security and communication [Atitallah et al., 2020]. Works like Rashid et al. [2019]

and [Popa et al. \[2019\]](#) provide supervised solutions with LSTM models to enable smart distribution of energy consumption in houses and cities. Supervised RNN and LSTM models are also employed to deal with public traffic data, predictability of automobiles routes and crowd density [[Liu and Shoji, 2018](#); [Atitallah et al., 2020](#); [Song et al., 2016](#)].

The industrial field (IIoT) takes advantage of IoT technologies not only to automate processes, but also to keep reliability, privacy and safety of data, which is a very sensitive information in this scope [[Arachchige et al., 2020](#); [Zolanvari et al., 2019](#)]. [Sun et al. \[2019\]](#) also bring a perspective on data offloading to be applied on imagery processing for industrial environments, which can be used alongside fog or edge environments. Talking about security, this is a field of main concern on many recent researches as shown on Section 3.2.5, not only because it enables protection of data and underlying environments, but also because it can be employed in scenarios related to all of the discussed fields: healthcare, smart cities, smart homes, industrial IoT and so on [[Atitallah et al., 2020](#); [Bolhasani et al., 2021](#)].

4.2.1.2 Requirements

Machine learning algorithms can be helpful to solve many recent IoT related challenges. That is because of its capacity to learn from non-linear dynamic information and predict the most probable outcomes [[Mahdavinejad et al., 2018](#)]. Moreover, deep learning strategies enable even more accurate and reliable results, but with a software and hardware consumption cost.

Thinking on the requirements of IoT systems, the data management and data standardization problems can be easily targeted by ML and DL solutions: those architectures require huge amounts of training data and can learn from non-linear relations [[Hussain et al., 2020](#)]. IoT systems should also be capable of performing decision making and, in that scenario, AI solutions can provide the "intelligence" needed to achieve this requirement in a reliable way.

On the communication and energy consumption fields, ML solutions are used to perform resource allocation and task scheduling. There are solutions to help on deciding which node or gateway should be used within an environment, determine transmission rates of data and group low-power nodes to achieve better communication latency by applying supervised, unsupervised and reinforcement strategies [[Samie et al., 2019](#); [Cui et al., 2018](#)]. Traffic profiling techniques with ML architectures are also implemented to obtain insightful information about traffic patterns, identify rare-case events, such as in [Janjua et al. \[2019\]](#), and improve application performance. The use of edge computing

systems with deep learning strategies has also been shown as a possible solution to enable real-time environments with low-power devices, specially because it prevents exchanging large amounts of data with cloud servers [Tuli et al., 2020].

Proper security of IoT environments has also been enabled by applying machine learning techniques to perform tasks like early detecting intruders or malware information, getting authentication and authorization for users in an automatized way, discovering patterns of possible types of incoming attacks, and so on. Borges et al. [2022], for instance, proposes a reliable model using a simple Isolation Forest ML algorithm by applying proper time series transformation to identify malign data, which provides great accuracy results. This trend can be also seen on the works of Aversano et al. [2021], Laguduva et al. [2019], Hussain et al. [2020] and Alsoufi et al. [2021].

4.2.2 Edge Computing Solutions with Machine Learning

As observed in Chapter 2's articles and previous discussion about applications requirements, the great challenge of machine learning within edge devices is the computational cost of such techniques and the inherent resource limitation of the nodes. Recent studies, however, show that it is possible to successfully deploy and execute machine and deep learning operations on edge or cloudlet nodes [Samie et al., 2019; Wang et al., 2018a; Cui et al., 2018; Capra et al., 2019].

Some modern works propose the management of ML algorithms with offloading methods [Serra et al., 2021; Gao et al., 2021; Kolomvatsos, 2021; Wang et al., 2018b; Roy et al., 2022], which appear to deliver interesting results. This technique intends to pre-train costly models in resourceful environments, usually the cloud, and continuously upload the model to the edge nodes where the learning tasks can be done.

There are also works that bet in developing small architectures of deep learning algorithms to deal with the problem. Lin et al. [2020] propose a tiny framework (TinyMCU) to apply neural networks on microcontrollers scope in an energy-efficient optimized manner. Some of the techniques employed to achieve this are pruning (deleting parts of the net that does not contribute to the model) and quantization (removing redundancy and thus, complexity of the code), achieving 70.7% of accuracy on the experiment. Alonso et al. [2020] also develop a compressed neural network architecture, in particular, a FCNN. The problem presented is the semantic segmentation of 3D images obtained by LiDAR (Light Detection and Ranging) sensors, which require well designed resourceful nets. They achieve a fast and accurate framework and show its advantages regarding the state-of-art modern works.

4.2.2.1 Applications

As seen in literature, the most utilized class of machine learning within edge computing is the supervised learning. Searching for "supervised learning edge computing" on Google Scholar returns about 365.000 articles. On the other hand, the same search with unsupervised learning generates 204.000 results, while reinforcement learning and semi-supervised learning comprises only 136.000 and 121.000 papers, respectively. Indeed, supervised learning is the most targeted area on recent machine learning studies, due to its advantages as high accuracy of the models and relatively simple implementation [Wang et al., 2020]. As it is a targeted area, there are plenty consolidated algorithms and architectures on state-of-art studies, with massive support on popular languages, like C++ (TensorFlow and Microsoft Cognitive Toolkit) and Python (TensorFlow and Keras libraries). The use of classifiers, regressors or deep learning techniques depends highly on the purpose of the scope. For instance, image processing and anomalous detection are fields in which the use of deep learning overcomes other methods, in particular CNNs and its derivatives. Nonetheless, classification algorithms usually provide good results and require less computational processing, being an interesting option to systems that lack of processing resources.

It is always important to define the purposes of the project in order to choose the best available approach, and occasionally blend it with other techniques. Revised reinforcement studies utilize supervised and unsupervised techniques to support decision making actions on edge computing scope [Castano et al., 2020; Ji et al., 2021; Chen et al., 2019]. Efforts on security and privacy areas comprise the use of extreme machine learning algorithms, specially semi-supervised methods that may be ran with fewer parameters, which can make the model run faster [Yahuza et al., 2020; Laguduva et al., 2019; Rathore and Park, 2018].

Among studied papers, unsupervised learning is commonly applied in clustering problems, being widely used on offloading solutions to perform grouping task. DNNs are also widely employed on this scope [Wang et al., 2018a; Sharma et al., 2021; Kolomvatsos, 2021; Shakarami et al., 2020]. Last but not least, reinforcement learning is a methodology that engenders great solutions, but it is the most hard and costly to implement. Nonetheless, literature shows that the field of self-aware, real-time and automatized systems is an interesting area to apply reinforcement learning, in particular Q-Learning, useful to solve decision making situations and build networks that could perform self-improvement [Mahdavinejad et al., 2018].

4.2.2.2 Requirements

In general, it is possible to conclude some characteristics of the recent use of machine learning in IoT and edge computing devices. Applying those algorithms on cloud or remote servers is easier, due to their resourceful components. Edge nodes usually have less memory and processing resources. For instance, a recent Raspberry Pi 4 has at most 8GB of RAM, while Google can provide virtual machines with 896GB of available memory¹; usually, ordinary microcontrollers present at most 1GB of RAM and limited flash memory. Although recent microchips with better capacities are being continuously designed, machine learning algorithms are costly, comprehending the train, test and validation steps for developing a model.

Turns out that the option of designing tiny and lightweight architectures is interesting in order to constraint the processing cost and keep high accuracy [Lin et al., 2020] [Sun et al., 2019]. Also, the discussed offloading techniques help to assign available nodes for processing steps, guaranteeing parallelization of the flow and achieving the best possible performance. In some of the studied architectures, the offloading process is also done together with cloud servers, generating an interesting edge-cloud application, but always considering the acceptable limitations of the model [Gao et al., 2021; Wang et al., 2018b]. In this scope, another possible solution with hybrid architecture is to train the algorithm on cloud, or disposable node, and continuously offloads and updates the obtained model for punctual use on the edge. Such a methodology is applied by Khani et al. [2020] to build a real-time adaptive inference IoT prototype based on videos recorded by a cell phone. Authors claim that this approach avoids overfitting by performing few updates on the model and delivers instant segmentation at 30 frames-per-second.

Another constraint related to communication latency can be assigned to edge computing environments. Although edge implementation reduces the time spent to generate responses, it is necessary to ensure that the routines on the cloudlet nodes can run quickly and enable the pursued latency reduction. Works like Corneo et al. [2021] and Chen et al. [2017] show that it is possible to reduce the system latency while using machine learning algorithms on the edge and keep high accuracy values.

It is also important to secure the communication channels that passes through edge nodes to ensure the protection of the data and the environment itself. Edge computing implementation add a protection level because it prevents data from being continuously sent to the cloud. However, edge devices usually have minimum defences against potential threats and are easy targets for physical and network attacks [Laguduva et al., 2019; Yahuza et al., 2020; Xiao et al., 2019]. ML and DL methods are being successfully

¹Link: [Google Compute Engine](#)

applied in recent works to secure the channels by performing early attack identification, proper authorization handling and protection of the data as explored by [Xiao et al. \[2019\]](#) and [Yahuza et al. \[2020\]](#). [Roy et al. \[2022\]](#) present a two-layer fog-cloud architecture with a machine learning algorithm capable of performing intrusion detection while keeping high accuracy and reducing the overall energy consumption. A similar work is proposed by [Gopalakrishnan et al. \[2020\]](#) and [Tian et al. \[2020\]](#), which employ ML techniques to perform anomaly detection of data on an edge node level. [Alsoufi et al. \[2021\]](#) points out in a recent study that CNNs, LSTMs and Autoencoders are the most employed architectures to deal with attack identification: the first two are supervised techniques while Autoencoders can be designed in a semi-supervised or unsupervised way.

4.3 Hyperparameter Optimization for Machine Learning Strategies

The main goal of the ML strategies is to learn trends and tendencies from some portion of data and reproduce its behavior to new, unknown data. The most recent algorithms, specially on the deep learning field, fit the learned models by using the studied architectures. However, different models are created for each type of problem, depending on the input data, its type, size and internal dynamics. To reach the most appropriate solutions, it is usual that the model should be filled with a set of hyperparameters that changes the dynamic of the algorithm [[Feurer and Hutter, 2019](#); [Gron, 2019](#)].

Choosing the best set of hyperparameters for a model is not an easy task. To achieve this assignment manually it would be necessary to search among a wide range of possible parameters, test the best outcomes and despise the ones that cause low results. Some recent efforts are done to enable algorithms that can automatically test many sets and converge to an optimal solution to minimize the loss function in a quicker way.

The problem that should be solved by the hyperparameter optimization is stated in Equation 4.1: given a space of possible hyperparameters $\Lambda = \Lambda_1 \times \Lambda_2 \times \dots \times \Lambda_n$. Given a dataset D , the goal is to find a combination of λ^* vector hyperparameters capable of minimizing the model loss L generated by some algorithm A [[Feurer and Hutter, 2019](#)].

$$\lambda^* = \arg \min_{\lambda \in \Lambda} L(A, D_{train}, D_{val}) \quad (4.1)$$

Some known optimization strategies are the grid search, random search, bayesian optimization, gradient-based optimization and evolutionary algorithms [[Feurer and Hut-](#)

ter, 2019]. The approach chosen to optimize the models created on this work is the bayesian optimization.

4.3.1 Hyperparameter Types

The hyperparameters that are tuned on the experiments presented on Chapters 5 and 6 are the number of epochs, the learning rate and the batch size of the models. One epoch in the machine learning context represents one complete pass of the training data on the fitted model. The higher the total number of epochs, more times the model is trained with the dataset. In many cases, a high number of epochs guarantees better results. However, a large number of epochs can cause overfitting of the model, which is an undesirable behavior. Because of that, tuning this parameter correctly shall ensure that the model will achieve the best possible results and prevent from overfitting.

The learning rate is an important parameter that defines the behavior of a deep learning solution. It represents the size of the update step that the optimizer should perform on the training phase, balancing the amount of attribute modifications in a neural network on each iteration. A lower learning rate usually causes the model to take more time to train, because it indicates that minor update steps should be done. However, a big learning rate can prevent the algorithm to properly learn the desired attributes. The optimization method should also balance these two characteristics to find the best value of learning rate.

Finally, the batch size is the size of samples processed at the same time on the training step. The higher the batch size, the faster the algorithm can pass through the training. This number should also be chosen carefully because it can impact the efficiency of the training phase [Gron, 2019].

4.3.2 Bayesian Optimization

The bayesian optimization is a black-box iterative optimizer that is mainly constructed by two components: a surrogate model and an acquisition function. The acquisition function uses a predictive distribution of the probabilistic model to estimate the objective function and trade off exploration (parameters for each the outcome is uncertain) and exploitation (parameters expected to be closer to the optimum value) features.

The surrogate model iteratively updates and fit all of the observations of the objective function made so far [Feurer and Hutter, 2019]. This way, the algorithm is capable of understanding if the loss has been improved or not with the last set of parameters, chooses the next most relevant set to explore and converges to an optimal solution.

Chapter 5 is the first case study proposed by this work. The chapter discusses the deep learning architecture chosen to perform the attack detection task on the datasets presented on Chapter 3, considering all of the information introduced on the present chapter, as well as the methodology applied on the experiments, the analysis of achieved results and comparison with other proposals.

Chapter 5

Botnet Attack Detection

In this chapter, a case study on the field of security for IoT environments is presented. This way, a deep learning architecture is proposed to solve the attack detection problem for IoT devices. The goal of this study is to demonstrate a methodology to choose a suitable machine learning method to tackle a known issue, as well as to analyse and evaluate the obtained results given the generated models. In this work, the attack detection task is considered as the step to purely identify whether the scanned data configures a botnet information or not. With this approach, only benign or attack data are classified, without any investigation about the types of the attacks.

The choice of the deep learning technique to perform the attack detection is explained in Section 5.1, followed by a description of the validation experiment in Section 5.2. The obtained results are discussed and compared with other works in Section 5.3, with a special analysis on the time required to train and test the models.

5.1 Defining the Machine Learning Architecture: Variational Autoencoder

With this study, it is proposed a methodology to detect and classify possible incoming attacks in an IoT environment. The detection step concerns only on identifying whether an incoming traffic data is an attack or not, thus, it is a binary classification problem. Because of that, the chosen ML architecture should be able to extract features from the both types of data and identify them correctly. Considering the recent efforts on the field, DL strategies are rising as promising architectures to solve many recent computing problems. They are known to be able to accomplish great reliability of metrics results (accuracy, precision, recall, and so on) while dealing with huge amounts of data, which is usually the case of IoT systems. A constraint that is normally related to deep learning architectures is the computational cost employed during the training phase, which should

be considered when choosing a technique.

For the attack detection step, the chosen strategy is the Variational Autoencoder (VAE), which is a neural network-based architecture. The highlighted characteristics of this type of architecture that weighed on the decision are [An and Cho, 2015; Kingma and Welling, 2019]:

- The family of Autoencoder (AE) algorithms is usually employed for reconstruction problems, since the main goal of this strategy is to deconstruct the input data into a latent space and reconstruct it as close as possible to the original. Because of this characteristic, Autoencoders are useful to identify anomalies on test data, given that the model was first trained using a set without anomalies.
- The Variational Autoencoder is a probabilistic subtype of the Autoencoder family that deconstructs data into a regularized latent space, which grants reliability to similarity measures because it ensures that similar datapoints lies closer.
- VAEs have been used on literature to deal with anomaly detection problems because they provide a more precise modelling and capture better representations of the input data than regular AEs. Some examples are Kim et al. [2020], Yang et al. [2019] and Lopez-Martin et al. [2017]
- The VAE training step can be faster than other DL architectures, like CNNs or RNNs, and may require less computational resources. This happens because it requires to be trained only with the benign data, which enables a reduced number of entities.

The VAE, as the regular AE, is composed by two main components: the encoder and the decoder. The encoder is responsible for learning a function ($p(z|x)$) that is capable of representing the input data into the latent space, that is a low dimensional reproduction of the data's features. However, differently from the traditional AE, the VAE learns to create two vectors for the latent space, one vector of means (μ) and one of standard deviations (σ). Those vectors represent a probability distribution $q(z|x)$ to the output z given an input x . The $q(z|x)$ term (Equation 5.1) can be understood as the multivariate Gaussian (N) of the latent vectors with a m dimension, generating the z points for each x entry. The term $\text{diag}(\sigma^2)$ is a vector represented by the diagonal of the variance-covariance matrix, where σ^2 is the variance values. Therefore, the matrix's diagonal contains the covariance values between all of the possible variable pairs.

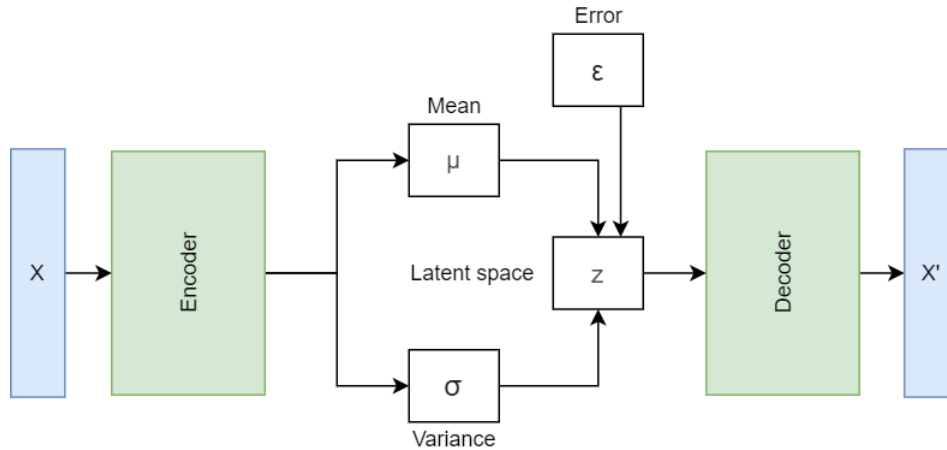
$$q(z|x) = N_m(z; \mu, \text{diag}(\sigma^2)), \quad (5.1)$$

The decoder, on the other hand, is responsible for reconstructing the input from the net of z points by using the learned function $p(x|z)$. When doing that, the decoder

assigns a reconstruction error to the output (x') that is backpropagated to the network, which indicates whether the reconstructed data are exactly or not. The reconstruction error is considered as a term of the loss function (Equation 5.2) to be minimized, as well as the Kullback-Leibler (KL) divergence that acts like a regularization term. The regularization term on ML models basically aims to simplify the final fitted model and prevent them from overfitting. The basic architecture of the VAE described in this section is shown on Figure 5.1, based on the works of [An and Cho \[2015\]](#) and [Kingma and Welling \[2019\]](#).

$$loss = \|x - x'\|^2 + KL[N(\mu_x, \sigma_x), N(0, 1)] \quad (5.2)$$

Figure 5.1: Basic architecture of a Variational Autoencoder.



Source: [Kingma and Welling \[2019\]](#).

The VAE has been proven as a good deep learning architecture to solve anomaly detection problems. [Lopez-Martin et al. \[2017\]](#) present one of the first works that apply a Conditional VAE on the context of attack detection by using the NSL-KDD dataset. [Yang et al. \[2019\]](#) apply a Conditional VAE both on the NSL-KDD and UNSW-B15 datasets. [Kim et al. \[2020\]](#), on the other hand, use a Recurrent VAE to classify the CTU-13 dataset in a real-time manner. [Song et al. \[2021\]](#) evaluate the use of AEs with different hyperparameters in different datasets, including the N-BaIoT. Moreover, [Alsoufi et al. \[2021\]](#) provide a study about the DL techniques applied for anomaly detection, pointing out that CNNs, LSTMs and AEs are the most employed architectures. Thus, these studies support the choice of using a Variational Autoencoder for attack detection on N-BaIoT and Bot-IoT databases.

5.2 Experiment Setup

This section demonstrates the steps taken to execute the algorithms and experiments for attack detection scenario. They were implemented with Python 3.6.9. All codes used in the experiments are available in a public repository¹. All experiments were performed on a computer with an Intel Core i9-9900X CPU at 3.50 GHz \times 20, 128 GB RAM, running a Linux Ubuntu 18.04.4 LTS 64-bit.

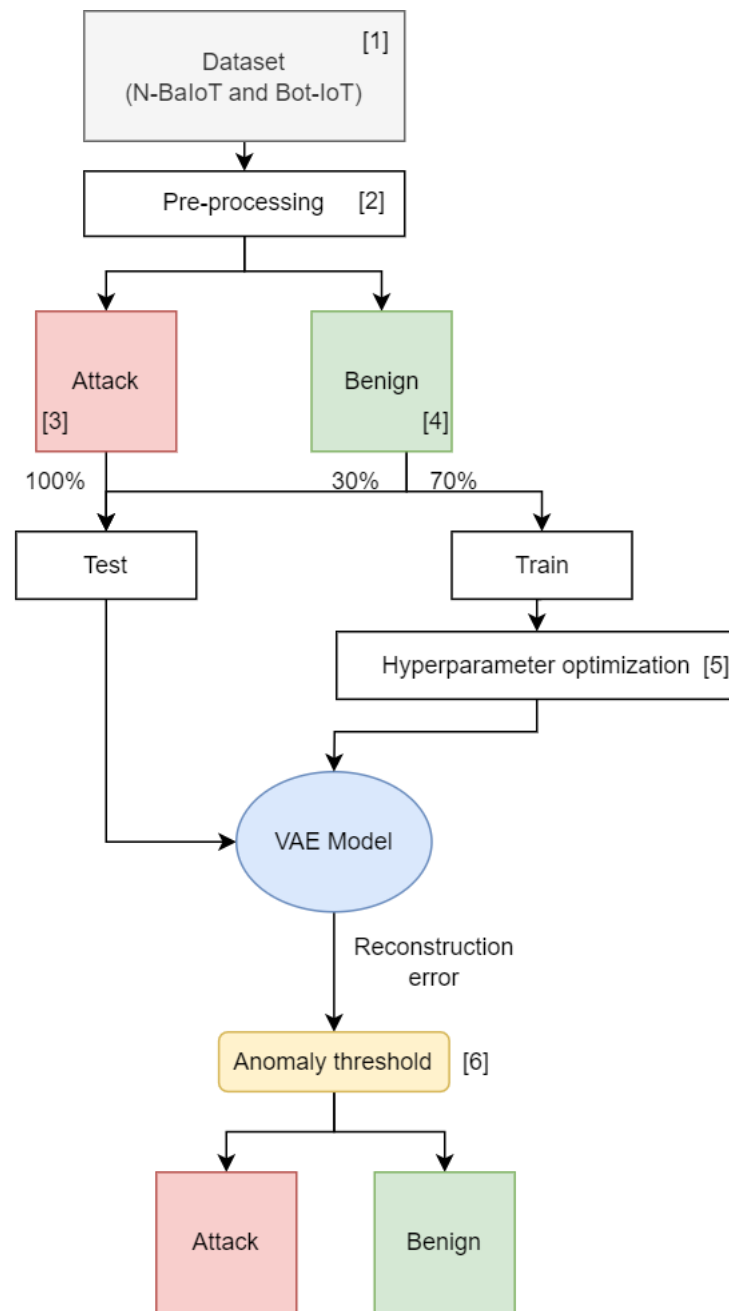
5.2.1 Input Data

The experiments are conducted with two datasets: N-BaIoT and Bot-IoT, which are explored in Chapter 3. The N-BaIoT base comprises nine different datasets, each one belonging to a different IoT device. A model is developed for each one of the devices, as well as an unique model that contains information from all of the devices, such as the methodology developed by [Borges et al. \[2022\]](#). Furthermore, the models are generated not only considering all of the 115 features of each dataset, but also considering a reduced feature space. This way, models are also created for 92, 69, 46 and 23 features. The reduction of the feature space is done concerning the obtaining time intervals L0.01, L0.1, L1, L3 and L5. For instance, the model with 115 features contains all of the columns; the model with 92 features contains all of the columns except from L0.01 columns; the model with 69 features contains L5, L3 and L1 columns and so on, until the model that contains only 23 features, with only L5 columns. With this approach it is intended to decrease the overall training time by reducing the feature space and select the features that are obtained in the shortest acquisition time, which could improve the detection time on a real case execution.

In the case of the Bot-IoT database, which has 43 features, only 36 features are selected. In addition to reducing the training interval, it is considered that the removed features (*stime*, *flgs*, *flgsnumber*, *saddr*, *sport*, *daddr*, and *dport*) do not add any relevant information to the detection task. Both N-BaIoT and Bot-IoT experiments are repeated five times each in order to understand if the results are maintained despite multiple iterations and obtain their standard deviation.

¹The repository with the code is available at <https://github.com/dekinks/DeepLearningSecurity>

Figure 5.2: Proposed methodology to execute the attack detection experiments on both datasets.



Source: Elaborated by the author.

