Instituto Tecnologico y de Estudios Superiores de Monterrey

Monterrey Campus

School of Engineering and Sciences

The Identification of DoS and DDoS Attacks to IoT Devices in Software Defined Networks by Using Machine Learning and Deep Learning Models

A thesis presented by

Josué Genaro Almaraz Rivera

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Monterrey, Nuevo León, May, 2022

Dedication

To God, my parents and my brothers, for being with me at every moment.

Acknowledgements

First of all, I want to say thanks to my advisors for their guidance during my masters degree studies. In addition, I would like to thank the GITA group from Universidad de Antioquia, and Aligo, for their support during my research stay in Colombia.

I want to especially say thanks to God for brought me to this moment. Thanks to my mom and dad for taught me the value of education and tenacity to pursue my goals. Also, thanks to my brothers for always be my example. And thanks to my class friends, because we have been through all this adventure together for 2 years.

Finally, I want to express my gratitude to Tecnológico de Monterrey and CONACyT for the scholarships during my graduate studies.

¡Gracias a todos!

The Identification of DoS and DDoS Attacks to IoT Devices in Software Defined Networks by Using Machine Learning and Deep Learning Models

by

Josué Genaro Almaraz Rivera

Abstract

This thesis project explores and improves the current state of the art about detection techniques for Distributed Denial of Service (DDoS) attacks to Internet of Things (IoT) devices in Software Defined Networks (SDN), which as far as is known, is a big problem that network providers and data centers are still facing. Our planned solution for this problem started with the selection of strong Machine Learning (ML) and Deep Learning (DL) models from the current literature (such as Decision Trees and Recurrent Neural Networks), and their further evaluation under three feature sets from our balanced version of the Bot-IoT dataset, in order to evaluate the effects of different variables and avoid the dependencies produced by the Argus flow data generator.

With this evaluation we achieved an average accuracy greater than 99% for binary and multiclass classifications, leveraging the categories and subcategories present in the Bot-IoT dataset, for the detection and identification of DDoS attacks based on Transport (UDP, TCP) and Application layer (HTTP) protocols.

To extend the capacity of this Intrusion Detection System (IDS) we did a research stay in Colombia, with Universidad de Antioquia and in collaboration with Aligo (a cybersecurity company from Medellín). There, we created a new dataset based on real normal and attack traffic to physical IoT devices: the LATAM-DDoS-IoT dataset. We conducted binary and multiclass classifications with the DoS and the DDoS versions of this new dataset, getting an average accuracy of 99.967% and 98.872%, respectively. Then, we did two additional experiments combining our balanced version of the Bot-IoT dataset, applying transfer learning and a datasets concatenation, showing the differences between both domains and the generalization level we accomplished.

Finally, we deployed our extended IDS (as a functional app built in Java and connected to an own cloud-hosted Python REST API) into a real-time SDN simulated environment, based on the Open Network Operating System (ONOS) controller and Mininet. We got a best accuracy of 94.608%, where 100% of the flows identified as attackers were correctly classified, and 91.406% of the attack flows were detected. This app can be further enhanced with the creation of an Intrusion Prevention System (IPS) as mitigation management strategy to stop the identified attackers.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Distributed Denial of Service (DDoS) attacks are one of the main threats to network systems, affecting the applications and devices that rely on them. DDoS attacks consist in grouping multiple devices against one target, preventing legitimate users to access services such as email and websites, damaging the money and time of people and organizations. They represent the most common and critical attack against Internet of Things (IoT) devices, Cloud Computing, and fifth-generation (5G) communication networks [63].

According to the Kaspersky Q2 2020 DDoS Attacks Report [50], in the second quarter of 2020, the number of DDoS attacks slightly increased compared to the first quarter of the same year (from 302.08% to 316.67%), and more than three-fold compared with the data for the same period in 2019 (see Figure 1.1). In addition, the Kaspersky Q3 2021 DDoS Attacks Report [35], when compared to the same quarter of 2020, shows that the total number of DDoS attacks increased by nearly 24%, and the total number of advanced and targeted DDoS attacks increased by 31%.