5.2.2 Methodology

Figure 5.2 represents the proposed flow for the botnet attack detection experiments. First, data from any of the two datasets [1] is pre-processed with a minimum-maximum

normalization scaler [2]. This scaler is used to limit the interval of the values, guaranteeing a better model performance. Data contains both benign [3] and attack [4] information. Data is then splitted into two sets: a training set and a testing set. The training set is composed only by 70% of the total benign data, while test set is composed by all of the attack data plus the remaining 30% of the benign data.

The VAE model is trained only with the benign data. With this approach, it is intended to teach the model how to reproduce only the benign data, assigning greater reconstruction errors to any incoming anomalous data. The hyperparameters used to fit the model are generated via a bayesian optimizer algorithm [5] described in Section 4.3.2. The test set can be finally inserted into the trained VAE model [7], which should deconstruct the input and reconstruct it while assigning high errors to any unknown data. If the error is higher than an anomaly threshold [6], that is automatically calculated considering the error between the actual train set and the predicted train output, the element should be assigned as an attack; on the other hand, if the error is lower than the anomaly threshold, the data is considered benign.

5.3 Results and Discussion

The experiments are executed with nine different models for each IoT device and with the unique model for the N-BaIoT with features size varying from 115 columns to 23 columns. On the Bot-IoT case, four sensors are studied, which compose one unique model with 36 columns. The first results that can be analyzed regarding those datasets are the hyperparameters optimized via the bayesian method (learning rate, epochs and batch size) and estimated with the training set (anomaly threshold), whose values are grouped in the following Table 5.1. The hyperparameters are used to generate the proper VAE model for each device that can solve the proposed binary classification problem.

As discussed in Section 5.2.2, the attack detection experiments comprise two main scenarios: (i) results obtained with each device model (N-BaIoT) and (ii) results obtained with the unique models (N-BaIoT and Bot-IoT). On each of the N-BaIoT scenarios, the number of statistical features are decreased in order to evaluate the overall performance of the model with limited information. This approach can provide lower training and testing intervals and also consider data with reduced acquisition time, which is beneficial for edge computing scenarios.

Table 5.1: Hyperparameters optimized for each model.

Device	Learning Rate	Epochs	Batch Size	Anomaly Threshold
BoT-IoT (BT)	0.000501	550	19	0.043
Danmini Doorbell (DD)	0.000518	152	99	0.020
Ecobee Thermostat (ET)	0.000121	422	74	0.040
Ennio Doorbell (ED)	0.000553	51	64	0.040
Phillips Baby Monitor (PB)	0.000261	50	97	0.040
P737 Security Camera (P7)	0.000257	464	78	0.050
P838 Security Camera (P8)	0.000908	168	99	0.040
S1002 Security Camera (S2)	0.000468	259	71	0.055
S1003 Security Camera (S3)	0.000162	252	23	0.055
Samsung Webcam (SW)	0.000336	51	40	0.040
Unique Model (UM)	0.000001	44	28	0.050

5.3.1 F1-Score and Accuracy

The results of precision, recall and f1-score obtained for N-BaIoT dataset on each experiment are shown in Table 5.2. Table 5.3 provides the same results for the Bot-IoT data as well as its accuracy result.

The precision indicates the amount of True Positives predicted over all of the positives (true and false), or, in other words, whether the information assigned as attack is really an attack considering the positive set. The higher this value is, the lower the False Positive rate, which indicates that a low amount of benign data is classified as attack. The recall, on the other hand, is the overall rate of True Positive data considering the set of True Positives plus the False Negatives, identifying the percentage of correctly predicted attacks among the real attacks. A high recall indicates that the False Negative rate is low, that is, a low percentage of attacks is classified as benign data. This is the most important behavior to consider in the anomaly detection scenario, since it would be specially harmful to the devices if the model classifies a high amount of attacks as benign. Finally, the f1-score is the harmonic mean between those two metrics. With a first look on the Table 5.2, it is possible to conclude that the proposed architecture is suited to generate models that can achieve high f1-score values, guaranteeing a good reliability for their results.

With the N-BaIoT, the experiments are executed with a reduction of the feature space. This approach helps to understand whether the features acquired within high intervals are relevant to the detection task or not. The experiments conducted with 115,

Table 5.2: Precision, recall and f1-Score results of N-BaIoT dataset.

(a) 115 features				(d) 46 features			
Model	Precision %	Recall %	F1-Score %	Model	Precision %	Recall %	F1-Score %
DD	99.98 ± 0	100.00 ± 0	99.99 ± 0	DD	99.98 ± 0	100.00 ± 0	99.99 ± 0
ET	99.99 ± 0	100.00 ± 0	99.99 ± 0	ET	99.98 ± 0	99.99 ± 0	99.99 ± 0
ED	99.97 ± 0	100.00 ± 0	99.98 ± 0	ED	99.97 ± 0	100.00 ± 0	99.98 ± 0
PB	99.93 ± 0.01	100.00 ± 0	99.97 ± 0	PB	99.93 ± 0	100.00 ± 0	99.97 ± 0
P7	99.97 ± 0	100.00 ± 0	99.99 ± 0.01	P7	99.97 ± 0	99.99 ± 0	99.98 ± 0
P8	99.96 ± 0	100.00 ± 0	99.98 ± 0	P8	99.94 ± 0.03	100.00 ± 0	99.97 ± 0.02
S2	99.98 ± 0	100.00 ± 0	99.99 ± 0	S2	99.98 ± 0	100.00 ± 0	99.99 ± 0
S3	99.99 ± 0	99.99 ± 0	99.99 ± 0	S3	99.99 ± 0	99.99 ± 0	99.98 ± 0
SW	99.95 ± 0	99.91 ± 0.12	99.93 ± 0.06	SW	99.88 ± 0	99.99 ± 0	99.94 ± 0
Mean	99.97 ± 0.02	99.99 ± 0.01	99.98 ± 0.01	Mean	99.96 ± 0.06	99.99 ± 0.01	99.98 ± 0.03
UM	99.87 ± 0.01	99.99 ± 0	99.91 ± 0.05	UM	99.76 ± 0.02	99.78 ± 0.04	99.77 ± 0.03

(b) 92 features				(e) 23 features			
Model	Precision %	Recall %	F1-Score %	Model	Precision %	Recall %	F1-Score %
DD	99.98 ± 0	100.00 ± 0	99.99 ± 0	DD	99.98 ± 0	100.00 ± 0	99.99 ± 0
ET	99.99 ± 0	99.99 ± 0	99.99 ± 0	ET	99.96 ± 0	99.99 ± 0	99.98 ± 0
ED	99.97 ± 0	100.00 ± 0	99.98 ± 0	ED	99.96 ± 0.02	99.99 ± 0	99.98 ± 0.01
PB	99.93 ± 0	100.00 ± 0	99.97 ± 0	PB	99.92 ± 0.03	99.99 ± 0	99.96 ± 0.01
P7	99.97 ± 0	100.00 ± 0	99.99 ± 0	P7	99.95 ± 0.01	99.87 ± 0.18	99.91 ± 0.09
P8	99.96 ± 0	99.99 ± 0	99.98 ± 0	P8	99.85 ± 0.05	99.94 ± 0.13	99.87 ± 0.09
S2	99.98 ± 0	100.00 ± 0	99.99 ± 0	S2	99.98 ± 0	99.98 ± 0.02	99.98 ± 0.01
S3	99.99 ± 0	99.98 ± 0.03	99.98 ± 0	S3	99.98 ± 0	99.98 ± 0.01	99.98 ± 0.01
SW	99.89 ± 0.01	99.99 ± 0	99.94 ± 0.01	SW	99.81 ± 0.02	99.99 ± 0	99.90 ± 0.01
Mean	99.96 ± 0.02	99.99 ± 0.01	99.98 ± 0.03	Mean	99.93 ± 0.05	99.97 ± 0.05	99.95 ± 0.05
UM	99.84 ± 0.01	99.81 ± 0.04	99.83 ± 0.01	UM	99.52 ± 0.03	99.50 ± 0.16	99.49 ± 0.15

(c) 69 features			
Model	Precision %	Recall %	F1-Score %
DD	99.98 ± 0	100.00 ± 0	99.99 ± 0
ET	99.99 ± 0	99.99 ± 0	99.99 ± 0
ED	99.97 ± 0	100.00 ± 0	99.99 ± 0
PB	99.93 ± 0	100.00 ± 0	99.97 ± 0
P7	99.98 ± 0.01	99.99 ± 0.01	99.98 ± 0.01
P8	99.96 ± 0	99.98 ± 0.03	99.97 ± 0.02
S2	99.98 ± 0	100.00 ± 0	99.99 ± 0
S3	99.99 ± 0	99.99 ± 0	99.99 ± 0
SW	99.89 ± 0.01	99.99 ± 0	99.94 ± 0.01
Mean	99.96 ± 0.02	99.99 ± 0.01	99.98 ± 0.03
UM	99.83 ± 0.01	99.82 ± 0.04	99.83 ± 0.02

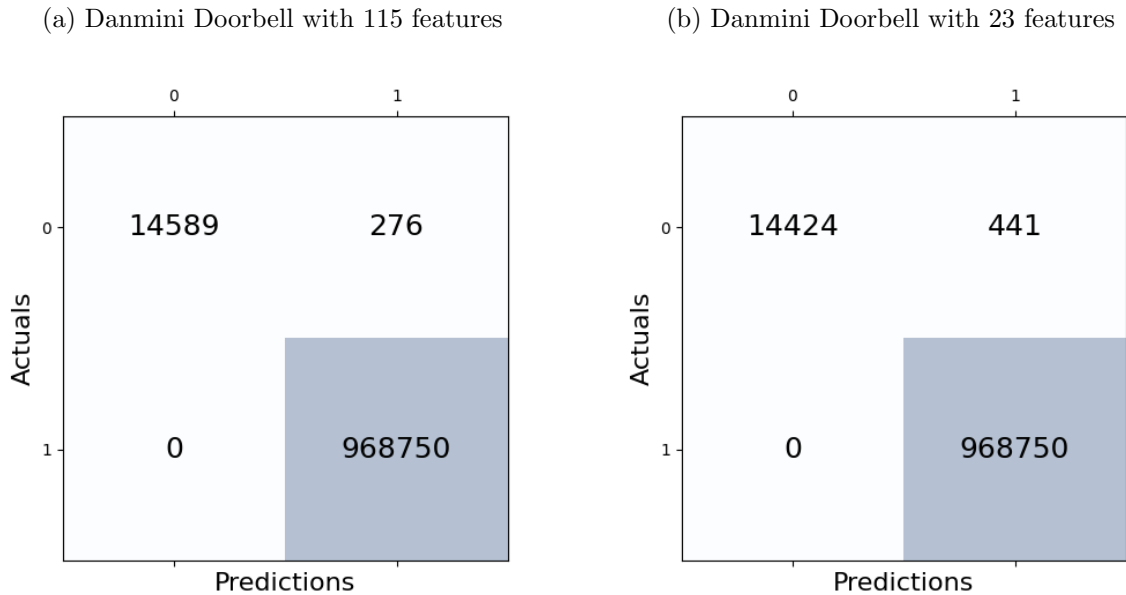
92 and 69 features show the highest f1-score results among models for the nine separate devices and for the unique model, which is the one that present the lowest results (values higher than 99.98 ± 0.03 and 99.83 ± 0.02 , respectively). The experiment with 46 features show a drop on the unique model f1-score results ($99.77 \pm 0.03\%$) while keeping an average f1-score of $99.98 \pm 0.03\%$ for the other models. The experiment with 23 features produces

the lowest overall f1-score results ($99.95 \pm 0.05\%$ for the separated models and $99.49 \pm 0.15\%$ for the unique model), which is an expected behavior, since the feature space is highly reduced. Despite of that, the values are still good, over 99.00% of confidence, which configures a tradeoff that can be considered when choosing a model. For the unique Bot-IoT model, the overall values are also considered good with an average of $99.98 \pm 0\%$ of f1-score and $99.96 \pm 0.01\%$ of accuracy.

Table 5.3: Precision, recall and f1-Score results of Bot-IoT dataset.

Model	Precision %	Recall %	F1-Score %	Accuracy %
BT	99.97 ± 0	99.99 ± 0	99.98 ± 0	99.96 ± 0.01

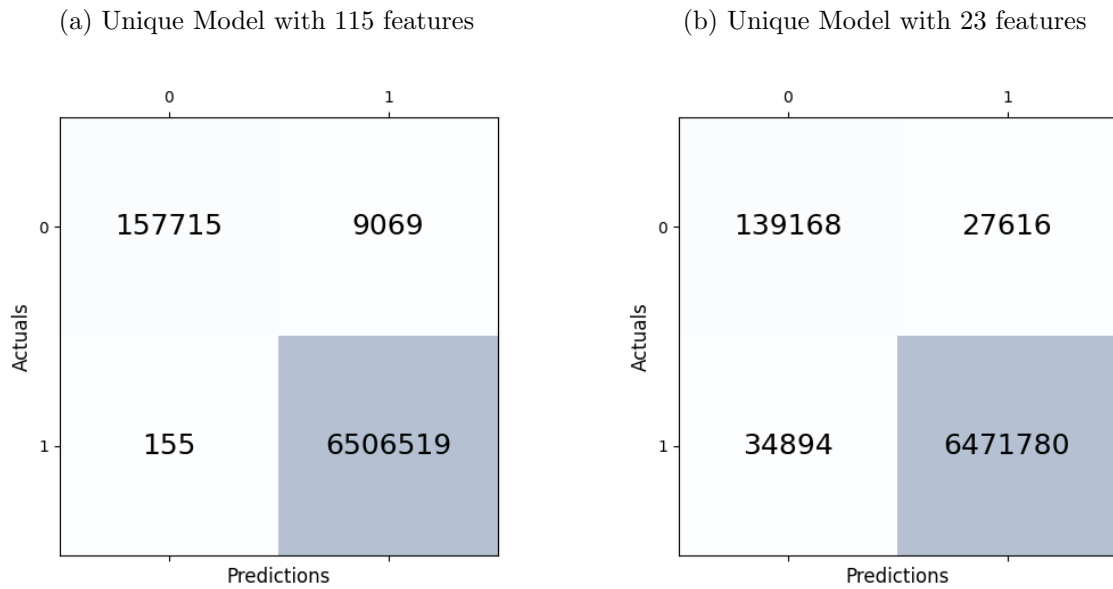
Figure 5.3: Confusion matrix of the attack detection on Danmini Doorbell model.



Source: Elaborated by the author.

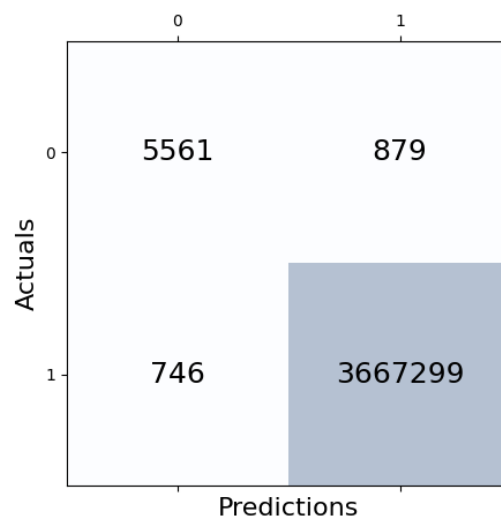
To better visualize the distribution of classified benign and attack data, Figures 5.3, 5.4 and 5.5 provide the confusion matrices of Danmini Doorbell device and the unique model for 115 and 23 features and of the Bot-IoT model, respectively. The confusion matrices of the other devices, which present a similar pattern as the Danmini Doorbell matrix, are provided in the Appendix A with Figures A.1, A.2, A.3, A.4, A.5, A.6, A.7 and A.8. The main diagonal values indicate the correctly predicted values: 0 labels are the True Negatives (TN), benign data; and 1 labels are the True Positives (TP), malware data. Values on the top right are the False Positives (FP), or benign data assigned as attack; and bottom left are the False Negatives (FN), or attack data assigned as benign.

Figure 5.4: Confusion matrix of the attack detection with N-BaIoT's unique model.



Source: Elaborated by the author.

Figure 5.5: Confusion matrix of the attack detection with Bot-IoT model.

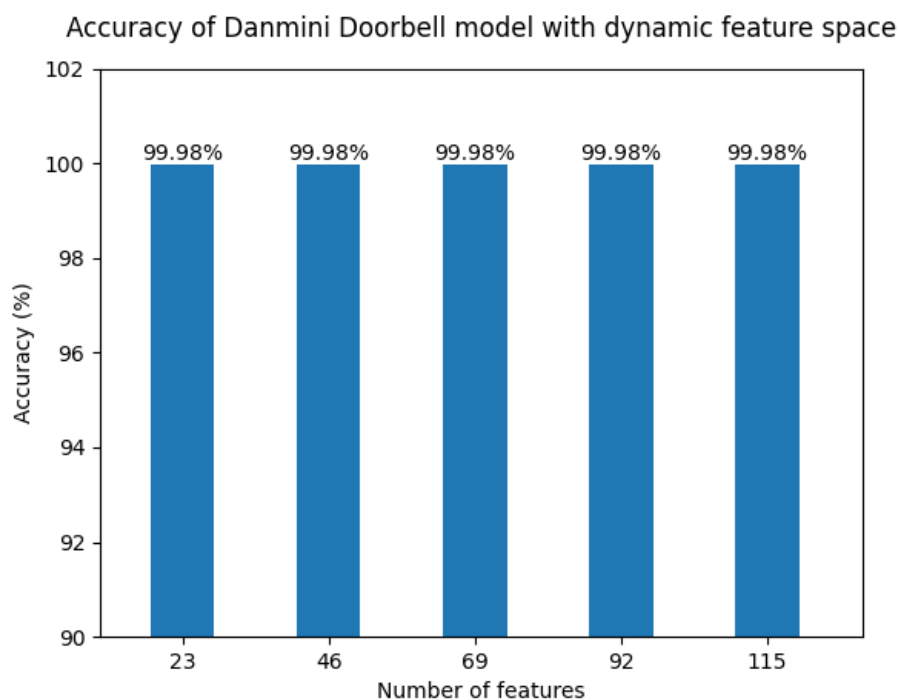


Source: Elaborated by the author.

It is possible to observe that for 115 features, models have high recall, with a false negative rate tending to 0. This is the pursued behavior, since it indicates that no malicious information is being wrongly assigned as benign. On the other hand, some of the benign data is being wrongly assigned as attack information, which indicates a lower precision for the overall benign information. However, the numbers are not expressive to the point of highly affecting the final f1-score values. It is also expected in the relationship of precision and recall metrics that one of the metrics is always higher than the other.

The unique model presents a higher rate of wrongly assigned data, however, the amount of FNs is still lower than the FPs. Using the unique model over the separated models is an interesting approach to consider because it would avoid training nine different models every iteration while being still able to achieve metrics higher than 99.00% even in the worst scenarios. The Bot-IoT unique model is also capable of achieving great results while keeping the FN rate lower than the FP one. There is a higher number of FNs in this case because the Bot-IoT dataset possesses a smaller amount of benign data to train the model. This problem was circumvented by using an upsampling technique to the Bot-IoT training data, but the reproduced benign data had the same known information, which can lead to punctual missclassifying.

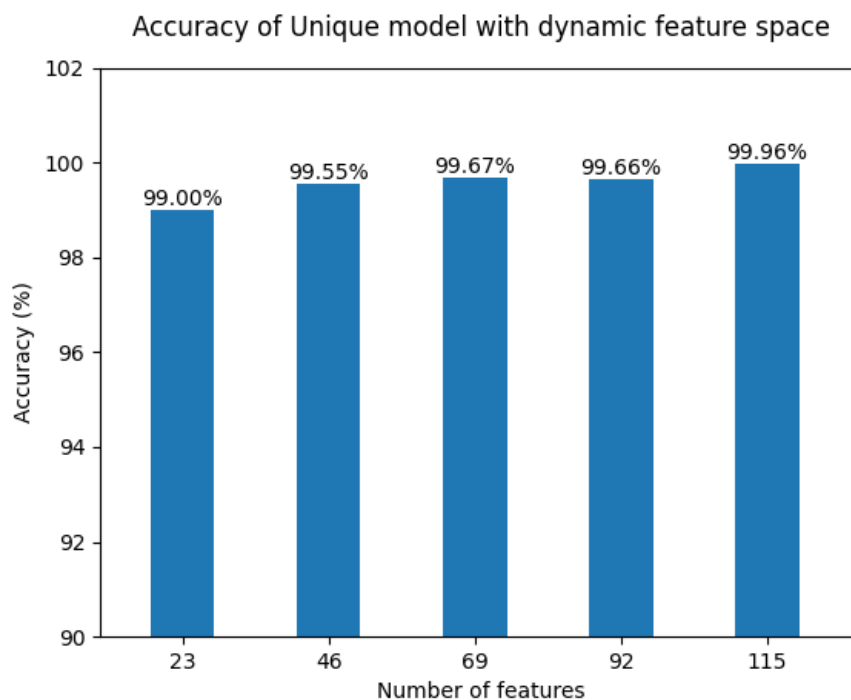
Figure 5.6: Accuracy results of Danmini Doorbell models on the attack detection task with dynamic feature space.



Source: Elaborated by the author.

The accuracy is the most used metric to evaluate ML models, and indicates the rate of hits the model achieves over the whole data. In this case, it is presented the

Figure 5.7: Accuracy results of unique models on the attack detection task with dynamic feature space.



Source: Elaborated by the author.

percentage accuracy for each model with the dynamic number of features. Figures 5.6 and 5.7 show that the reduction of the feature space based on the acquisition interval does not affect deeply the outcomes of the N-BaIoT set until the experiments with 23 features, which achieve the lowest results. The accuracy results of the other devices also follow the same pattern as the Danmini Doorbell model, which are provided in Appendix A with Figures A.9, A.10, A.11, A.12, A.13, A.14, A.15, A.16

In some models, like the Samsung Webcam (Figure A.16), Provision 737 and SimpleHome 1003 Security Cameras (Figures A.12 and A.15), the accuracy value increases with the reduction of features space. This phenomena occurs with deep learning architectures when a feature that is not relevant to the classification process is removed from the set, or when the removed feature was producing noise for the data. This study corroborates with the usage of a reduced number of columns to generate the models, since it can decrease training and testing intervals while keeping great results, as it will be discussed in Section 5.3.2. The unique model in particular is the one that present the lowest accuracy values, which is expected considering the variety of information that compose the set. However, the accuracy of the the worst scenario, with 23 features, still achieves a high accuracy percentage of 99.00%.

The Bot-IoT model presents an accuracy of $99.96 \pm 0.01\%$, which is close to its f1-score value showed on Table 5.3. It is important to evaluate different datasets with the

proposed architecture because it validates the reliability of the models and prove that the architecture can be successfully applied for different scenarios.

5.3.2 Training and Testing Intervals

Another metric that is very important to evaluate is the time spent on training and testing steps. Given the constraints presented for IoT and edge applications in this work, the computational power and time employed to run the algorithms are relevant when choosing a technique. This way, Tables 5.4 and 5.5 show, respectively, the time consumed by training and testing phases for N-BaIoT and Table 5.6 for Bot-IoT.

Table 5.4: Training interval of each evaluated model of N-BaIoT dataset.

Model	Size (rows)	Training Time(s)				
		115 c.	92 c.	69 c.	46 c.	23 c.
DD	34,683	141.42	135.26	146.88	87.90	81.02
ET	9,179	113.43	91.97	77.78	78.93	73.24
ED	27,370	46.63	46.64	44.87	35.29	32.84
PB	122,667	190.20	109.78	104.98	92.27	86.38
P7	43,507	697.85	504.00	484.88	437.48	333.14
P8	68,959	367.60	192.16	184.74	179.89	188.36
S2	32,609	207.59	193.84	179.18	158.07	174.35
S3	13,669	343.18	198.72	204.72	187.78	253.46
SW	36,505	99.74	65.44	59.03	55.56	102.77
UM	389,148	917.40	716.90	811.78	829.68	785.14

The training interval results show how long the architecture takes to create and fit the model that will be used to predict real data. The final values of each model are calculated by taking the average interval between all of the five experiments. With Table 5.4 it is possible to observe that the tendency of the training interval is to decrease with the reduction of the number of features. This behavior makes sense, since there are less columns to iterate over while fitting the model. Experiments with only 23 columns are faster to train in six out of ten results and experiments with 46 columns in the remaining four. The 23 column experiments does not hold all of the lowest results due to outlier results present on some of the five test sets, probably impacted by the number of executions performed by the machine concurrently. The same scenario happens with detection interval results in Table 5.5, whose values have a tendency to decay with lower number of features. 23 column experiments have four out of ten of the lowest results, 46 columns also have four of ten and 69 and 92 columns show the lowest testing intervals in

one out of ten experiments each. This tendency encourages the use of a smaller number of features to generate the models because it usually reduces the time consumed to perform the detection of the attacks while keeping high metrics as shown in Section 5.3.1.

Table 5.5: Detection, or testing interval, of each evaluated model. The ratio column is calculated by taking the mean value between the experiments and dividing it by the number of rows on the test set. This column shows the average interval that the model would take to detect an attack per incoming data.

Model	Size(rows)	Detection Time(s)					Ratio(ms)
		115 c.	92 c.	69 c.	46 c.	23 c.	
DD	983,615	39.37	37.69	34.69	24.60	21.46	0.03
ET	826,697	31.02	17.84	16.30	15.17	14.59	0.02
ED	328,130	10.41	10.41	11.43	6.55	6.17	0.03
PB	976,010	38.74	17.81	17.00	16.29	18.54	0.02
P7	785,753	34.07	21.12	18.23	19.61	14.52	0.03
P8	767,932	31.31	17.81	12.59	12.48	21.56	0.02
S2	830,447	26.54	16.72	13.99	13.12	18.25	0.02
S3	837,157	33.56	14.59	16.42	16.46	20.40	0.02
SW	338,717	12.51	6.53	7.95	8.77	9.91	0.03
UM	6,673,458	138.43	147.60	121.43	149.97	130.06	0.02

The Ratio column is calculated by taking the average detection time between the experiments with different number of columns and dividing it by the size of the data. This number provides an estimate of how long the model would take to detect an incoming threat per network packet. This average value for all of N-BaIoT models is 0.024 ms and 0.02 ms for the Bot-IoT. Considering the average latency of 300 ms required by an edge computing environment shown by [Chen et al. \[2017\]](#), the testing step time cost would fit the requirement and be able to process between 12,500 and 15,000 entities at once.

Table 5.6: Training and testing intervals of Bot-IoT dataset. The ratio column is calculated by taking the mean value between the experiments and dividing it by the number of rows on the test set. This column shows the average interval that the model would take to detect an attack per incoming data.

Model	Training Size(rows)	Testing Size(rows)	Training Time(s)	Detection Time(s)	Ratio (ms)
BT	15,025	3,674,485	669.67	90.40	0.02

5.3.3 Comparison with Related Works

As an important role to consider in the security field for IoT environments, detecting incoming attacks from the network has gained substantial space among recent researches, with prolific and relevant solutions [Laguduva et al., 2019; Alsoufi et al., 2021]. In this section, the goal is to evaluate the results and improvements achieved with the proposed VAE architecture in comparison to state-of-art works. Tables 5.7 and 5.8 contain a visual comparison between the studied metrics. The metrics of the current work showed on these tables are the ones collected as a mean value of all nine devices with the whole feature space (115 columns) for a full comparison.

- The attack detection performed by Meidan et al. [2018], creators of the N-BaIoT dataset, returns a mean precision of 99.30% with Deep Autoencoders (value calculated with provided TP and FP rates). This precision is outperformed by the proposed solution with a mean result of 99.97%. They also provide an average detection time analysis, with a mean value of 174 ms to detect attacks on their simulated setup. An analysis of recall, f1-score and accuracy values is not provided by the authors.
- Koroniotis et al. [2019] perform the binary-classification task with some ML and DL architectures. The most outstanding results are obtained by the RNN architecture with a mean value of 99.35% of accuracy and 89.64% and 78.89% of precision and recall. The proposed solution achieves greater results of 99.97% and 99.99% of precision and recall and 99.96% of accuracy.
- Nõmm and Bahşi [2018] evaluate only the accuracy and precision values with an unsupervised architecture. A differential of their work is that they evaluate the dynamic of the results with unbalanced and balanced datasets. The present experiment considers only the unbalanced scenario. In that case, the mean accuracy value of the tested devices is 96.64% and the mean precision is 98.71% against 99.96% and 99.97% for the same devices with the proposed architecture. Authors also do not provide any analysis about the time consumption during training and testing steps.
- Alqahtani et al. [2020] utilize an XGBoost algorithm with feature selection and an ensemble model to perform the detection task. When using the reduced feature space with 3 features only, their model is able to achieve 99.93%, 99.96%, 99.95% and 99.96% of precision, recall, f1-score and accuracy, being outperformed by little by the respective mean results of the proposed solution of 99.97%, 99.99%, 99.98%

Table 5.7: Comparison between the detection metrics achieved in this work and recent results from the literature for the N-BaIoT dataset.

Work	Precision	Recall	F1-score	Accuracy
Current Work	99.97%	99.99%	99.98%	99.96%
Meidan et al. [2018]	99.30%	-	-	-
Nõmm and Bahşi [2018]	98.71%	-	-	96.64%
Alqahtani et al. [2020]	99.93%	99.96%	99.95%	99.96%
Popoola et al. [2022]	99.97%	99.98%	99.98%	99.99%
Cunha et al. [2022a]	99.95%	98.10%	98.98%	99.95%

Table 5.8: Comparison between the detection metrics achieved in this work and recent results from the literature for the Bot-IoT dataset.

Work	Precision	Recall	F1-score	Accuracy
Current Work	99.97%	99.99%	99.98%	99.96%
Koroniotis et al. [2019]	89.64%	78.89%	-	99.35%
Popoola et al. [2022]	98.56%	100.00%	99.27%	100.00%

and 99.96% with the whole feature space. Authors also evaluate their training and testing steps intervals.

- **Popoola et al. [2022]** perform the binary classification of N-BaIoT and Bot-IoT data with complete feature space. N-BaIoT’s results are 99.99%, 99.97%, 99.98% and 99.98% of accuracy, precision, recall and f1-score respectively, while Bot-IoT achieves 100.00%, 98.56%, 100.00% and 99.27% for the same metrics. The proposed CNN is capable of achieving the same results of precision, recall and f1-score for the N-BaIoT, with a lower accuracy of 99.96%, while outperforming precision and f1-score metrics with results of 99.97% and 99.98% respectively. Authors also evaluate training and testing intervals, but with a different computational set from the current experiment.
- The VAE models provided by **Cunha et al. [2022a]** for the N-BaIoT dataset are improved on the current work by bringing the bayesian hyperparameter optimization and improving the built VAE architecture. As a result, the outcomes are better than the previous work, which produces an average value for all devices of 99.95%, 98.10%, 98.98% and 99.95% of precision, recall, f1-score and accuracy. The detection time is also improved with an average rate of 0.024 ms per incoming data against 0.04 ms from the previous experiment.

An overview across the presented works shows that the proposed solution is capable of achieving and outperforming some state-of-art results while keeping a low detection time and a reasonable training interval. Also, by restricting the feature space with a less number of columns it is possible to think about only keeping data with low acquisition time to possibly perform better on the edge computing context. This study also encourages diminishing the feature space for future experiments computational cost improvement, as seen in the next chapter.

Chapter 6

Botnet Attack Classification

The attack classification task is also considered in this work with the intention of determining which is the particular type of some incoming attack. This step is proposed because it can be useful to gain more knowledge about the botnets and possibly identify new types of attacks, according to the OpenSet paradigm. For this task, another architecture is proposed to solve the classification problem and validated according to the final results and comparisons. This chapter is organized similarly to the previous one, with Section 6.1 exposing the chosen deep learning method; Section 6.2 describing the methodology applied on the experiment; and Section 6.3 discussing the results and comparison with other works.

6.1 Defining the Machine Learning Architecture: Convolutional Neural Network

Unlike the proposal for attack detection, the attack classification task can assign data to more than two different labels. The proposed workflow considers as a first step to detect whether incoming information is malicious or not, and then identify which type of threat it is in the case of a positive result. The model should be able to extract the most outstanding features of the data and use them to predict the correct labels. For this task, the chosen architecture is the Convolutional Neural Network (CNN), a common technique used for classification problems. The relevant characteristics of this architecture are [Wibawa et al., 2022]:

- CNNs are usually used to classify 3D and 2D imagery information because of its characteristic to reduce the number of parameters across the filters without losing quality of the data. However, it has been proven as a suited model to deal with 1-dimensional data as well.
- The recent models that are built to treat images are usually complex and composed

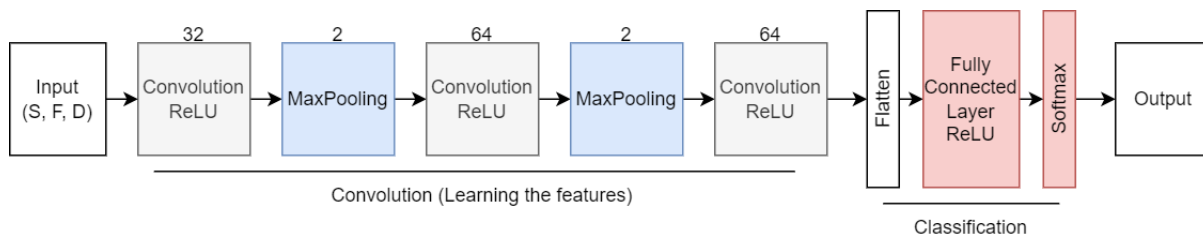
by many connected layers. However, to deal with data with less dimensions it is possible to pursue simpler and resourceless models [Zhanga et al., 2020].

- The CNN is one of the most employed deep learning architectures to perform anomaly classification as well as LSTMs and Autoencoders [Alsoufi et al., 2021].
- While training phases are usually time consuming, it is possible to achieve low testing intervals, which can be suitable for edge computing solutions.

A regular CNN is composed by the following layers: input, convolutional layer, pooling and fully-connected layers, which generates the desired output. Those architectures can also have some additional layers such as dropout, normalization and padding layers. The proposed flow is exposed in Figure 6.1. The Input size can be described as a combination of number of samples, number of features and depth size (S, F, D), and, in this case, the depth is 1 since it is a flatten data. The Input is inserted into a sequence of convolutional and max-pooling layers. The first layer is responsible for passing filters through the data, creating an activation map that is used to highlight its most outstanding characteristics by the latter. The values above the boxes are the number of filters applied on the convolutional layers and the size of the sliding window on the max-pooling layers.

The filters are applied with an activation function, which is used to introduce non-linearity in the process making it possible to learn more than just the linear relationships between the variables. The most used activation function on neural networks has been the ReLU (Rectified Linear Unit) because it is easy to optimize and it is computationally cheap, although it is a non-linear function. The function is basically used to decide whether a neuron on the network shall be activated or not, or, in other words, if the information is relevant or not.

Figure 6.1: Basic architecture of the proposed Convolutional Neural Network.



Source: Elaborated by the author.

After those steps, the data is flattened to a single dimension and inserted into the fully-connected layer. This layer is responsible to perform the classification of the extracted features by calculating class scores from the previous activations. The results of the last dense layer are passed through a Softmax activation function that is capable

of creating a distribution probability of the data that indicates the probability of the analysed data to belong to a class or not, providing the desired output. High percentages indicate that the information shall belong to the label, and low percentages indicate that it shall not. The proposed architecture is based on the work of [Zhang et al., 2020].

6.2 Experiment Setup

This section shows the steps followed to prepare and execute the experiments for the classification of the types of attacks. The experiments are conducted both using presented N-BaIoT and Bot-IoT datasets. The possible botnet attack labels that exist on both datasets are shown in Tables 6.1 and 6.2. The methodology explained below is repeated five times for each model in order to gather information about the standard deviation of the acknowledged metrics with the proposed solution.

Table 6.1: Types of botnet attacks on N-BaIoT dataset and their labels.

Botnet Attack Type	Label	Botnet Attack Type	Label
Ack	0	Combo	5
Scan	1	Junk	6
Mirai Syn	2	Bashlite Scan	7
Udp	3	Tcp	8
Udpplain	4	Udp	9

Table 6.2: Types of botnet attacks on Bot-IoT dataset and their labels.

Botnet Attack Type	Label
DDoS	0
DoS	1
Reconnaissance	2
Theft	3

6.2.1 Input Data

The input data for the classification experiments are similar to the one described in Section 5.2.1. In this case, however, the models are generated considering the 23 features related to the L5 time interval only. This approach is employed to train the model with the lowest possible acquisition time while keeping it simpler to reduce training and testing intervals. The Bot-IoT is studied with the 36 features that remained from the data extraction discussed on Section 5.2.1.

On the N-BaIoT, a different model is generated for each device. This approach is needed because each device has its own types of possible attacks, making it impossible to develop an unique model as in Chapter 5. Both devices Ennio Doorbell and Samsung Webcam only have information of attacks of the Bashlite type. The Bot-IoT, on the other hand, is constructed as a single model in accordance to the previous chapter.

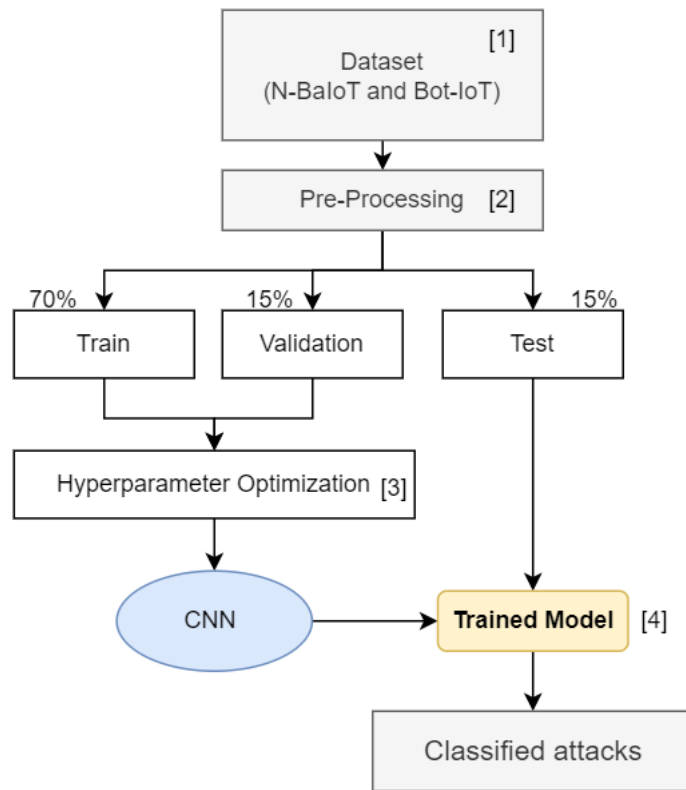
6.2.2 Methodology

The methodology applied for classification experiments is represented on Figure 6.2. Data from both datasets [1] is pre-processed with the minimum-maximum normalization scaler [2] and used to generate train, test and validation sets. 70% of the data is directed to the train set and 15% for validation and test sets each. The chosen CNN architecture is trained with train and validation sets to generate the final trained model [4], whose hyperparameters are chosen by the bayesian optimization discussed in Section 4.3.2. The time interval needed to train the model is collected on this step.

Finally, the test set can be inserted in the trained model, which should be able at this point to correctly classify each type of attack according to Tables 6.1 and 6.2. The classification results show to which label each data entry is assigned to. At this point, the results can be compared to the ground-truth information to evaluate the overall accuracy of the solution. The interval spent during the test phase is also evaluated because it indicates how much time would be necessary to classify the botnet attacks on a real case scenario.

To better analyze the accuracy of the proposed solution, the outcomes of the experiments are compared with two other machine learning strategies: the K-Nearest Neighbors (KNN) and the Naive Bayes (NB). The same methodology is applied in both architectures and the final results are evaluated.

Figure 6.2: Proposed methodology to execute the attack classification experiments on both datasets.



Source: Elaborated by the author.

6.3 Results and Discussion

The experiments are executed with the nine different IoT devices from N-BaIoT and with the unique dataset of Bot-IoT. The number of columns employed on experiments is fixed with 23 features for the first set and 36 for the second. With the bayesian algorithm it is possible to estimate the best set of hyperparameters to fit the proposed architecture, which are shown in Table 6.3.

Table 6.3: Hyperparameters optimized for each model.

Device	Learning Rate	Epochs	Batch Size
BoT-IoT (BT)	0.0009473	9	19
Danmini Doorbell (DD)	0.072929	89	50
Ecobee Thermostat (ET)	0.000746	18	27
Ennio Doorbell (ED)	0.003711	44	51
Phillips Baby Monitor (PB)	0.089869	14	33
P737 Security Camera (P7)	0.001000	85	12
P838 Security Camera (P8)	0.000615	43	17
S1002 Security Camera (S2)	0.000093	16	14
S1003 Security Camera (S3)	0.000397	35	12
Samsung Webcam (SW)	0.039677	35	49

Table 6.4: Precision, Recall and F1-Score results of the evaluated models.

(a) Precision values of each architecture				(c) F1-score values of each architecture			
Model	CNN	Precision % KNN	NB	Model	CNN	F1-Score % KNN	NB
DD	100.00 ± 0	99.96 ± 0.01	95.73 ± 0.47	DD	100.00 ± 0	99.96 ± 0.01	93.72 ± 1.00
ET	99.98 ± 0.01	99.97 ± 0	99.58 ± 0	ET	99.98 ± 0.01	99.97 ± 0	99.57 ± 0
ED	99.97 ± 0	99.92 ± 0	99.91 ± 0	ED	99.97 ± 0	99.92 ± 0	99.91 ± 0
PB	100.00 ± 0	99.95 ± 0.01	95.63 ± 0.20	PB	100.00 ± 0	99.95 ± 0.01	93.60 ± 0.49
P7	100.00 ± 0	99.75 ± 0.01	93.86 ± 0.44	P7	100.00 ± 0	99.76 ± 0.01	84.18 ± 9.07
P8	100.00 ± 0	99.76 ± 0.01	94.00 ± 1.55	P8	100.00 ± 0	99.75 ± 0.01	89.29 ± 2.75
S2	100.00 ± 0	99.96 ± 0	97.98 ± 0.87	S2	100.00 ± 0	99.96 ± 0	97.63 ± 1.20
S3	100.00 ± 0	99.95 ± 0.01	98.15 ± 0.53	S3	100.00 ± 0	99.95 ± 0.01	97.91 ± 0.75
SW	99.98 ± 0.01	99.92 ± 0.01	99.92 ± 0	SW	99.98 ± 0.01	99.92 ± 0.01	99.92 ± 0
Mean	99.99 ± 0	99.90 ± 0.01	97.20 ± 2.08	Mean	99.99 ± 0.01	99.90 ± 0.11	95.08 ± 5.84
Bot-IoT	100.00 ± 0	99.99 ± 0	97.77 ± 1.61	Bot-IoT	100.00 ± 0	99.99 ± 0	97.53 ± 0.04

(b) Recall values of each architecture			
Model	CNN	Recall % KNN	NB
DD	100.00 ± 0	99.96 ± 0.01	93.63 ± 0.91
ET	99.98 ± 0.01	99.97 ± 0	99.57 ± 0
ED	99.97 ± 0	99.92 ± 0	99.91 ± 0
PB	100.00 ± 0	99.95 ± 0.01	93.71 ± 0.52
P7	100.00 ± 0	99.75 ± 0.01	83.79 ± 8.33
P8	100.00 ± 0	99.76 ± 0.02	88.59 ± 3.22
S2	100.00 ± 0	99.96 ± 0	97.67 ± 1.16
S3	100.00 ± 0	99.95 ± 0.01	97.93 ± 0.72
SW	99.98 ± 0.01	99.92 ± 0.01	99.92 ± 0
Mean	99.99 ± 0	99.90 ± 0.11	94.99 ± 6.10
Bot-IoT	100.00 ± 0	99.99 ± 0	94.97 ± 0.07

6.3.1 F1-Score and Accuracy

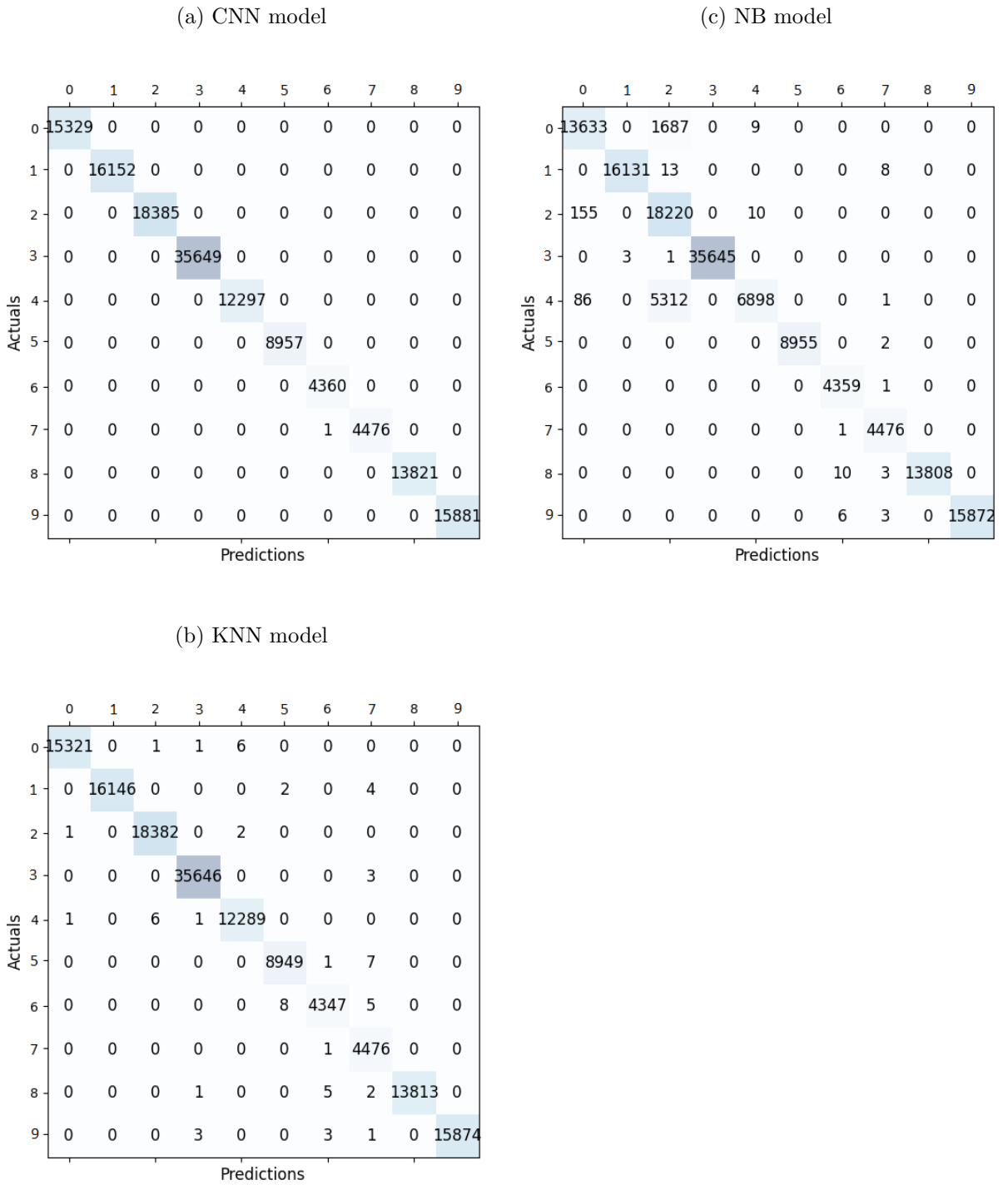
Results obtained for precision, recall and f1-score metrics are shown in Table 6.4. The table also indicates the metrics evaluated with the other two machine learning architectures for comparison, KNN and NB. All of N-BaIoT's metrics results are outperformed by the proposed CNN architecture, with a mean value of $99.99 \pm 0\%$ of precision, recall and f1-score. The KNN reaches $99.90 \pm 0.11\%$ on each metric, followed by the NB that achieves only $97.20 \pm 2.08\%$ of precision, $94.99 \pm 0.07\%$ of recall and $95.08 \pm 5.84\%$ of f1-score. For the Bot-IoT, the CNN achieves the highest results, reaching $100.00 \pm 0\%$ of precision, recall and f1-score against $99.99\% \pm 0\%$ with the KNN and $97.77 \pm 1.61\%$ of precision, $94.97 \pm 0.07\%$ of recall and $97.53 \pm 0.04\%$ of f1-score with the NB architecture. The CNN remains as the model that achieves the best final results, as expected of a deep learning strategy. It is also the most stable strategy, with a lower standard deviation of the mean results.

The confusion matrices generated by the experiments should also be considered on the evaluation. Those results are presented with Figures 6.3 and 6.4. Figures B.1, B.2, B.3, B.4, B.5, B.6, B.7 and B.8 represent the matrices of the other devices and are provided in Appendix B because they keep the same pattern results as the Danmini Doorbell model. Those matrices show the correlation between labels predicted by the DL strategy and ground-truth labels. This way, the main diagonal indicates the predictions that are correctly classified. It is possible to observe that the number of hits is according to the f1-score metrics gathered in Table 6.4, with a very low error rate. The matrices values also corroborate with the calculated results, showing that the CNN commit less classification errors than the KNN and NB strategies.

Finally, the accuracy reached with each studied architecture (CNN, KNN, NB) for each device is shown on Figures 6.5 and 6.6. The tendency is similar to the f1-score results, with CNN results reaching the highest values, followed by KNN results and the NB outcomes, with the lowest values. Figures B.9, B.10, B.11, B.12, B.13, B.14, B.15 and B.16 found in the Appendix B present the same behavior for the other N-BaIoT's devices.

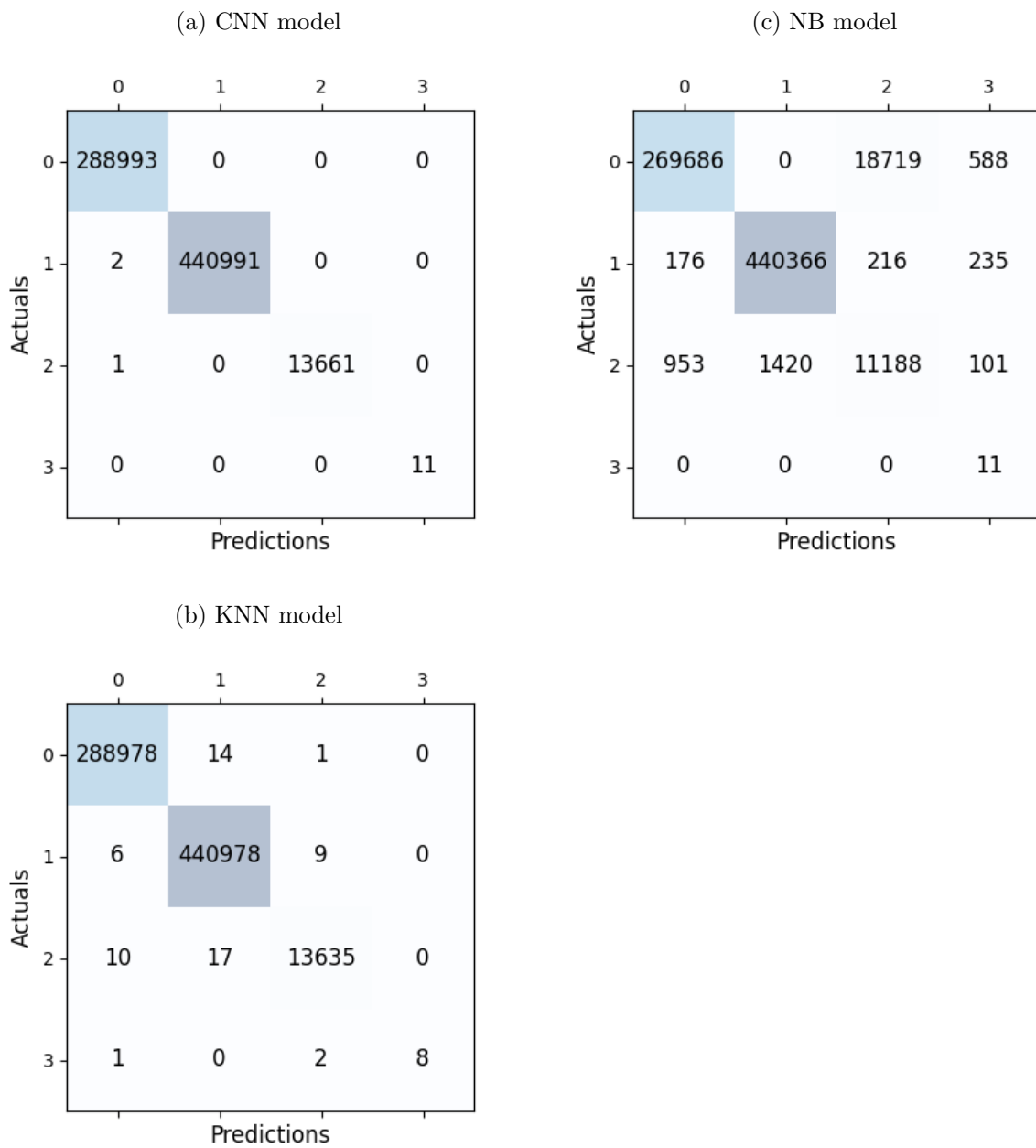
In particular, accuracy results presented on Figures B.12 and B.13 show a considerable drop of accuracy to around 99.75% with KNN and 86.00% with NB models. This phenomena might happen due to the existence of non-linear relationships among the device's data, which is better identified and classified by DL techniques than regular ML algorithms (in fact, NB is a linear solution). In that case, the proposed CNN keep 100.00% of accuracy.

Figure 6.3: Confusion matrix of the attack classification on Danmini Doorbell with CNN, KNN and NB models.



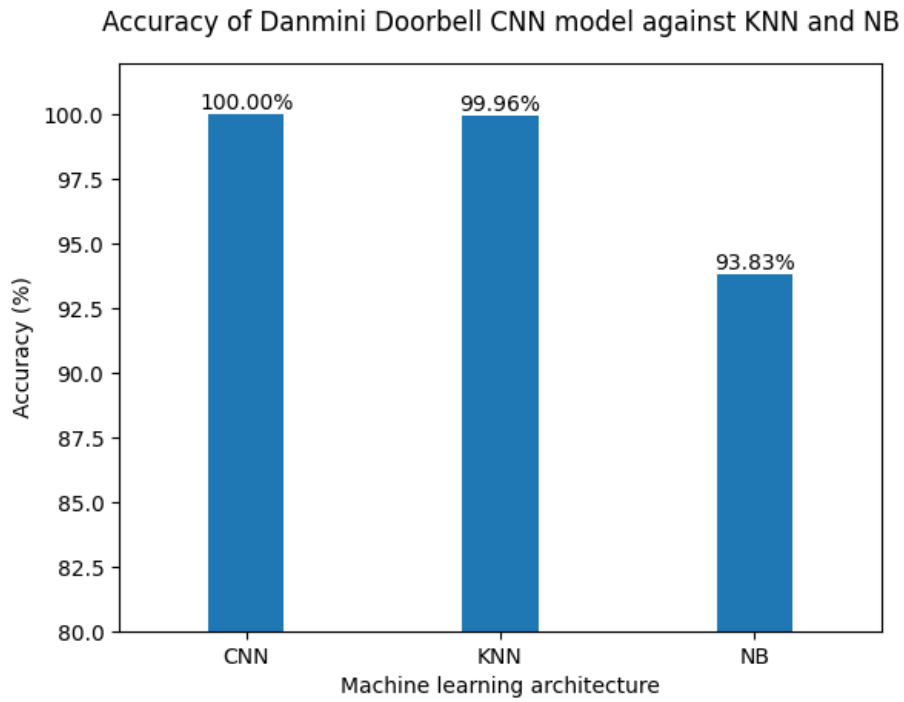
Source: Elaborated by the author.

Figure 6.4: Confusion matrix of the attack classification on Bot-IoT with CNN, KNN and NB models.



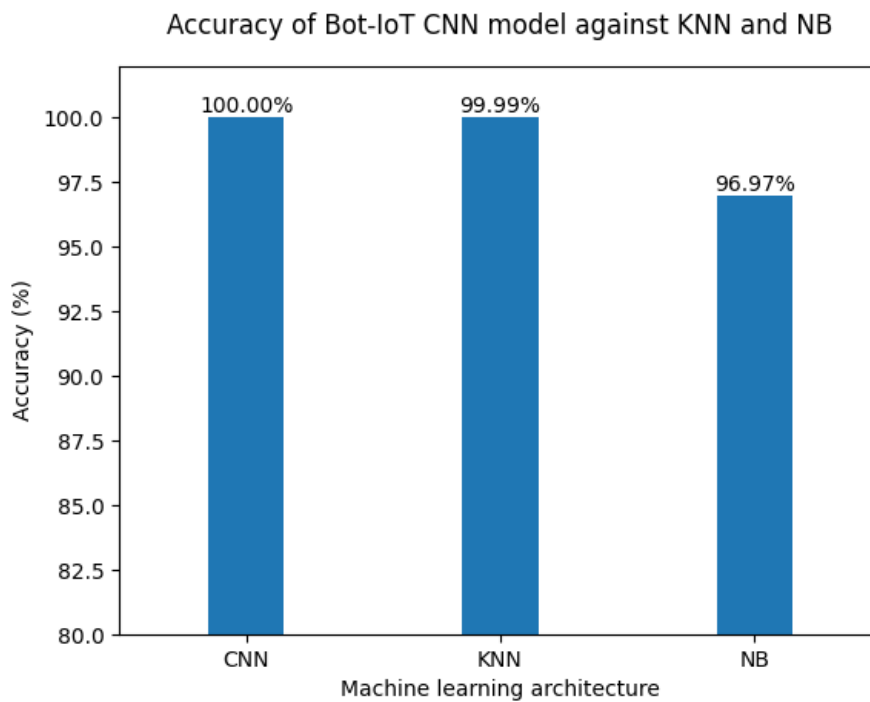
Source: Elaborated by the author.

Figure 6.5: Accuracy results for Danmini Doorbell on the attack classification task with CNN, KNN and NB models.



Source: Elaborated by the author.

Figure 6.6: Accuracy results for the Bot-IoT dataset on the attack classification task with CNN, KNN and NB models.



Source: Elaborated by the author.

6.3.2 Training and Testing Intervals

As in Chapter 5, the time intervals evaluation is very important to consider specially when thinking about edge devices. The lower the response latency is on those devices, the better for the user and for the environment management. Table 6.5 contains information of the time consumed to train the models for each proposed architecture. As expected, the CNN is the strategy that demands the higher training time since it is a more complex DL architecture with a high number of epochs and low learning rate (shown in Table 6.3) in most of the models. For instance, the model with the highest training interval of 22,710.44s is the Provision 737 Camera, which has a high number of epochs (85), a low learning rate (0.001) and a low batch size (12). Each of these hyperparameters contribute to the high training latency. ML strategies, on the other hand, are proven to be trained faster, where the KNN is the one that achieves the lowest training interval.

Table 6.5: Total training time of each model. The Size field corresponds to the number of rows. The ratio between total training time and the number of rows is not considered in this evaluation because the training step is always done with the whole data.

Model	Size(rows)	Training Time(s)		
		CNN	KNN	NB
DD	678,120	9522.05	0.31	1.08
ET	575,928	2684.97	0.22	0.90
ED	221,478	1527.84	0.07	0.38
PB	646,401	1607.70	0.44	1.25
P7	536,269	22710.44	0.25	0.80
P8	516,858	6173.30	0.29	0.95
S2	571,526	2801.31	0.35	1.07
S3	581,904	7453.16	0.25	0.88
SW	226,148	981.73	0.11	0.38
Bot-IoT	2,567,630	4707.59	0.49	1.51

Table 6.6, however, shows the most important interval to be considered on this analysis: the detection time. The CNN interval for this metric drops to the second fastest strategy, losing only for the NB models that are lighter and simpler. In this case, the KNN is the solution that causes the highest overhead to the detection interval, being almost 10 times slower than the proposed CNN. The ratio column, as in the previous chapter, indicates how long the algorithm would take to classify incoming network data. The proposed solution achieves a great average of 0.067 ms to process a single packet for N-BaIoT devices and 0.07 ms for Bot-IoT data. If considering the 300 ms interval required

for a real-time edge device brought by [Chen et al. \[2017\]](#), it would be possible to process between 4,300 and 4,500 packets at once. Considering the results discussed in Section 6.3, the CNN emerges as the architecture that enables the best results with reduced testing time.

Table 6.6: Classification time of each model. The Size field corresponds to the number of rows, and the Ratio columns indicate the proportion of the time interval spent in relation to the total number of rows.

Model	Size(rows)	Detection Time(s)			Ratio (ms)		
		CNN	KNN	NB	CNN	KNN	NB
DD	145,308	10.49	78.72	1.10	0.07	0.54	0.008
ET	123,410	9.55	240.96	0.81	0.08	1.95	0.007
ED	47,458	2.92	158.47	0.38	0.06	3.34	0.004
PB	138,512	9.02	79.52	0.97	0.06	0.57	0.007
P7	114,912	10.84	38.12	0.67	0.09	0.33	0.006
P8	110,752	5.64	55.44	0.80	0.05	0.50	0.007
S2	122,466	6.85	68.15	0.88	0.06	0.56	0.007
S3	124,690	9.50	54.30	0.78	0.07	0.43	0.006
SW	48,458	2.81	11.67	0.17	0.06	0.25	0.003
Bot-IoT	743,659	33.79	916.34	0.64	0.07	1.23	0.001

6.3.3 Comparison with Related Works

Since the efforts on attack detection and classification fields are a current trend, some works can be revisited to better understand possible gaps and improvements achieved by this research. A comparison between achieved results between the proposed solution and state-of-art works is exposed in Tables 6.7 and 6.8 for N-BaIoT and Bot-IoT, respectively. The N-BaIoT results are represented by the mean value of the metrics obtained with all models.

- The N-BaIoT dataset is created by [Meidan et al. \[2018\]](#). Although they have performed the botnet detection task to evaluate the dataset, a multi-classification step is not proposed by them. Current work exceeds the binary-classification proposition to also instigate a better understanding about the possible types of attacks dynamics.

- The Bot-IoT dataset is created by [Koroniotis et al. \[2019\]](#). Authors perform both detection and multi-classification procedures on this case. They achieve 99.99% of precision and accuracy and 100.00% of recall with a SVM architecture, but do not provide an evaluation of f1-score and the time consumption of the testing phase.
- [Htwe et al. \[2020\]](#) propose the ML model called CART and compare its outcomes with a NB model. Authors achieve 99.00% of accuracy against 99.99% with the proposed architecture for the N-BaIoT data. They also do not provide an evaluation of the training and testing intervals using their architecture.
- [Tran and Dang \[2022\]](#) achieve 90.00% of accuracy and recall with their ANN and 84.00% and 86.00% of precision and f1-score respectively, which are outperformed by the proposed CNN. They also provide an evaluation of training and testing intervals, but under a time complexity analysis.
- [Popoola et al. \[2022\]](#) perform the multi-classification analysis with a DNN and hyperparameter tuning both on N-BaIoT and Bot-IoT datasets. The N-BaIoT outcomes reveal an accuracy of 99.99%, and precision, recall and f1-score values of 99.91%, 99.98% and 99.89%, respectively. The proposed architecture outperforms f1-score metrics and achieves the accuracy value. For the Bot-IoT, their results are 99.96% of accuracy and 99.91%, 99.96% and 99.57% of precision, recall and f1-score, which are also outperformed by the 100.00% results of the proposed CNN. The authors also evaluate the training and testing phases of the DNN architecture with their computer set.
- [Parra et al. \[2020\]](#) develop a LSTM to execute the multi-classification task on N-BaIoT data with a per-attack approach. They achieve 94.90% of accuracy and 84.30%, 99.99% and 96.86% of precision, recall and f1-score. Their results are outperformed by the proposed CNN. They do not provide any evaluation regarding the time consumption of training and testing phases.
- The model proposed by [Cunha et al. \[2022b\]](#) is improved on this work with a new set of tuned hyperparameters and improvement of the developed CNN architecture. With that, it is possible to note an advance of the metrics values. The previous work has precision, recall, f1-score and accuracy results of 97.82%, 97.75%, 97.59% and 97.75% respectively with the N-BaIoT set. They are outperformed by the newest architecture with a 99.99% result for all metrics. The values for the Bot-IoT set increase to 100.00%. The training interval is slightly better for some cases and worse for others due to the change of the hyperparameter set (a higher number of epochs, lower learning rate and lower batch size can cause the model to take a longer time to train, as well as the state of the remote machine and its parallel operations at the time), but the testing step keep an average value of 0.07 ms per-packet.

Table 6.7: Comparison between the classification metrics achieved in this work and recent results from the literature for the N-BaIoT dataset.

Work	Precision	Recall	F1-score	Accuracy
Current Work	99.99%	99.99%	99.99%	99.99%
Htwe et al. [2020]	-	-	-	99.00%
Tran and Dang [2022]	84.00%	90.00%	86.00%	90.00%
Popoola et al. [2022]	99.91%	99.88%	99.89%	99.99%
Parra et al. [2020]	84.30%	99.99%	96.86%	94.80%
Cunha et al. [2022b]	97.82%	97.75%	97.59%	97.75%

Table 6.8: Comparison between the classification metrics achieved in this work and recent results from the literature for the Bot-IoT dataset.

Work	Precision	Recall	F1-score	Accuracy
Current Work	100.00%	100.00%	100.00%	100.00%
Koroniotis et al. [2019]	99.99%	100.00%	-	99.99%
Popoola et al. [2022]	99.91%	99.96%	99.57%	99.96%
Cunha et al. [2022b]	99.99%	99.99%	99.99%	99.99%

Considering the presented works, it is possible to infer that the proposed methodology is suitable and capable of outperforming many state-of-art results for the attack classification task. In fact, the proposed model achieves the best set of results, considering all of the studied metrics, for both datasets experiments. The evaluation of training and testing phases is also a recent concern for IoT environments, specially when considering the application of complex DL models to edge devices. This evaluation is not done on some of the studied papers, which is a relevant gap acknowledged by [Aversano et al. \[2021\]](#) survey.

Chapter 7

Conclusion

The presented work explores an important and recent field among computing technologies, which is the Internet of Things. IoT systems are proven to be suitable and reliable solutions to many recent necessities in the globalized society, such as smart healthcare, by providing ways to early detect diseases, monitor patients and ensure better quality of life to elderly and people with disabilities [Bolhasani et al., 2021]; smart cities, enabling smart homes and transportation, efficient distribution of energy and object and people detection [Atitallah et al., 2020]; smart agriculture and environment, helping on predicting weather tendencies, identifying deforestation and detecting suitable land relief, to cite some examples [Aversano et al., 2021; Wang et al., 2020].

As a part of the current society, it is important to study, research and analyse efforts made on the IoT field to predict tendencies and explore and improve solutions to the many possible challenges. Across its chapters, this work focuses on providing the main characteristics and tendencies of IoT environments, considering the possible architectures to build systems that provide Quality of Service and the most relevant challenges and constraints pointed out by the researched papers. In this context, the edge computing paradigm arises as a solution to enable secure, low-power, energy-efficient and real-time environments, requirements that cloud-only systems are not able to achieve anymore. This way, the work also provides a deep analysis about those requirements and possible strategies to tackle them when building IoT systems, with a special emphasis to security for IoT environments and its challenges.

Another trend in the IoT context is to use artificial intelligence algorithms to elevate the capabilities of the systems and provide reliable and automated solutions. Not only those types of algorithms help on decision making tasks, since they are able to learn trends and feature relationships from environment data, but also are proven to be helpful to overcome constraints and achieve requirements belonging to the IoT world. The researched topics show that machine and deep learning solutions are being successfully used to organize data governance, provide reliable communication within the environment, optimize gateway usage, enable real-time operations, such as object detection, guarantee security and privacy of the devices and so on. In this context, the work presents recent efforts that combine ML and IoT technologies with an analysis to help on defining the

architectures to be used when building those types of systems.

The experiment chapters of the dissertation show a case study precisely in the security field, specifically in botnet attack detection and classification for IoT environments. Two deep learning strategies are proposed to (i) early identify incoming attack information and (ii) classify the possible types of incoming attacks using datasets that comprise real network information of IoT devices when functioning with normal data and when receiving abnormal information. The architecture proposal is to first identify the threats and then classify its types to gain knowledge about the taxonomy of botnet attacks to help on defining new and better protection methods. The final evaluations show that the generated models are able to achieve and overcome known state-of-art results, with a special analysis of the time needed to detect and classify attacks coming from the network, since low intervals enable edge computing implementation.

Analyzing purely the testing step, which require less computational power and represents a real world situation, the results imply that the developed models could be applied within edge devices for a real-time attack detection and classification. However, improvements to the algorithms can be done, as well as adding more steps to create a full automated system capable of dealing with attack information. The training step is a particularly concerning phase, because it requires a longer time and considerable computational power to execute. Also, since IoT environments are commonly dynamic, it should be considered how to train the models when new devices are inserted on the network or excluded from it. The continuation ideas for this work are explored in the following section.

The research and the results obtained during this dissertation culminated in two published papers, referred in [Cunha et al. \[2022a\]](#) and [Cunha et al. \[2022b\]](#).

7.1 Future Work

Considering the DL architectures proposed in this work and the generated models to deal with botnet attacks in IoT devices, some ideas could be used to improve their results and generate more knowledge to the user. Despite the low time intervals achieved in the testing step, the training phase of DL solutions is costly considering the required computational power and the time needed to fit the model. This scenario is bad for edge computing solutions, which require fast executions to enable real-time systems and usually have limited computational resources.

Thinking about the issues related to the training of the model, a technique that has been implemented with great results is to offload the DL model from the edge node

to the cloud to perform the training step in a resourceful environment and load it back to the device once it is fitted [Gao et al., 2021; Merenda et al., 2020; Gopalakrishnan et al., 2020; Tian et al., 2020]. This loop could be done from time to time to keep the model up to date and converge to the best metric results. Finally, the last loaded model would be able to perform detection and classification tasks in the edge devices while keeping low-latency responses.

Other approaches that are being used in the deep learning context are to develop tiny DL models that consumes less power and perform better, pre-train models to enable algorithms with a lower cost and apply distributed learning techniques to parallelize the learning processes [Lin et al., 2020; Tuli et al., 2020; Sun et al., 2019]. Also, to keep the privacy of the data while training shared models and enable an edge device-learning approach, a recent solution is to apply the Federated Learning, as seen in Gao et al. [2021], which is also an interesting option. As a continuation suggestion, the proposed techniques could be implemented in a real world installation with an edge architecture to validate the theoretical hypothesis raised by this work and evaluate the new propositions.

Another knowledge that is interesting to add in the attack detection context is to identify whether an attack type is known or not. The detection phase only discovers if the data is benign or malicious and the classification identifies the types of malicious data that are used to harm the devices. However, if an attack type is out of the known set, the model would not be able to correctly classify it. With this motivation, the OpenSet paradigm emerges as a solution to identify new types of incoming attacks and expand the knowledge about botnet possible threats [Bendale and Boulton, 2016]. This way, another step that is proposed to be added in a further work is to develop a DL architecture capable of identifying open world data and assign unknown classes.

From the initial proposals of this dissertation and the accomplished work among research and development steps, it can be stated that the pursued contributions were achieved and the work provides strong results. Not only an extensive research with a deep analysis about IoT environments requirements is summarized to help on developing applications with ML strategies considering the novel edge computing paradigm, but also the implemented architectures reach great results among the state-of-art and it is possible to expand the efforts to develop a solid architecture that could be applied in many IoT contexts to provide real-time, reliable and secure solutions.

Bibliography

- Mohammed J. Zaki and Wagner Meira. *Data Mining and Machine Learning - Fundamental Concepts and Algorithms*. Cambridge University Press, 2020.
- Aurlien Gron. *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2019.
- Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiot: network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(9):12–22, 2018. doi: 10.1109/MPRV.2018.03367731.
- Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv:1802.09089*, 2018. URL <https://doi.org/10.48550/arXiv.1802.09089>.
- Mnahi Alqahtani, Hassan Mathkour, and Mohamed Maher Ben Ismail. Iot botnet attack detection based on optimized extreme gradient boosting and feature selection. *Sensors (Switzerland)*, 20:1–21, 2020.
- Samizadeh Nikoui, Rahmani A., Balador A., and Haj Seyyed Javadi H. Internet of things architecture challenges: A systematic review. *International Journal of Communication Systems*, 34(4), 2021.
- Awishkar Ghimire, Surendrabikram Thapa, Avinash Kumar Jha, Amit Kumar, Arunish Kumar, and Surabhi Adhikari. Ai and iot solutions for tackling covid-19 pandemic. *Research Gate*, 2020.
- Lerina Aversano, Mario Luca Bernardi, Marta Cimitile, and Riccardo Pecori. A systematic review on deep learning approaches for iot. *Computer Science Review*, 40, 2021.
- Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain. Machine learning in iot security: Current solutions and future challenges. *IEEE Communications Surveys and Tutorials*, 22:1686–1721, 2020. doi: 10.1109/COMST.2020.2986444.
- Vishalini R. Laguduva, Sheikh Ariful Islam, Sathyanarayanan Aakur, Srinivas Katkoori, and Robert Karam. Machine learning based iot edge node security attack and counter-measures. *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, pages 670–675, 2019.

- Nazrul Hoque, Dhruva K. Bhattacharyya, and Jugal K. Kalita. Botnet in ddos attacks: Trends and challenges. *IEEE Communications Surveys and Tutorials*, 17, 2015. URL [10.1109/COMST.2015.2457491](https://doi.org/10.1109/COMST.2015.2457491).
- Constantinos Koliass, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50:80–84, 2017. doi: 10.1109/MC.2017.201.
- Nickolaos Koroniotis, Nour Moustafaa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796, 2019.
- Mohammad S. Mahdavinejad, Mohammadreza Rezvan, Mohammadamin Barekatin, Peyman Adibi, Payam Barnaghi, and Amit P. Sheth. Machine learning for internet of things data analysis: a survey. *Digital Communications and Network*, 4:161–175, 2018. URL <https://doi.org/10.1016/j.dcan.2017.10.002>.
- He Li, Kaoru Ota, and Mianxiong Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, 32(1):96–101, 2018.
- Wen Sun, Jiajia Liu, and Yanlin Yue. Ai-enhanced offloading in edge computing: When machine learning meets industrial iot. *IEEE Network*, 33(5):68–74, 2019.
- Maurizio Capra, Riccardo Peloso, Guido Masera Massimo Ruo Roch, and Maurizio Martina. Edge computing: A survey on the hardware requirements in the internet of things world. *Future Internet*, 11, 2019.
- Minghua Wang, Lihua Zhu, Laurence T. Yang, Man Lin, Xianjun Deng, and Lingzhi Yi. Offloading-assisted energy-balanced iot edge node relocation for confident information coverage. *IEEE Internet of Things Journal*, 6(3):4482–4490, 2019.
- Meng Ma, Ping Wang, and Chao Hsien-Chu. Data management for internet of things: Challenges, approaches and opportunities. *Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCoM*, 13:1144–1151, 2013.
- Mukhtar Yahuza, Mohd Yamani Idna Bin Idris, Ainuddin Wahid Bin Abdul Wahab, Anthony T. S. Ho, Suleman Khan, Siti Nurmayana Binti Musa, and Azni Zarina Bini Taha. Systematic review on security and privacy requirements in edge computing: State of the art and future research opportunities. *IEEE Access*, 8:76541–76567, 2020.
- Yinhao Xiao, Yizhen Jia, Chunchi Liu, Xiuzhen Cheng, Jiguo Yu, and Weifeng Lv. Edge computing security: State of the art and challenges. In *Proceedings of the IEEE*, volume

107. IEEE, 2019. doi: 10.1109/JPROC.2019.2918437. URL <https://ieeexplore.ieee.org/document/8741060>.
- Massimo Merenda, Carlo Porcaro, and Demetrio Iero. Edge machine learning for ai-enabled iot devices: A review. *Sensors (Switzerland)*, 20(9), 2020.
- Farzad Samie, Lars Bauer, and Jörg Henkel. From cloud down to things: An overview of machine learning in internet of things. *IEEE Internet of Things Journal*, 6(3):4921–4934, 2019. URL <https://doi.org/10.1016/j.dcan.2017.10.002>.
- Lorenzo Corneo, Nitinder Mohan, Aleksandr Zavodovski, Walter Wong, Christian Rohner, Per Gunningberg, and Jussi Kangasharju. (how much) can edge computing change network latency? *2021 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2021.
- Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. *IEEE Conference*, 2018a.
- Wei Yu, Fan Liang, Xiaofei He, William G. Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the internet of things. *IEEE Access*, 6:6900–6919, 2017.
- Xiaofei Wang, Yiwen Han, Victor C. M. Leung, Dusit Niyato, Xueqiang Yan, and Xu Chen. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys and Tutorials*, 22:869–904, 2020.
- Shreshth Tuli, Nipam Basumatary, Sukhpal Singh Gill, Mohsen Kahani, Rajesh Chand Arya, Gurpreet Singh Wander, and Rajkumar Buyya. Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments. *Future Generation Computer Systems*, 104:187–200, 2020. URL <https://doi.org/10.1016/j.future.2019.10.043>.
- Douglas Adams. *The Salmon of Doubt*. Arqueiro, 2002.
- Lulwah AlSuwaidan. Data management model for internet of everything. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 331–341, 2019.
- J.G. Sikwela. Data management considerations in the design of internet of things applications. *Thesis (Doctorate Degree) - University of Johannesburg*, 2012.
- P.P. Ray. A survey on internet of things architectures. *Journal of King Saud University - Computer and Information Sciences*, 30(3):291–319, 2018.

- Pathum Chamikara Mahawaga Arachchige, Peter Bertok, Ibrahim Khalil, Dongxi Liu, Seyit Camtepe, and Mohammed Atiquzzaman. A trustworthy privacy preserving framework for machine learning in industrial iot systems. *IEEE Transactions on Industrial Informatics*, 16(9):6092—6102, 2020.
- Laizhong Cui, Shu Yang, Fei Chen, Zhong Ming, Nan Lu, and Jing Qin. A survey on application of machine learning for internet of things. *International Journal of Machine Learning and Cybernetics*, 9:1399–1417, 2018.
- Yansong Gao, Minki Kim, Chandra Thapa, Alsharif Abuadbbba, Zhi Zhang, Seyit Camtepe, Hyounghick Kim, and Surya Nepal. Evaluation and optimization of distributed machine learning techniques for internet of things. *ArXiv*, 2021.
- Wendel Serra, Warley Junior, Isaac Barros, Hugo Kuribayashi, and João Carmona. Monitoring and smart decision architecture for drone-fog integrated environment. *Brazilian Symposiums of Ubiquitous and Pervasive Computing (SBCUP 2021)*, pages 102–111, 2021.
- Zaffar Haider Janjua, Massimo Vecchio, Mattia Antonini, and Fabio Antonelli. Irese: An intelligent rare-event detection system using unsupervised learning on the iot edge. *Engineering Applications of Artificial Intelligence*, 84:41–50, 2019.
- Kostas Kolomvatsos. Proactive tasks management for pervasive computing applications. *Journal of Network and Computer Applications*, 176, 2021.
- Pranav Sharma, Marcus Rub, Daniel Gaida, Heiko Lutz, and Axel Sikora. Deep learning in resource and data constrained edge computing systems. *Machine Learning for Cyber Physical Systems*, pages 43–51, 2021.
- Shicai Ji, Ying Peng, Hongjia Zhang, and Shengbo Wu. An online semisupervised learning model for pedestrians’ crossing intention recognition of connected autonomous vehicle based on mobile edge computing. *Wireless Communications and Mobile Computing*, 2021, 2021.
- Shailendra Rathore and Jong Hyuk Park. Semi-supervised learning based distributed attack detection framework for iot. *Applied Soft Computing Journal*, 72:78–79, 2018.
- Jienan Chen, Siyu Chen, Qi Wang, Bin Cao, Gang Feng, and Jianhao Hu. Iraf: A deep reinforcement learning approach for collaborative mobile edge computing iot networks. *IEEE Internet of Things Journal*, 6(4):7011–7024, 2019.
- Fernando Castano, Stanislaw Strzelczak, Alberto Villalonga, Rodolfo E. Haber, and Joanna Kossakowska. Sensor reliability in cyber-physical systems using internet-of-things data: A review and case study. *Remote Sensing*, 11, 2020.

- Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. Iotpot: A novel honeypot for revealing current iot threats. *Journal of Information Processing*, 24(3):522–533, 2016. doi: <https://doi.org/10.2197/ipsjjip.24.522>.
- Huang Zeyu, Xia Geming, Wang Zhaohang, and Yuan Sen. Survey on edge computing security. In *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, Fuzhou, China, 2020. IEEE. doi: 10.1109/ICBAIE49996.2020.00027.
- Sven Nõmm and Hayretdin Bahşi. Unsupervised anomaly based botnet detection in iot networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1048–1053, Orlando, FL, 2018. IEEE. doi: 10.1109/ICMLA.2018.00171.
- Joao Batista Borges, Joao P. S. Medeiros, Luiz P. A. Barbosa, Heitor S Ramos, and Antonio A Loureiro. Iot botnet detection based on anomalies of multiscale time series dynamics. *IEEE Transactions on Knowledge and Data Engineering*, 2022.
- Segun Popoola, Bamidele Adebisi, Guan Gui, Mohammad Hammoudeh, Haris Gacanin, and Darren Dancey. Optimizing deep learning model hyperparameters for botnet attack detection in iot networks. *IEEE Internet of Things Journal*, 2022.
- Chaw Su Htwe, Yee Mon Thant, and Mie Mie Su Thwin. Botnets attack detection using machine learning approach for iot environment. *Journal of Physics: Conference Series*, 2020. URL <https://iopscience.iop.org/article/10.1088/1742-6596/1646/1/012101>.
- Thanh Cong Tran and Tran Khanh Dang. Machine learning for multi-classification of botnet attacks. In *2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, Seoul, Korea, 2022. IEEE. doi: 10.1109/IMCOM53663.2022.9721811. URL <https://ieeexplore.ieee.org/document/9721811>.
- Gonzalo De La Torre Parra, Paul Rad, Kim-Kwang Raymond Choo, and Nicole Beebe. Detecting internet of things attacks using distributed deep learning. *Journal of Network and Computer Applications*, 163, 2020. URL <https://doi.org/10.1016/j.jnca.2020.102662>.
- T. Gopalakrishnan, D. Ruby, Fadi Al-Turjman, Deepak Gupta, Irina V. Pustokhina, Denis A. Pustokhin, and K. Shankar. Deep learning enabled data offloading with cyber attack detection model in mobile edge computing systems. *IEEE Access*, 8, 2020. doi: 10.1109/ACCESS.2020.3030726.

- Zhihong Tian, Chaochao Luo, Jing Qiu, Xiaojiang Du, and Mohsen Guizani. A distributed deep learning system for web attack detection on edge devices. *IEEE Transactions on Industrial Informatics*, 16:1963–1971, 2020. doi: 10.1109/TII.2019.2938778.
- Souradip Roy, Juan Li, and Yan Bai. A two-layer fog-cloud intrusion detection model for iot networks. *Internet of Things*, 19, 2022. doi: <https://doi.org/10.1016/j.iot.2022.100557>.
- Ana Paula Dalla Corte, Deivison Venicio Souza, Franciel Eduardo Rex, Carlos Roberto Sanquetta, Midhun Mohan, Carlos Alberto Silva, Angelica Maria Almeyda Zambrano, Gabriel Prata, Danilo Roberti Alves de Almeida, Jonathan William Trautemüller, Carine Klauberg, Anibal de Moraes, Mateus N. Sanquetta, Ben Wilkinson, and Eben North Broadbent. Forest inventory with high-density uav-lidar: Machine learning approaches for predicting individual tree attributes. *Computers and Electronics in Agriculture - Elsevier*, 179, 2020.
- Antonio Lazaro, Ramon Villarino, and David Girbau. A survey of nfc sensors based on energy harvesting for iot applications. *Sensors (Switzerland)*, 18(11), 2018.
- Alberto Ceselli, Marco Premoli, and Stefano Secci. Mobile edge cloud network design optimization. *IEEE/ACM Transactions on Networking*, 25(3):1818–1831, 2017.
- M. Anandhalli and V. P. Baligar. A novel approach in real-time vehicle detection and tracking using raspberry pi. *Alex. Eng. J.*, 57:1597–1607, 2018.
- Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. *Proceedings - 2018 3rd ACM/IEEE Symposium on Edge Computing*, pages 159–173, 2018b.
- R. L. Devi and V. Kalaivani. Machine learning and iot-based cardiac arrhythmia diagnosis using statistical and dynamic features of ecg. *J Supercomput*, 76:6533–6544, 2020.
- Otatile Khutsoane, Basseyy Isong, and Adnan M Abu-Mahfouz. Iot devices and applications based on lora/lorawan. *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 6107–6112, 2017.
- Inigo Alonso, Luis Riazuelo, Luis Montesano, and Ana C. Murillo. 3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation. *IEEE Robotics and Automation Letters*, 5, 2020.
- Zhuo Chen, Wenlu Hu, Junjue Wang, Siyan Zhao, Brandon Amos, Guanhang Wu, Kiryong Ha, Khalid Elgazzar, Padmanabhan Pillai, Roberta Klatzky, Daniel Siewiorek, and Mahadev Satyanarayanan. An empirical study of latency in an emerging class of

- edge computing applications for wearable cognitive assistance. *SEC '17: Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pages 1–14, 2017. URL <https://doi.org/10.1145/3132211.3134458>.
- Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. Mcunet: Tiny deep learning on iot devices. *ArXiv:2007.10319*, 2020.
- Rafal Kozika, Michal Choras, Massimo Ficcob, and Francesco Palmieric. A scalable distributed machine learning approach for attack detection in edge computing environments. *Journal of Parallel and Distributed Computing*, 119:18–26, 2018.
- Muaadh A. Alsoufi, Shukor Razak, Maheyazah Md Siraj, Ibtehal Nafea, Fuad A. Ghaleb, Faisal Saeed, and Maged Nasser. Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review. *Applied Sciences (Switzerland)*, 11, 2021.
- Mahbod Tavallae, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. *Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.
- Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. An empirical comparison of botnet detection methods. *Computers and Security Journal*, 45:100–123, 2014. URL <http://dx.doi.org/10.1016/j.cose.2014.05.011>.
- Matin Farhoumandi, Quan Zhou, and Mohammad Shahidehpour. A review of machine learning applications in iot-integrated modern power systems. *The Electricity Journal*, 34, 2020.
- Hamidreza Bolhasani, Maryam Mohseni, and Amir Masoud Rahmani. Deep learning applications for iot in health care: A systematic review. *Informatics in Medicine Unlocked*, 23, 2021. URL <https://doi.org/10.1016/j.imu.2021.100550>.
- Kanchan Pradhan and Priyanka Chawla. Medical internet of things using machine learning algorithms for lungcancer detection. *Journal of Management Analytics*, 7(4):591–623, 2020. URL <https://doi.org/10.1080/23270012.2020.1811789>.
- Safa Ben Atitallah, Maha Driss, Wadii Boulila, and Henda Ben Ghézala. Leveraging deep learning and iot big data analytics to support the smart cities development: Review and future directions. *Computer Science Review*, 38, 2020. URL <https://doi.org/10.1016/j.cosrev.2020.100303>.
- Rozeha A. Rashid, Leon Chin, M.A. Sarijari, Rubita Sudirman, and Teruji Ide. Machine learning for smart energy monitoring of home appliances using iot. *2019 Eleventh*

- International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 66–71, 2019.
- Dan Popa, Florin Pop, Cristina Serbanescu, and Aniello Castiglione. Deep learning model for home automation and energy reduction in a smart home environment platform. *Neural Computing and Applications*, 31:1317–1337, 2019. URL <https://doi.org/10.1007/s00521-018-3724-6>.
- Wei Liu and Yozo Shoji. Applying deep recurrent neural network to predict vehicle mobility. *2018 IEEE Vehicular Networking Conference (VNC)*, pages 1–6, 2018.
- Xuan Song, Hiroshi Kanasugi, and Ryosuke Shibasaki. Deeptransport: prediction and simulation of human mobility and transportation mode at a citywide level. *IJCAI'16: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2618—2624, 2016.
- Maede Zolanvari, Marcio A. Teixeira, Lav Gupta, Khaled M. Khan, and Raj Jain. Machine learning-based network vulnerability analysis of industrial internet of things. *IEEE Internet of Things Journal*, 6(4):6822—6834, 2019.
- Ali Shakarami, Mostafa Ghobaei-Arani, and Ali Shahidinejad. A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. *Computer Networks - Elsevier*, 182, 2020.
- Mehrdad Khani, Pouya Hamadian, Arash Nasr-Esfahany, and Mohammad Alizadeh. Real-time video inference on edge devices via adaptive model streaming. *ArXiv:2006.06628*, 2020.
- Matthias Feurer and Frank Hutter. *Automated Machine Learning*. Springer, 2019.
- Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.
- Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 2019.
- Jeeyung Kim, Alex Sim, Jinoh Kim, and Kesheng Wu. Botnet detection using recurrent variational autoencoder. In *GLOBECOM 2020 - IEEE Global Communications Conference*, pages 1–6, 2020.
- Yanqing Yang, Kangfeng Zheng, Chunhua Wu, and Yixian Yang. Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. *Sensors*, 19, 2019.

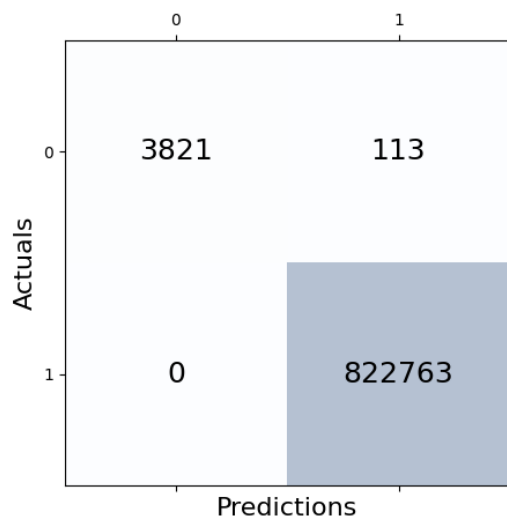
- Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot. *Sensors*, 17, 2017.
- Youngrok Song, Sangwon Hyun, and Yun-Gyung Cheong. Analysis of autoencoders for network intrusion detection. *Sensors*, 21, 2021.
- Andressa A. Cunha, João B. Borges, and Antonio A. F. Loureiro. Detecção de ataques de botnets em iot via variational autoencoder. *VI Workshop de Computação Urbana (CoUrb) do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2022a.
- Aji P. Wibawa, Agung B. P. Utama, Hakkun Elmunyah, Utomo Pujianto, Felix A. Dwiyanto, and Leonel Hernandez. Time-series analysis with smoothed convolutional neural network. *Journal of Big Data*, 9(44), 2022. URL <https://doi.org/10.1186/s40537-022-00599-y>.
- Hongpo Zhanga, Lulu Huanga, Chase Q. Wu, and Zhanbo Lia. An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset. *Computer Networks*, 177, 2020. URL <https://doi.org/10.1016/j.comnet.2020.107315>.
- Andressa A. Cunha, João B. Borges, and Antonio A. F. Loureiro. Classification of botnet attacks in iot using a convolutional neural network. *Q2SWinet '22: Proceedings of the 18th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, pages 63–70, 2022b. URL <https://doi.org/10.1145/3551661.3561374>.
- Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572, 2016.

Appendix A

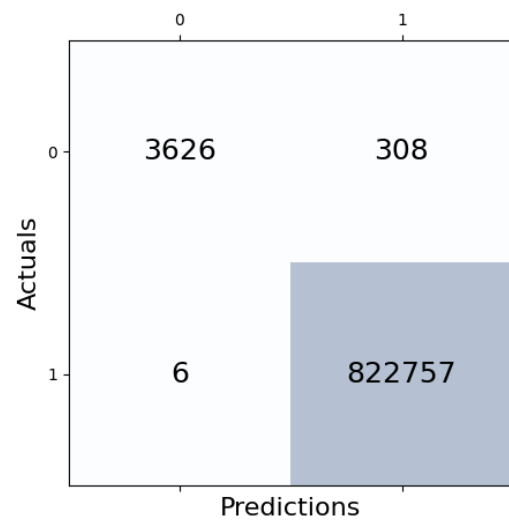
Botnet Attack Detection Results

Figure A.1: Confusion matrix of the attack detection on Ecobee Thermostat model.

(a) Ecobee Thermostat with 115 features

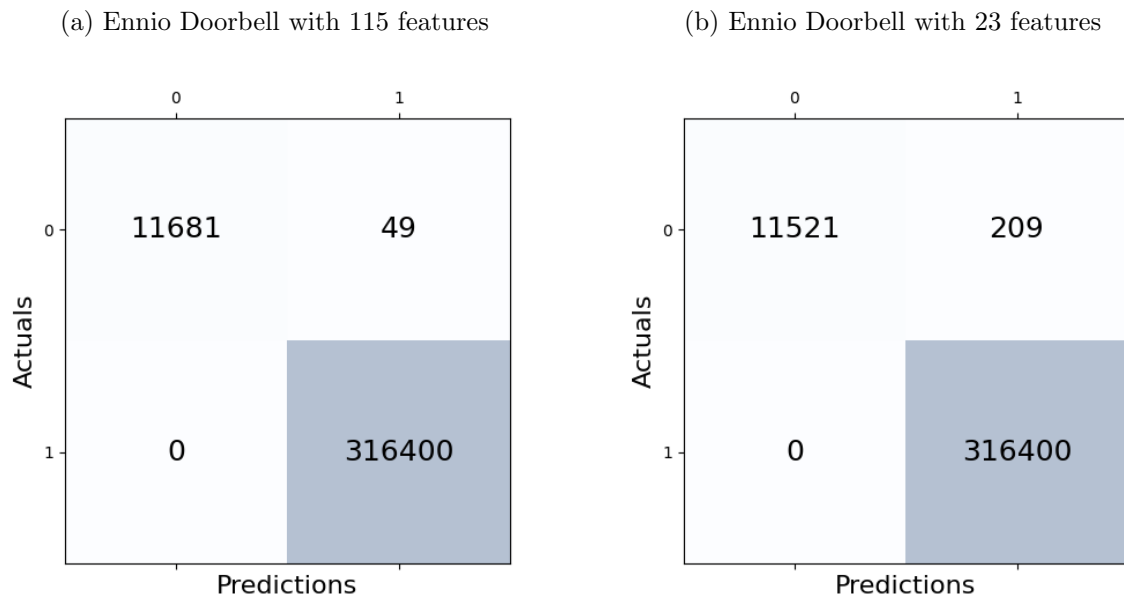


(b) Ecobee Thermostat with 23 features



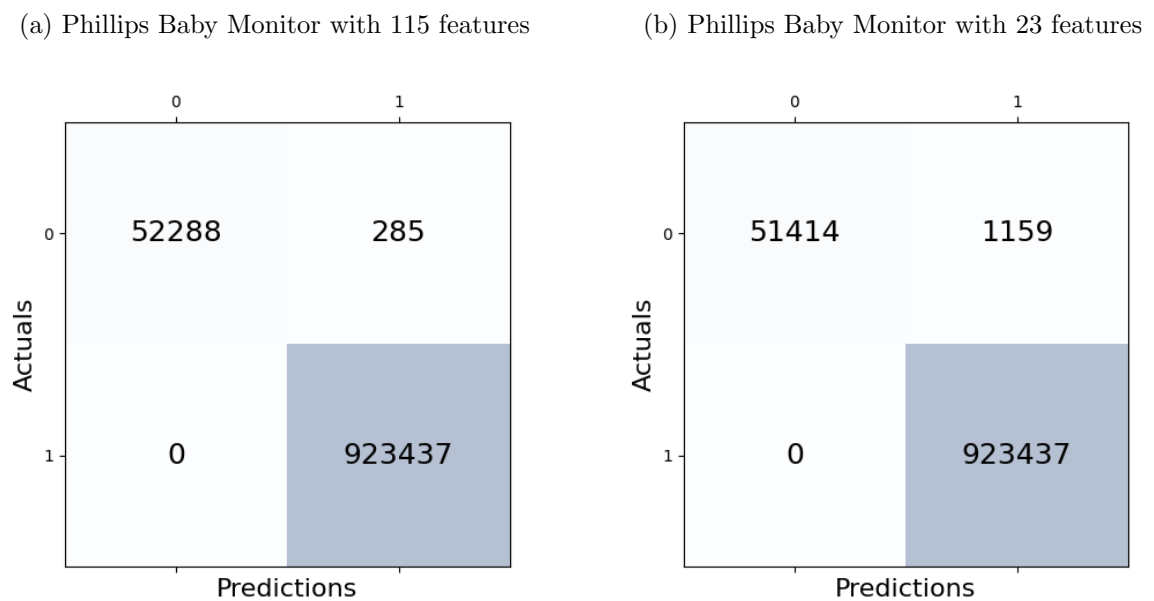
Source: Elaborated by the author.

Figure A.2: Confusion matrix of the attack detection on Ennio Doorbell model.



Source: Elaborated by the author.

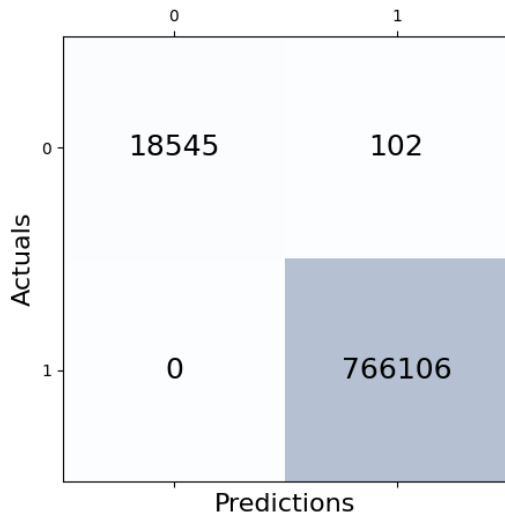
Figure A.3: Confusion matrix of the attack detection on Phillips Baby Monitor model.



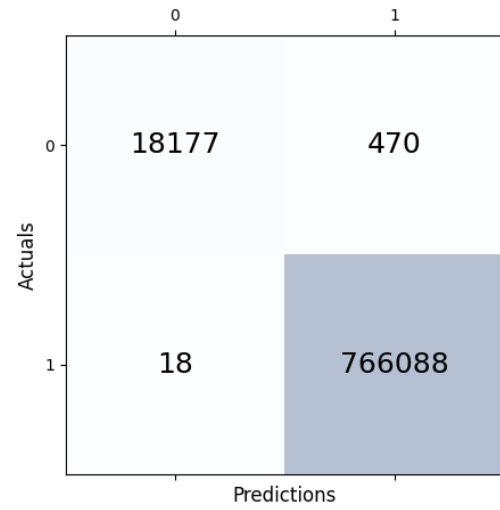
Source: Elaborated by the author.

Figure A.4: Confusion matrix of the attack detection on P737 Security Camera model.

(a) Provision 737 Camera with 115 features



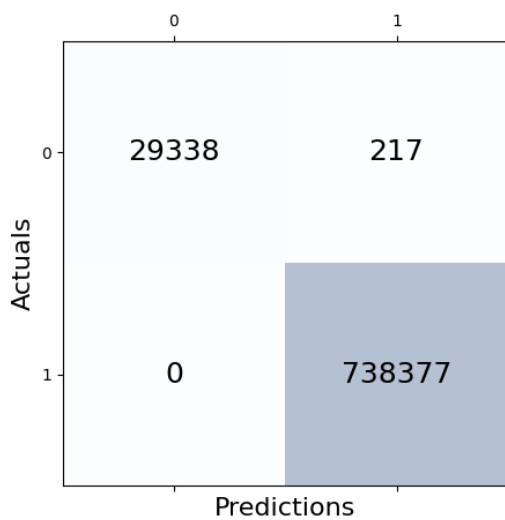
(b) Provision 737 Camera with 23 features



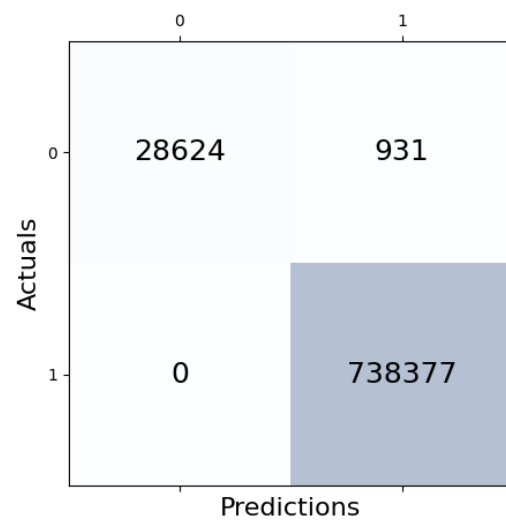
Source: Elaborated by the author.

Figure A.5: Confusion matrix of the attack detection on P828 Security Camera model.

(a) Provision 838 Camera with 115 features



(b) Provision 838 Camera with 23 features

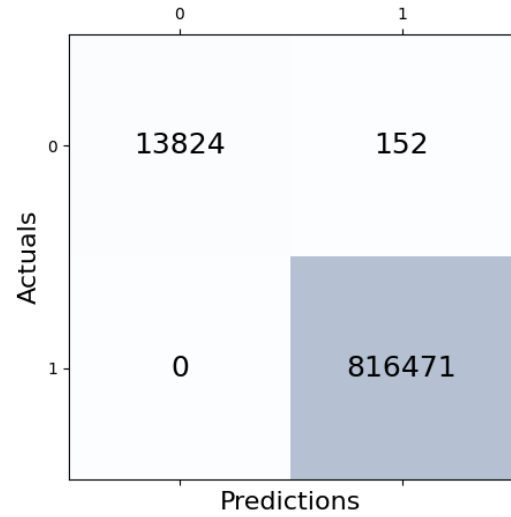
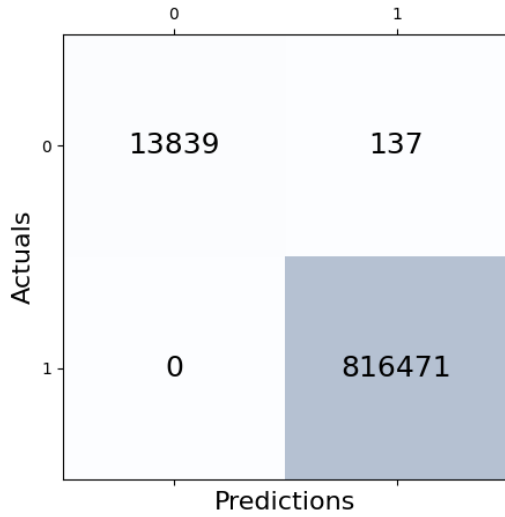


Source: Elaborated by the author.

Figure A.6: Confusion matrix of the attack detection on S1002 Security Camera model.

(a) SimpleHome 1002 Camera with 115 features

(b) SimpleHome 1002 Camera with 23 features

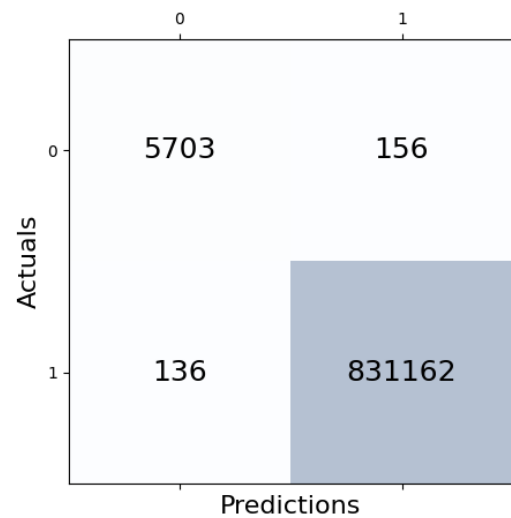
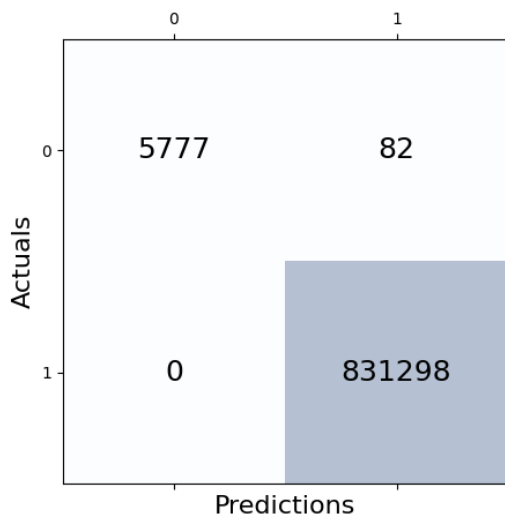


Source: Elaborated by the author.

Figure A.7: Confusion matrix of the attack detection on S1003 Security Camera model.

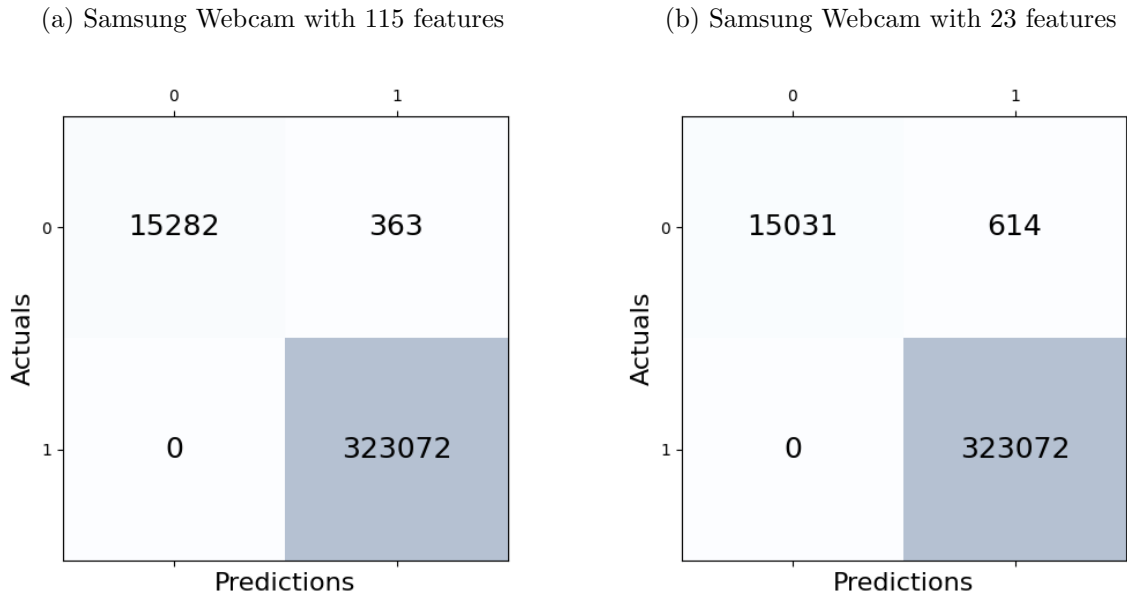
(a) SimpleHome 1003 Camera with 115 features

(b) SimpleHome 1003 Camera with 23 features



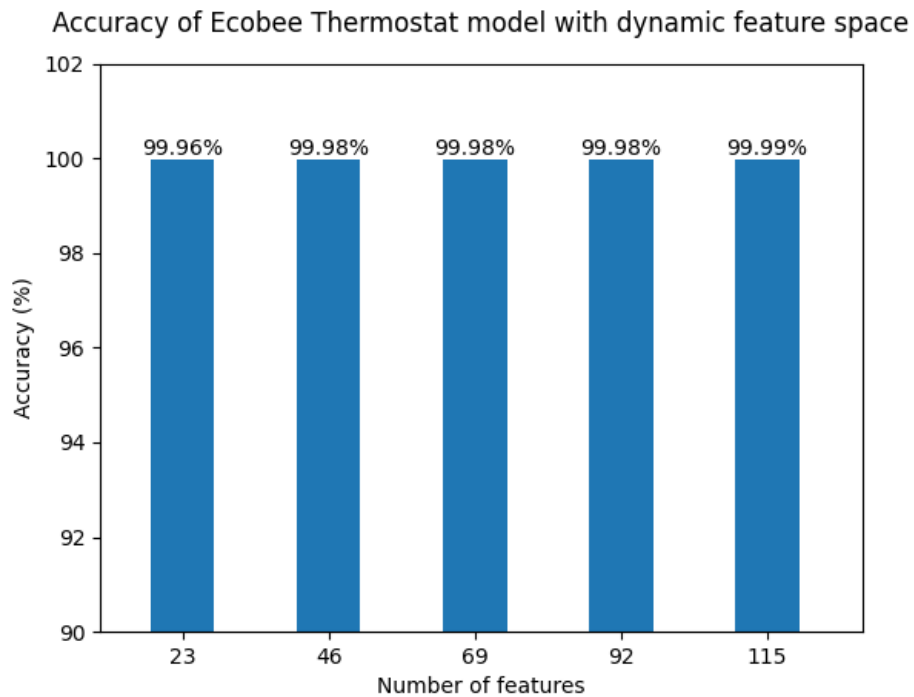
Source: Elaborated by the author.

Figure A.8: Confusion matrix of the attack detection on Samsung Webcam model.



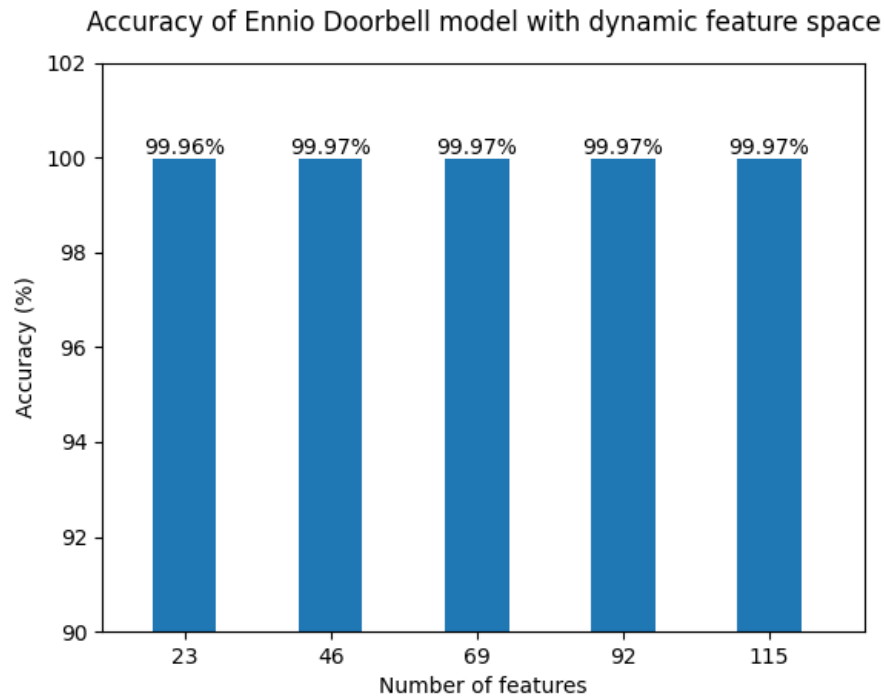
Source: Elaborated by the author.

Figure A.9: Accuracy results of Ecobee Thermostat models on the attack detection task with dynamic feature space.



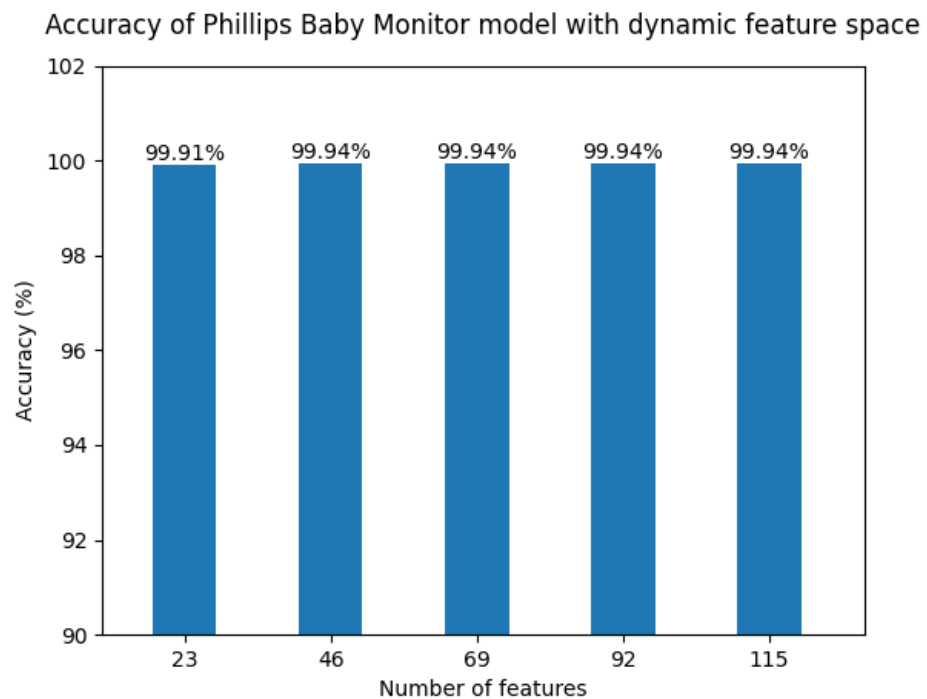
Source: Elaborated by the author.

Figure A.10: Accuracy results of Ennio Doorbell models on the attack detection task with dynamic feature space.



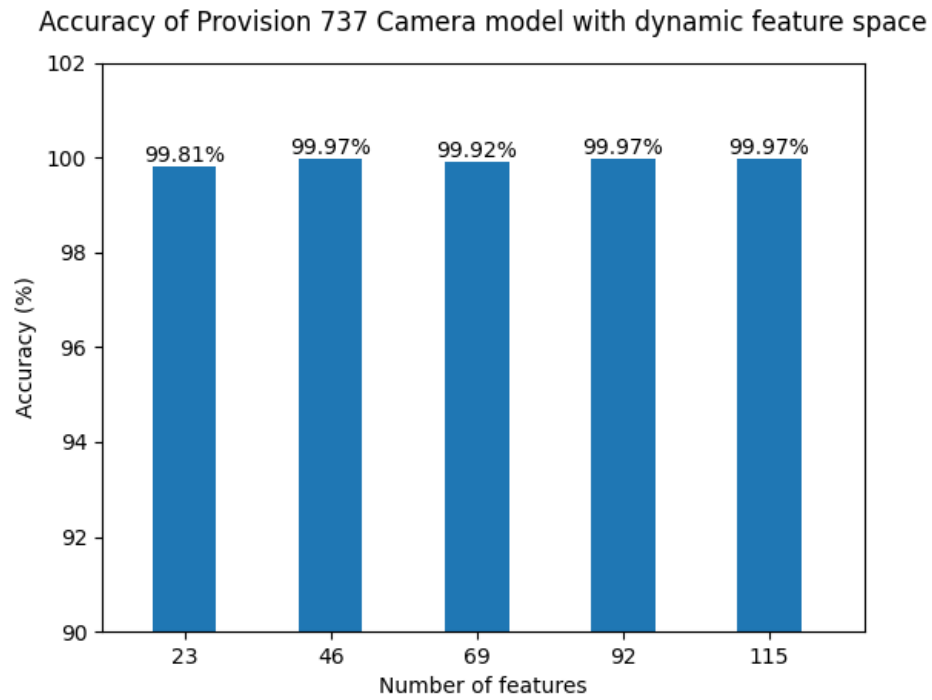
Source: Elaborated by the author.

Figure A.11: Accuracy results of Phillips Baby Monitor models on the attack detection task with dynamic feature space.



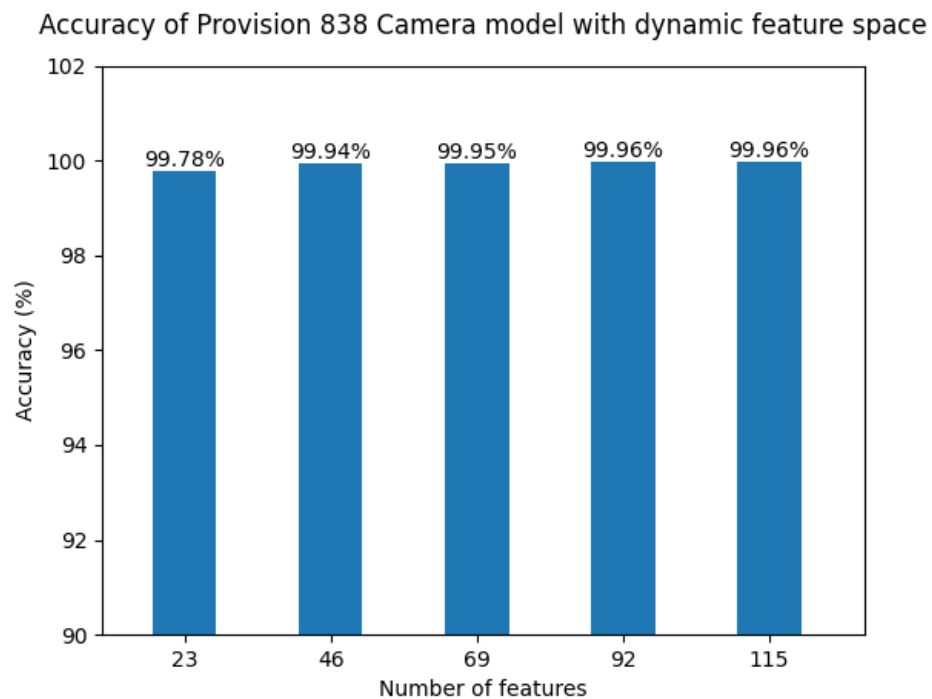
Source: Elaborated by the author.

Figure A.12: Accuracy results of Provision 737 Security Camera models on the attack detection task with dynamic feature space.



Source: Elaborated by the author.

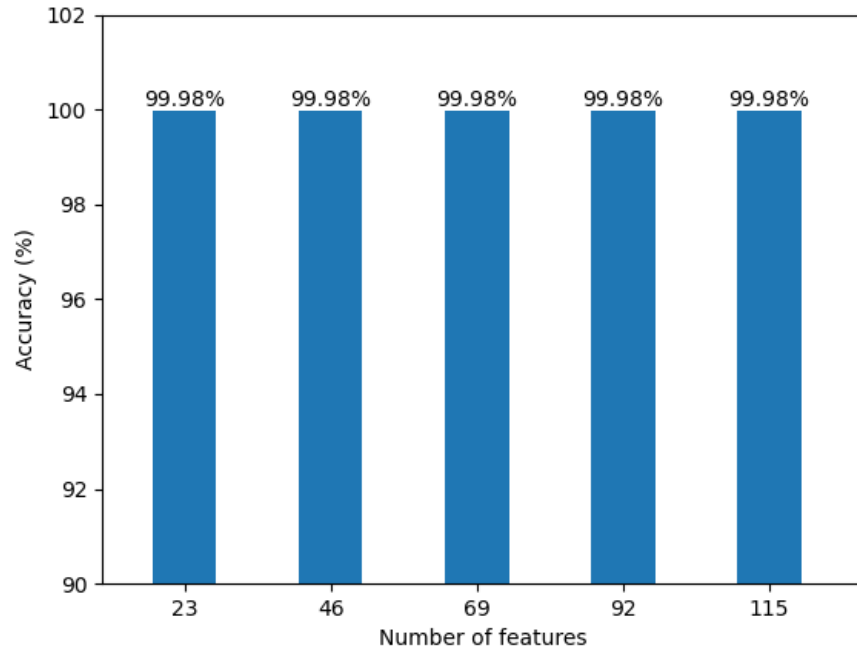
Figure A.13: Accuracy results of Provision 838 Security Camera models on the attack detection task with dynamic feature space.



Source: Elaborated by the author.

Figure A.14: Accuracy results of SimpleHome 1002 Security Camera models on the attack detection task with dynamic feature space.

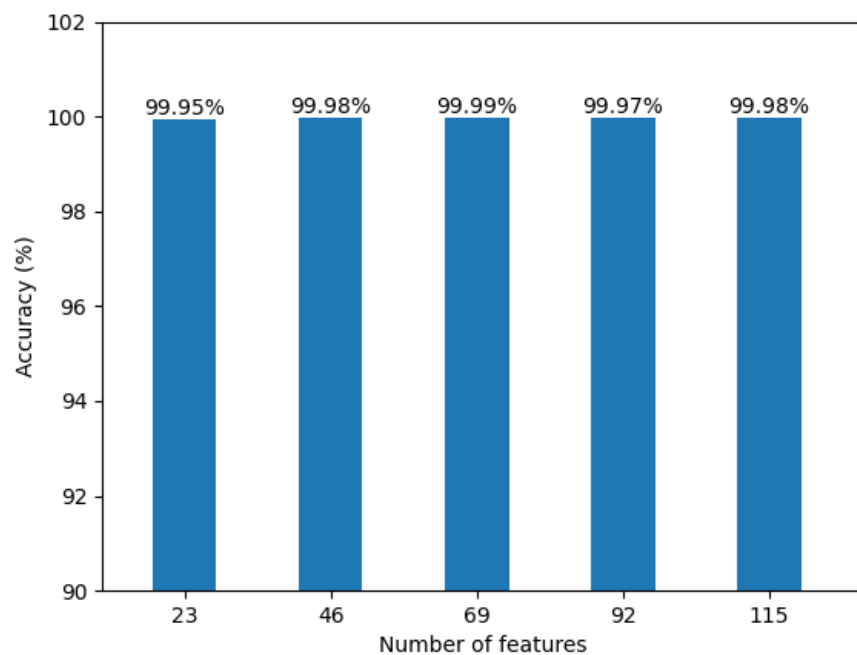
Accuracy of SimpleHome 1002 Camera model with dynamic feature space



Source: Elaborated by the author.

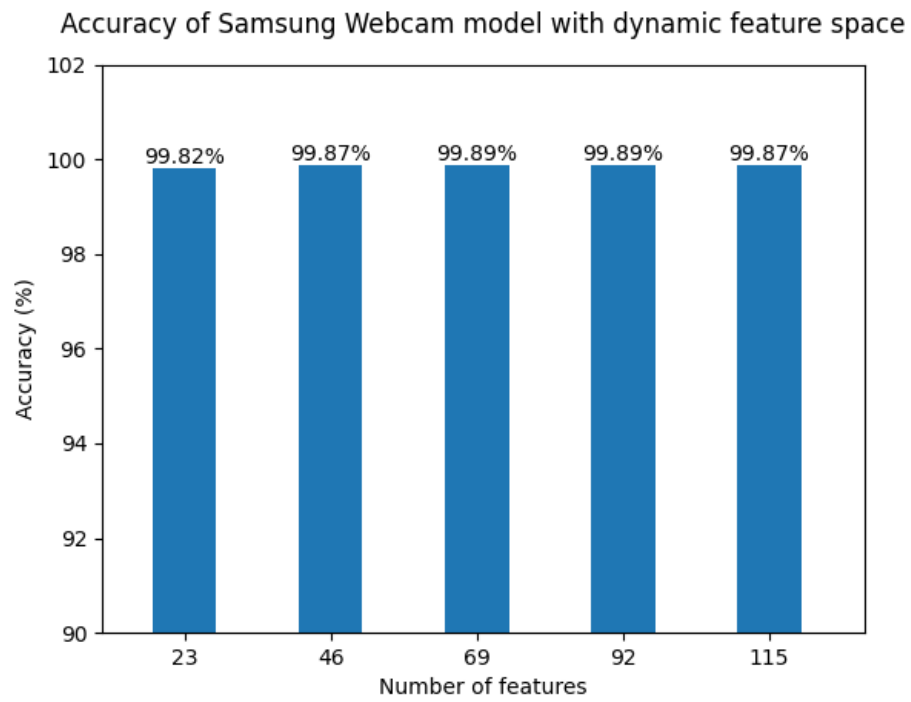
Figure A.15: Accuracy results of SimpleHome 1003 Security Camera models on the attack detection task with dynamic feature space.

Accuracy of SimpleHome 1003 Camera model with dynamic feature space



Source: Elaborated by the author.

Figure A.16: Accuracy results of Samsung Webcam models on the attack detection task with dynamic feature space.

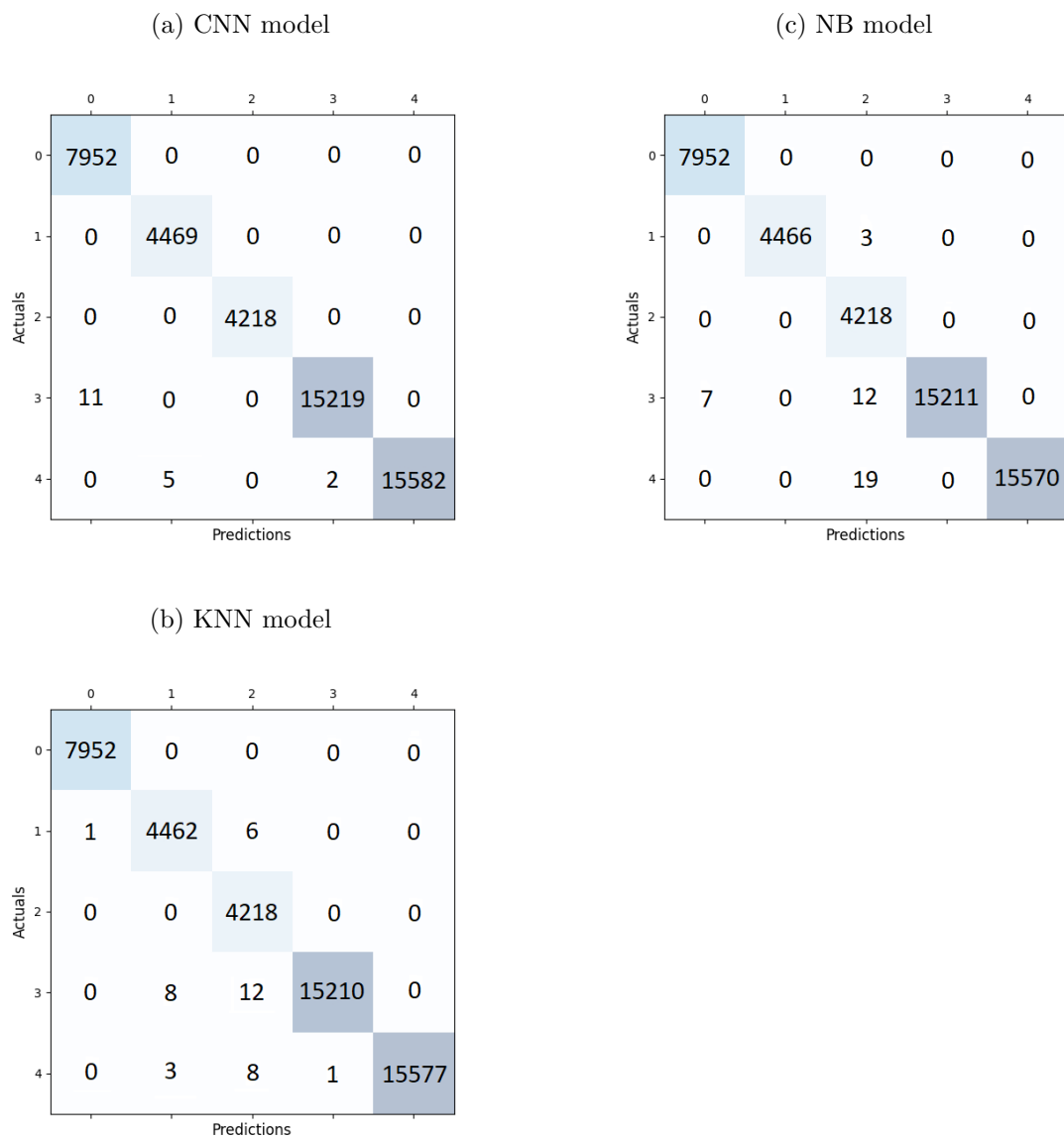


Source: Elaborated by the author.

Appendix B

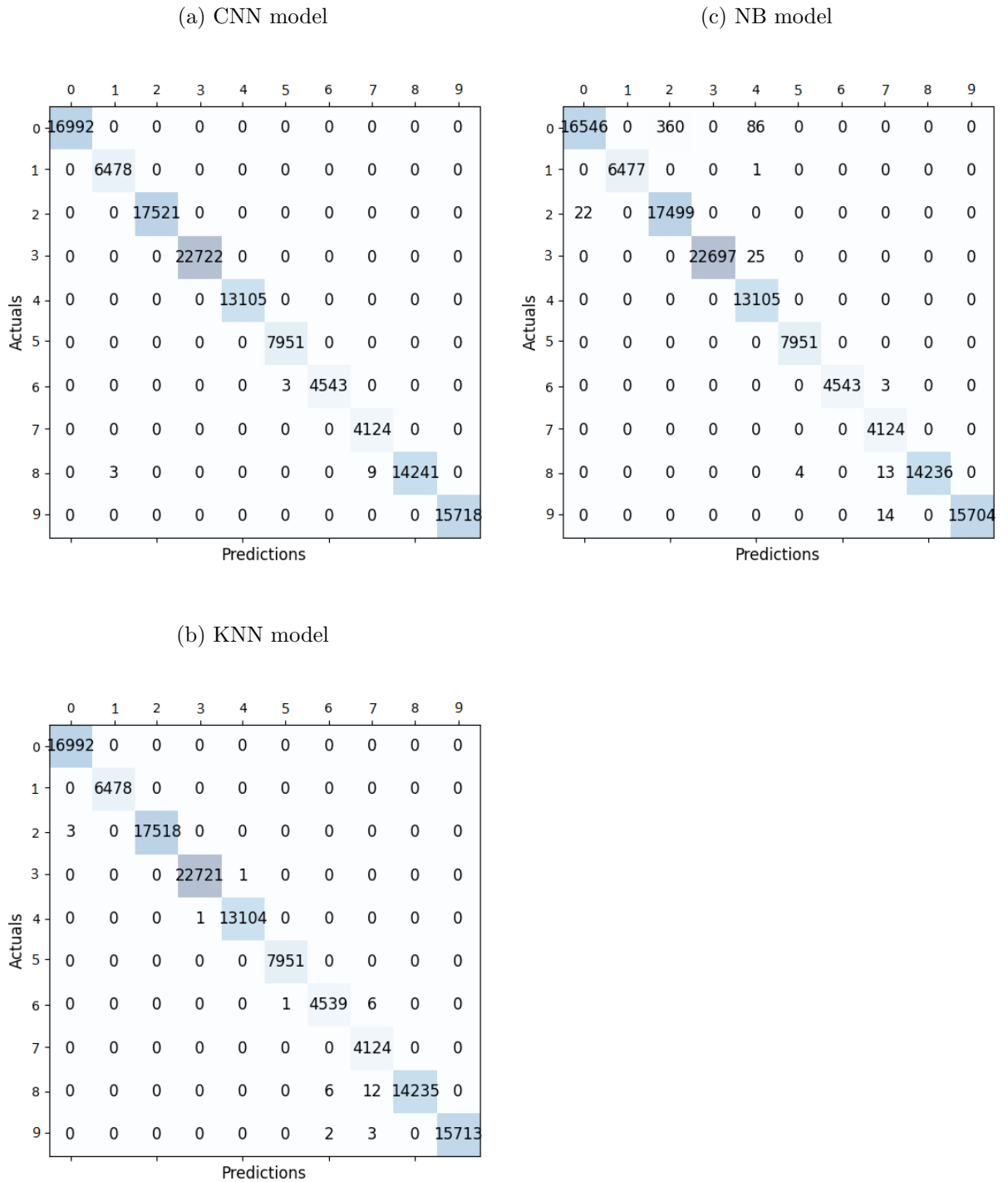
Botnet Attack Classification Results

Figure B.1: Confusion matrix of the attack classification on Ennio Doorbell with CNN, KNN and NB models.



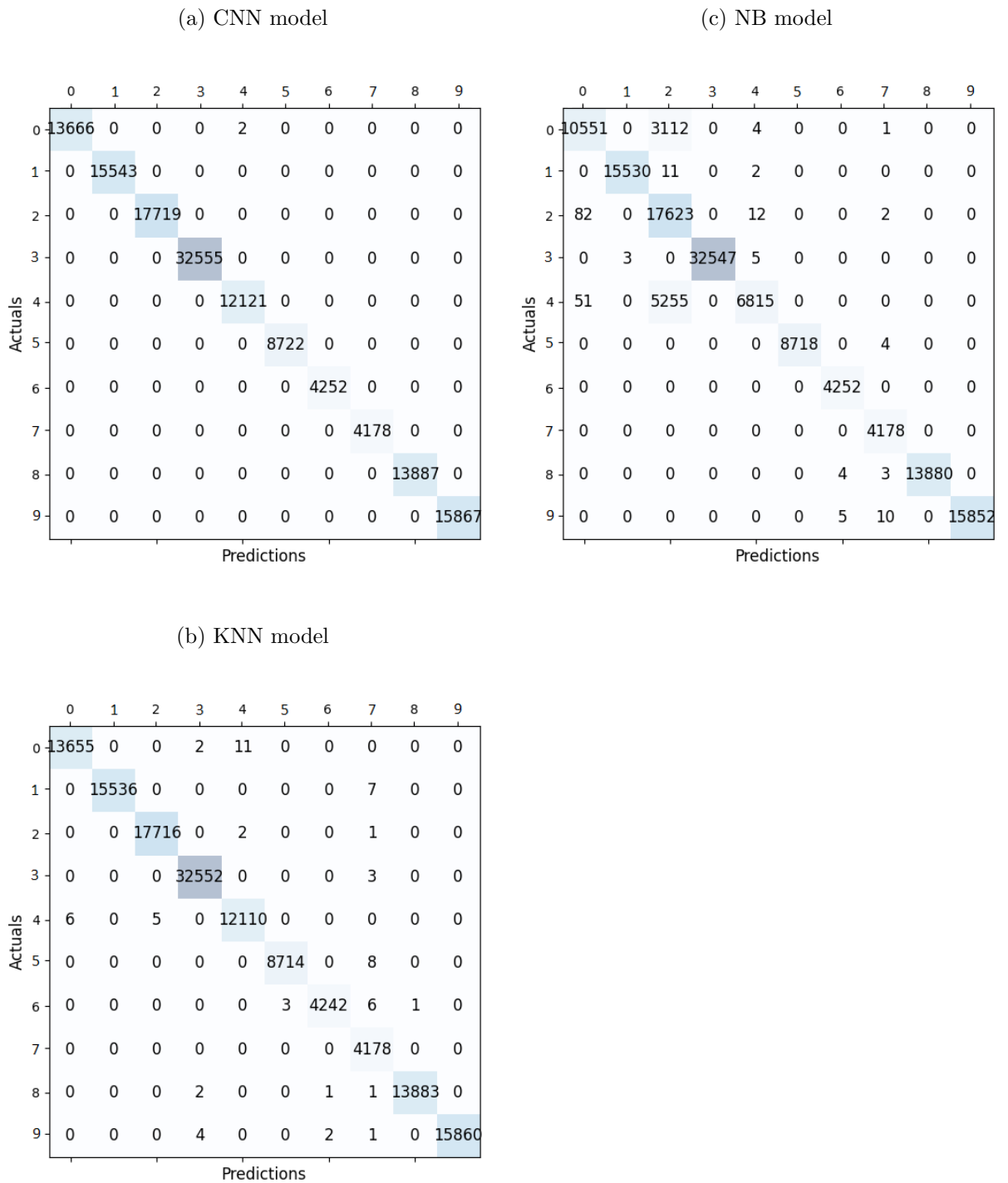
Source: Elaborated by the author.

Figure B.2: Confusion matrix of the attack classification on Ecobee Thermostat with CNN, KNN and NB models.



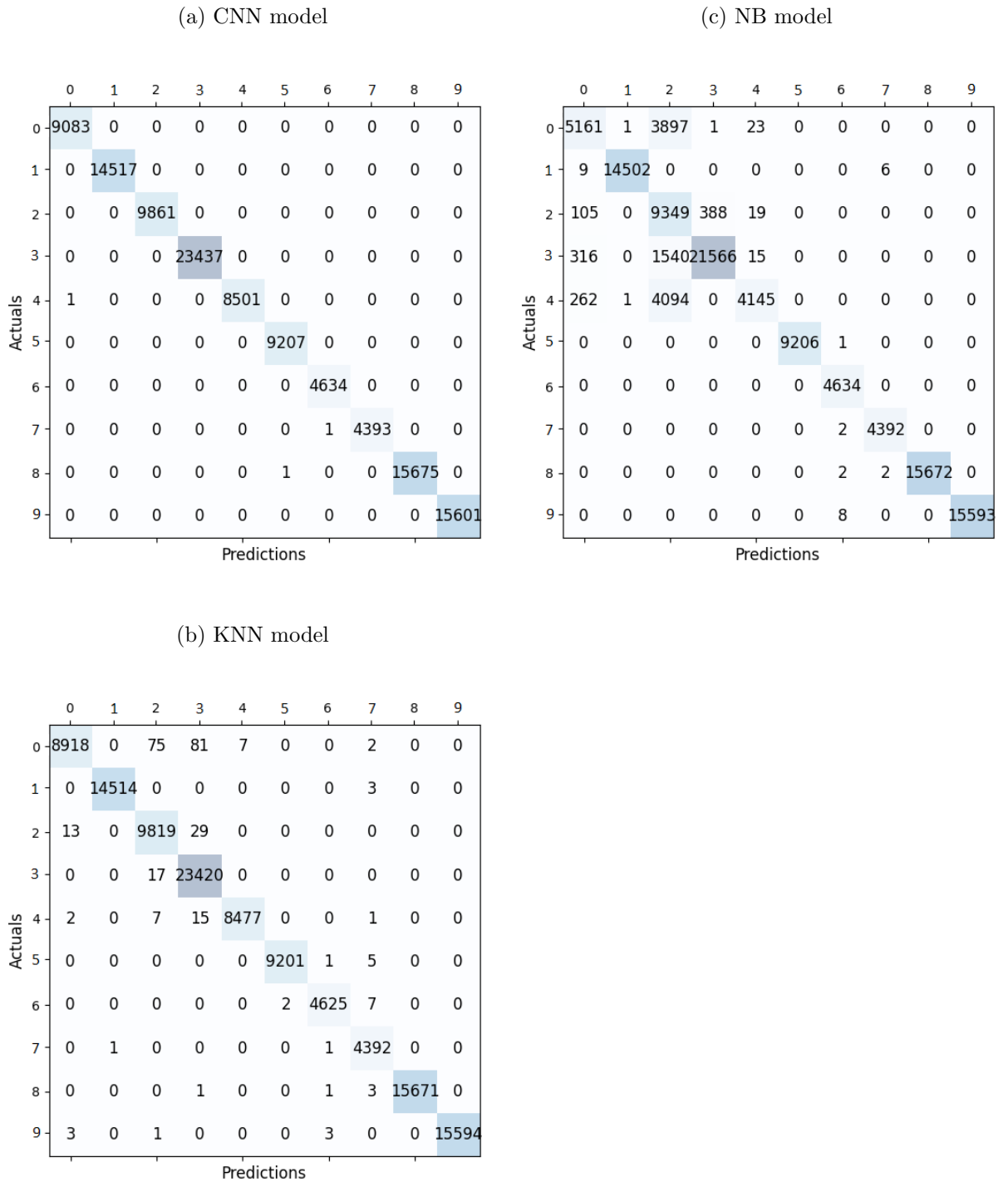
Source: Elaborated by the author.

Figure B.3: Confusion matrix of the attack classification on Phillips Baby Monitor with CNN, KNN and NB models.



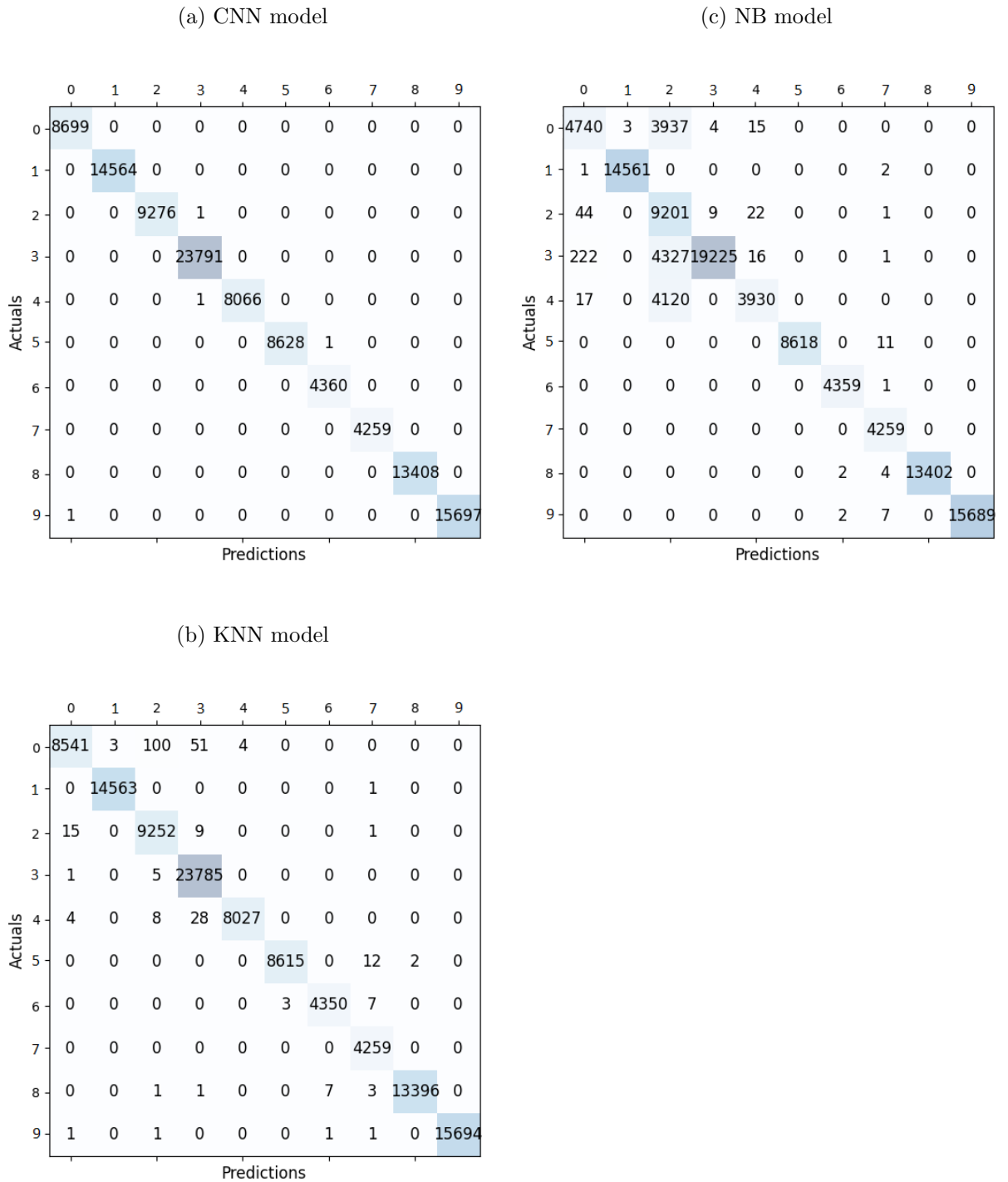
Source: Elaborated by the author.

Figure B.4: Confusion matrix of the attack classification on P737 Security Camera with CNN, KNN and NB models.



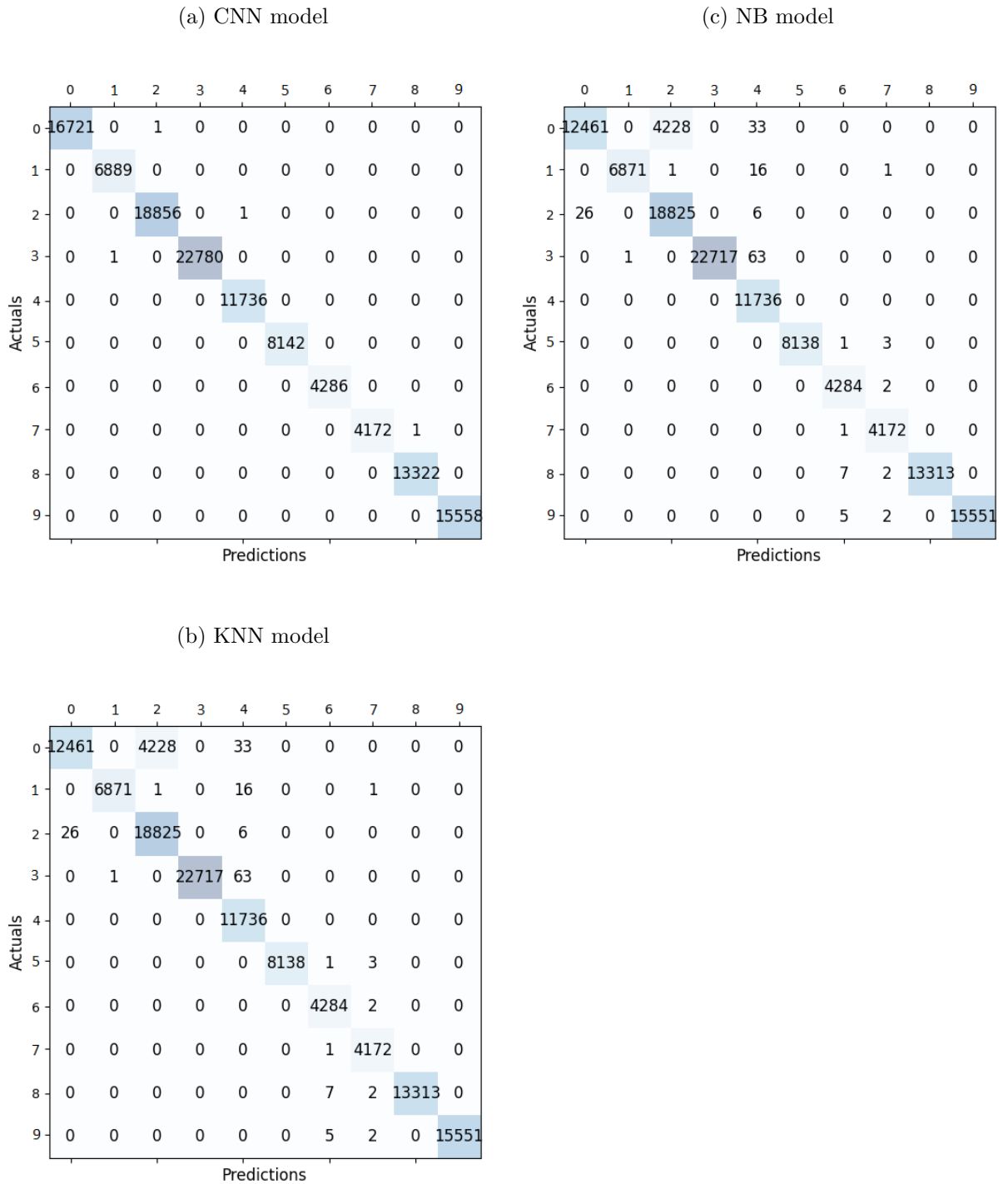
Source: Elaborated by the author.

Figure B.5: Confusion matrix of the attack classification on P838 Security Camera with CNN, KNN and NB models.



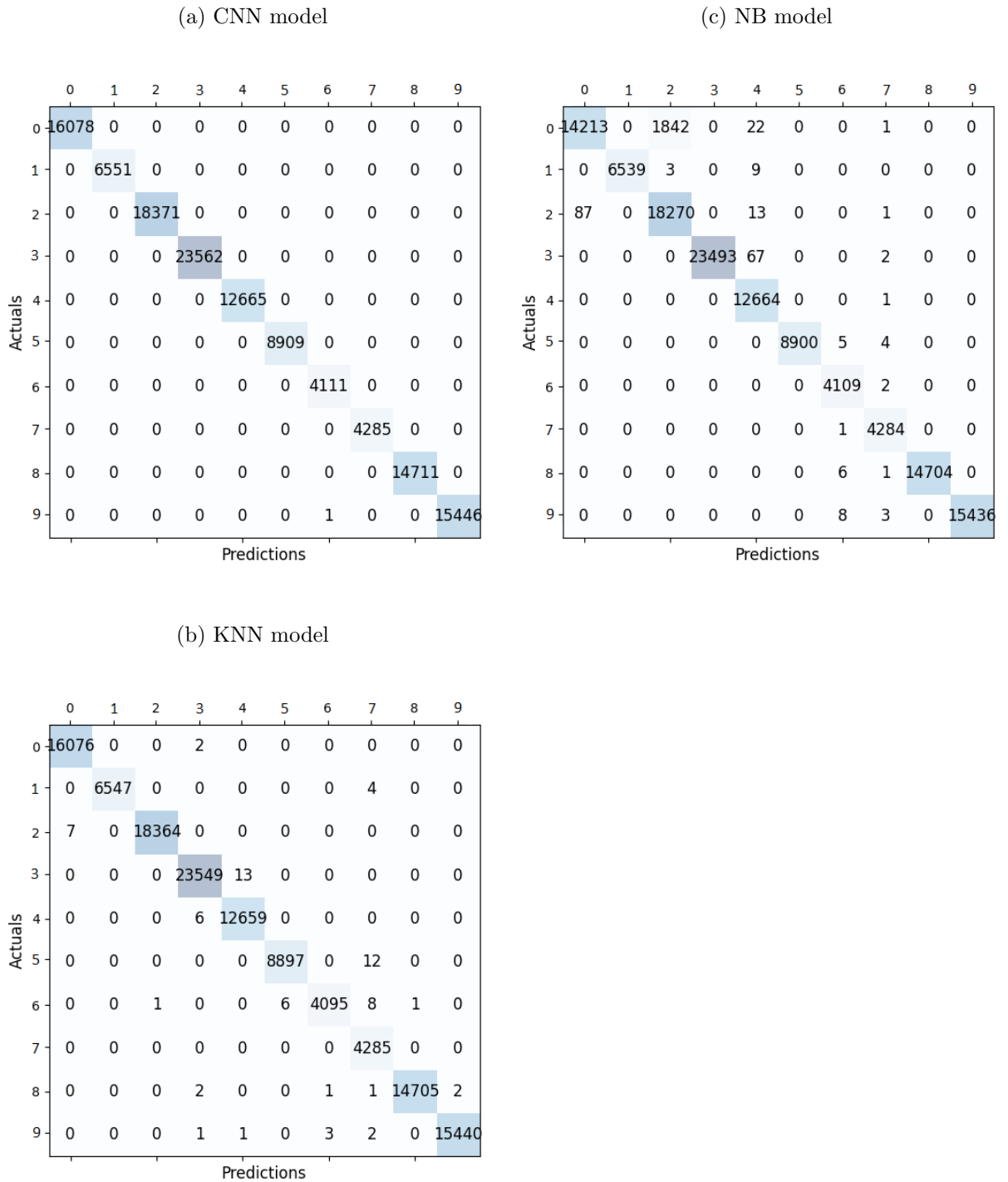
Source: Elaborated by the author.

Figure B.6: Confusion matrix of the attack classification on S1002 Security Camera with CNN, KNN and NB models.



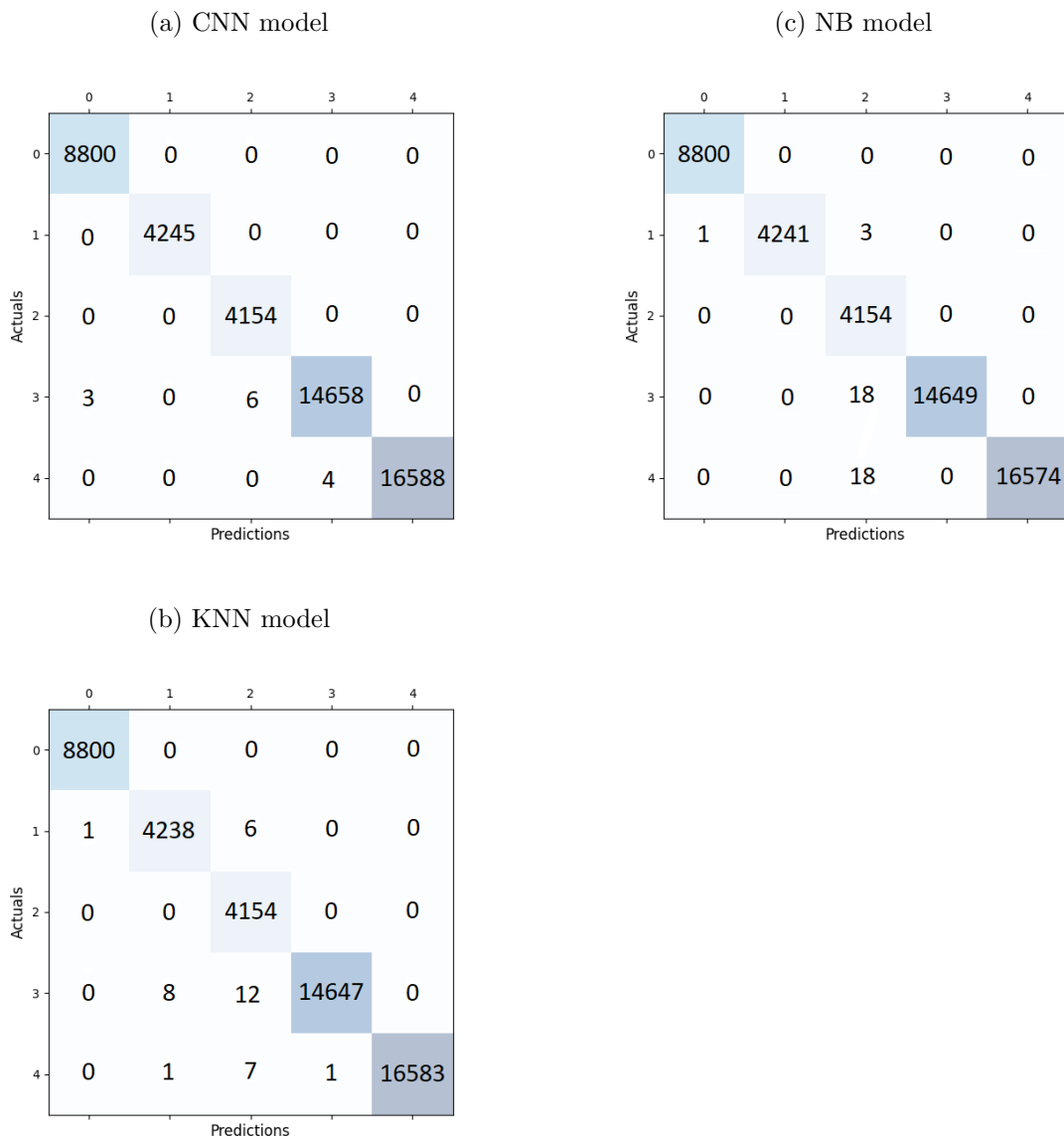
Source: Elaborated by the author.

Figure B.7: Confusion matrix of the attack classification on S1003 Security Camera with CNN, KNN and NB models.



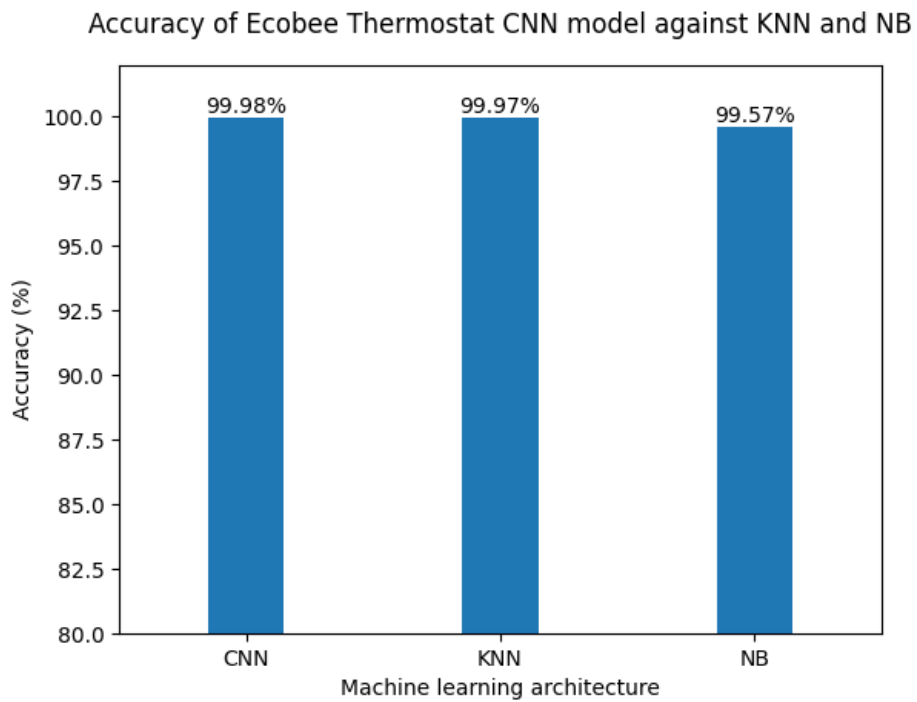
Source: Elaborated by the author.

Figure B.8: Confusion matrix of the attack classification on Samsung Webcam with CNN, KNN and NB models.



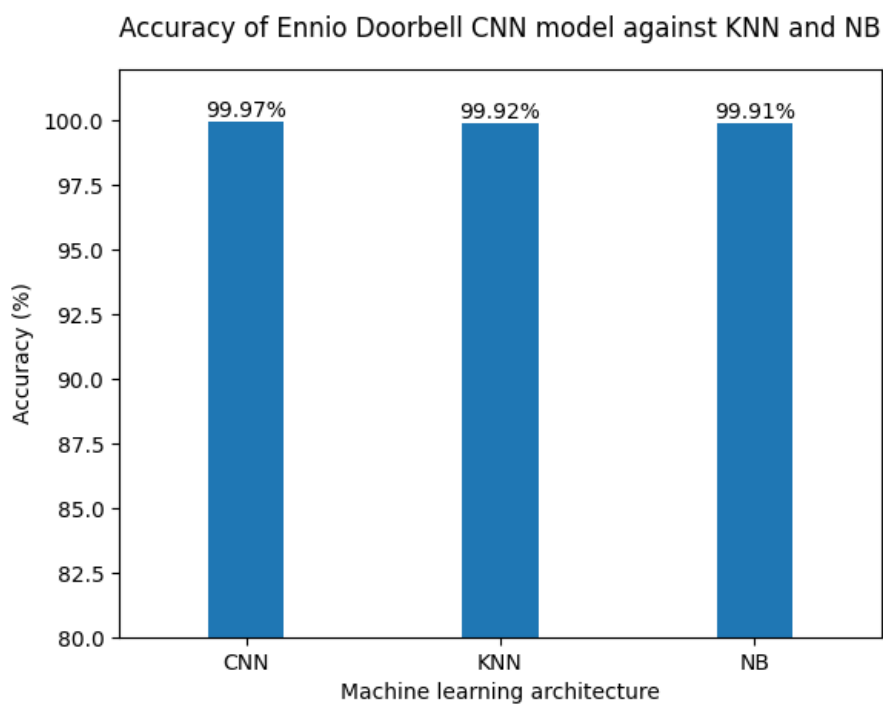
Source: Elaborated by the author.

Figure B.9: Accuracy results for Ecobee Thermostat on the attack classification task with CNN, KNN and NB models.



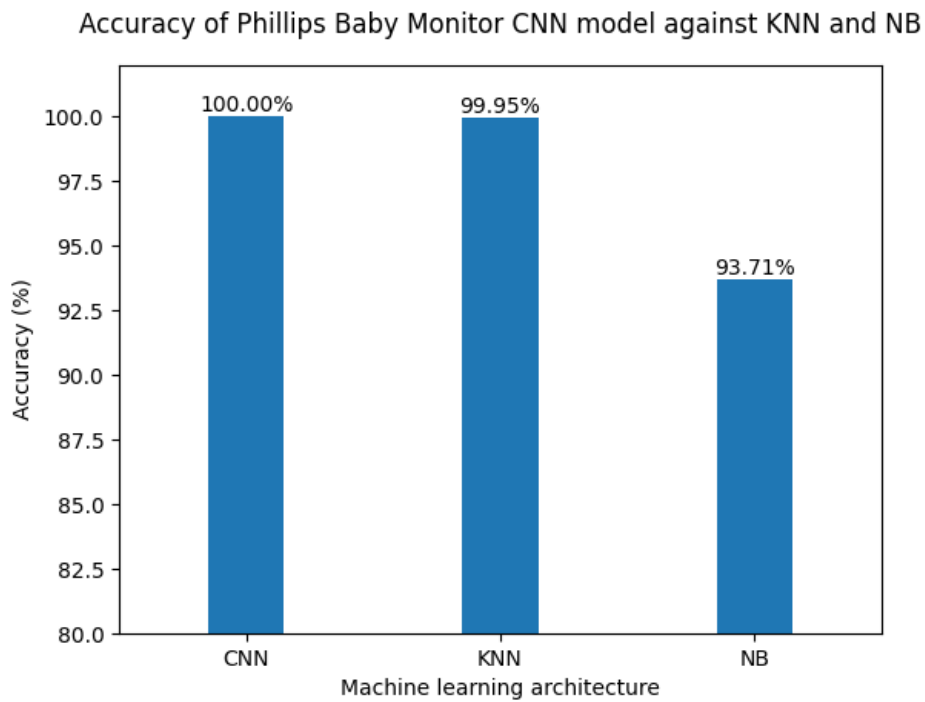
Source: Elaborated by the author.

Figure B.10: Accuracy results for Ennio Doorbell on the attack classification task with CNN, KNN and NB models.



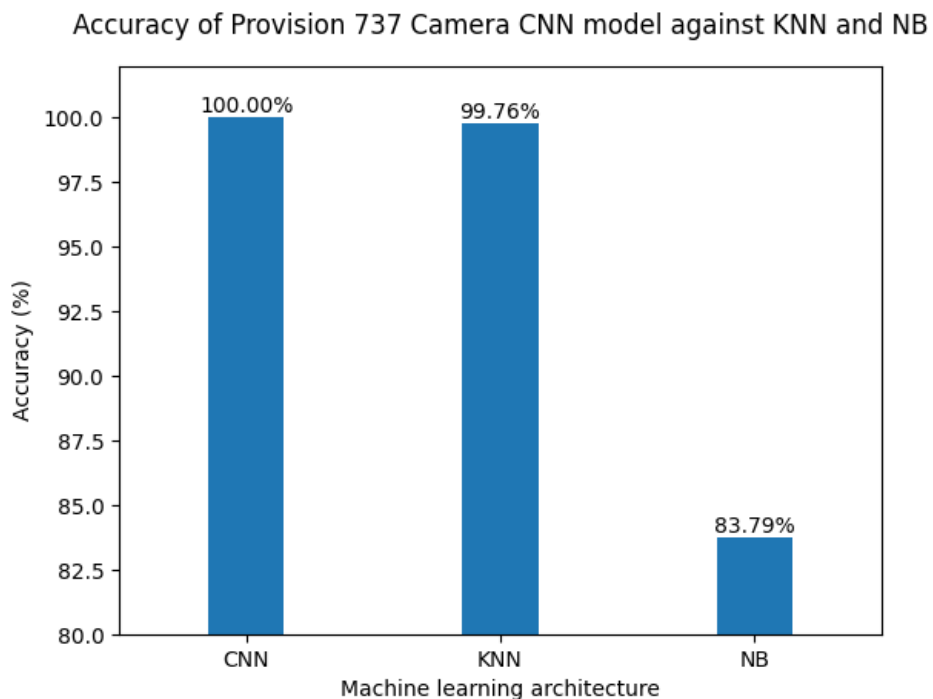
Source: Elaborated by the author.

Figure B.11: Accuracy results for Phillips Baby Monitor on the attack classification task with CNN, KNN and NB models.



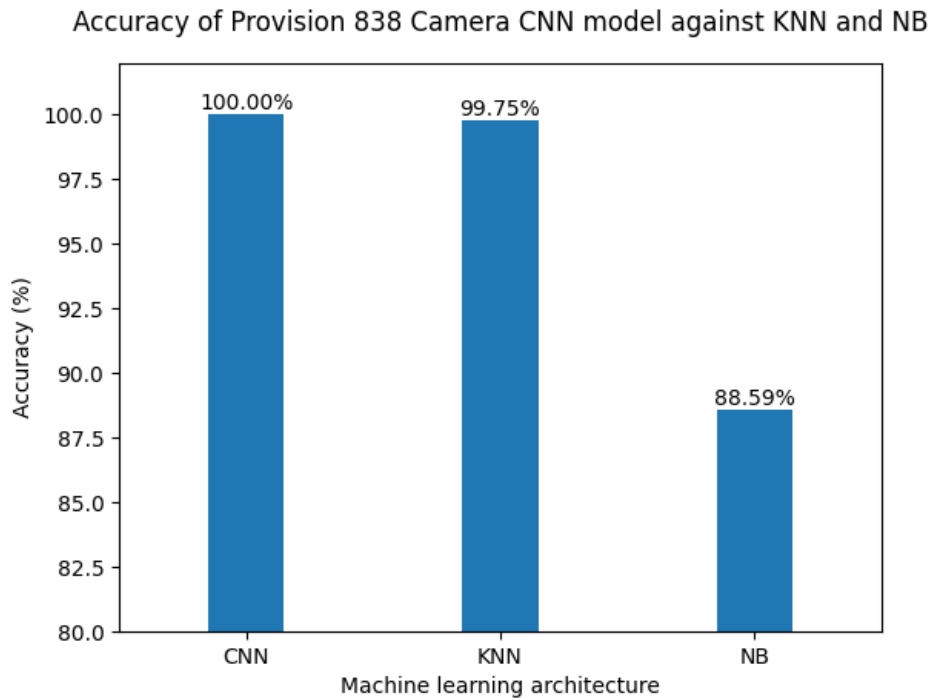
Source: Elaborated by the author.

Figure B.12: Accuracy results for Provision 737 Security Camera on the attack classification task with CNN, KNN and NB models.



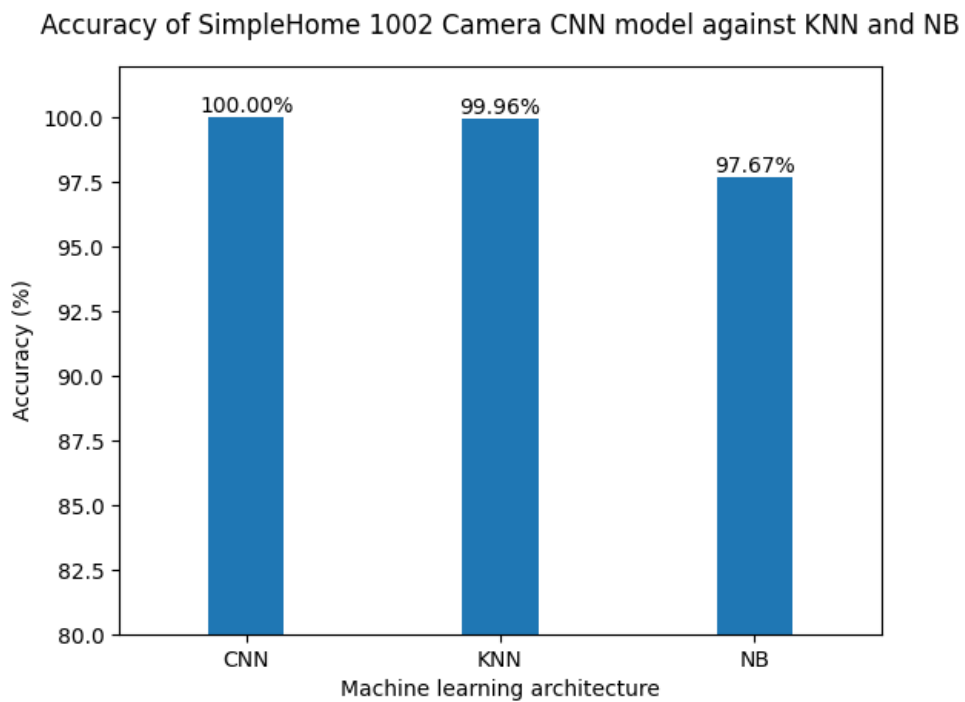
Source: Elaborated by the author.

Figure B.13: Accuracy results for Provision 838 Security Camera on the attack classification task with CNN, KNN and NB models.



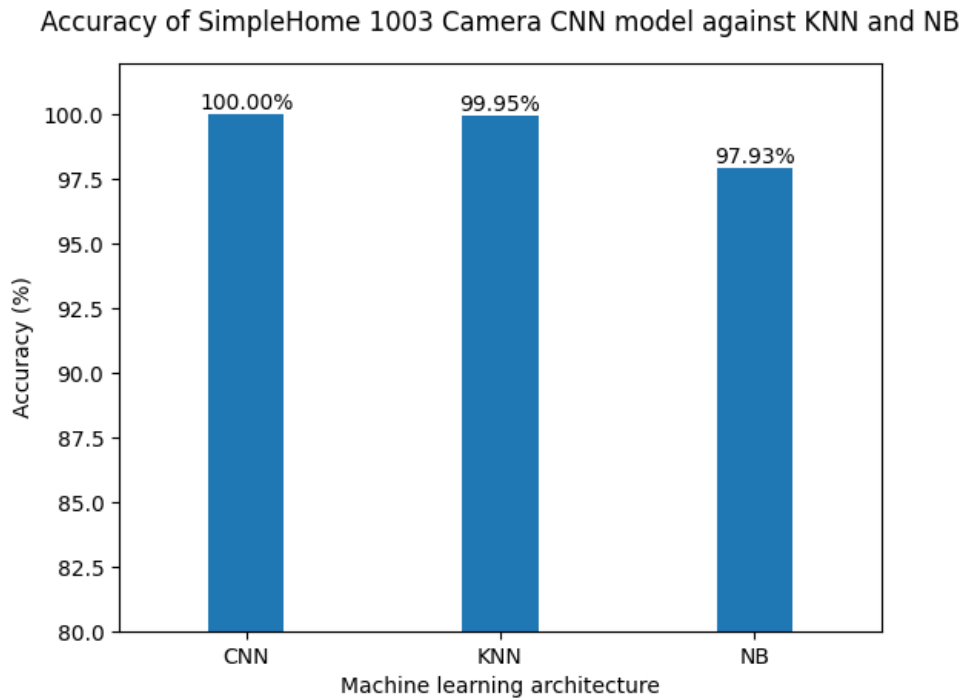
Source: Elaborated by the author.

Figure B.14: Accuracy results for SimpleHome 1002 Security Camera on the attack classification task with CNN, KNN and NB models.



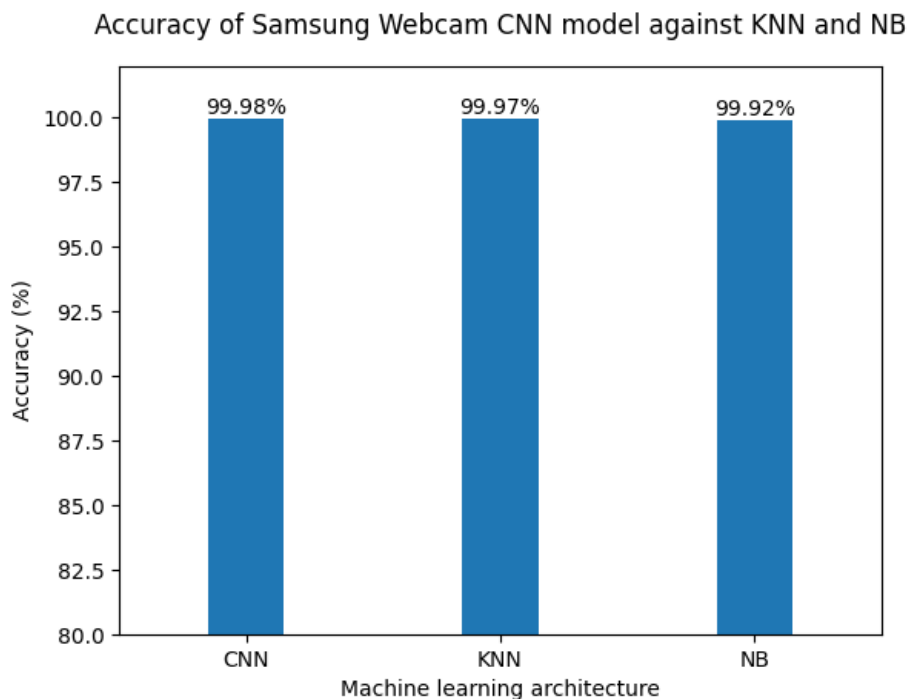
Source: Elaborated by the author.

Figure B.15: Accuracy results for SimpleHome 1003 Security Camera on the attack classification task with CNN, KNN and NB models.



Source: Elaborated by the author.

Figure B.16: Accuracy results for Samsung Webcam on the attack classification task with CNN, KNN and NB models.



Source: Elaborated by the author.