A factor that intensifies these denial of service attacks, is that everyday more and more devices are connected to and through the Internet (a trend which does not seem to stop). From smart homes to industrial environments, the IoT is an ally to easing daily activities, where some of them are critical. By 2025, there will likely be more than 27 billion IoT connections [57], and due to the large amount of different manufacturers, it leads to the lack of robust security standards between them for the fabrication of these devices, that in turn allows for the creation of botnets after IoT hijacking [22]. Also, these IoT devices often use weak passwords which makes them more vulnerable to compromise and exploitation. This infection often goes unnoticed by the users, and an attacker could easily conduct a high-scale attack without the device owner's knowledge [22]. For instance, the Mirai botnet unleashed massive DDoS attacks on major websites from millions of compromised devices in 2016, showing the power of IoT attacks [46]. Another example is

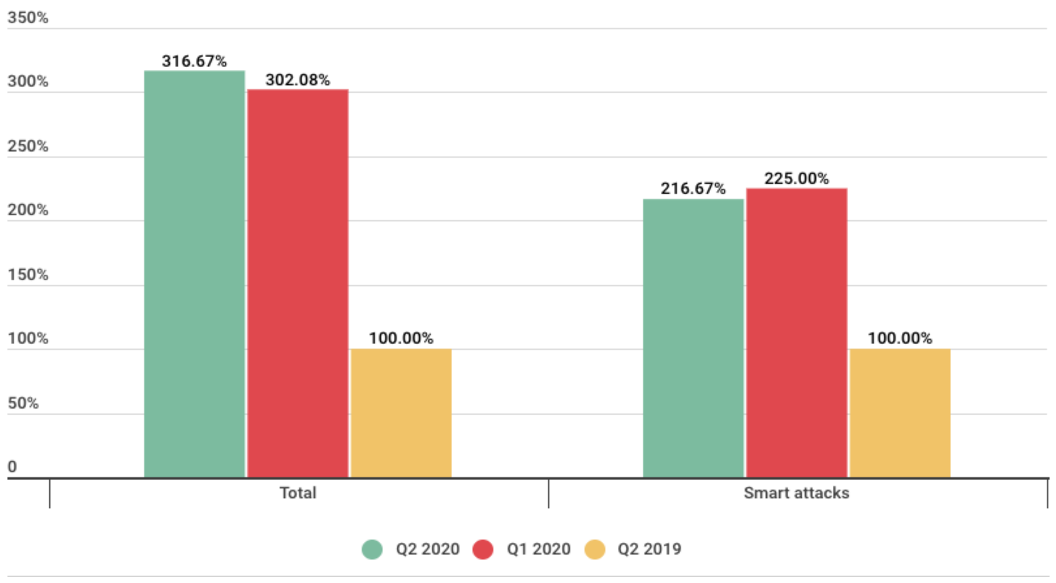


Figure 1.1: Quarter trends of the total number of DDoS attacks in 2020 compared to 2019. Image from [50].

Mozi, which emerged in 2019, and is the most active Mirai-type variant, controlling approximately 438,000 hosts which target routers and cameras, and in 2020 accounted for 89% of the total IoT attacks detected by IBM for the year [46]. These data corroborate the need for computer systems capable of protecting the IoT infrastructure.

There exists a vast variety of DDoS attacks, but also there are many existing defense systems, and these applications still create the impression that the research community efforts are not enough. However that is the reality, because cybersecurity, as many others fields, changes so fast and it is needed to accelerate it even further.

These defense systems are divided into two main categories:

- Intrusion Detection Systems (IDS).- These systems are responsible for capturing network anomalies [44], and detecting if an attack is being conducted or not.

- Intrusion Prevention Systems (IPS).- These systems are responsible for responding to the detected attacks.

There has been research and proposals using Artificial Intelligence (AI) to tackle these detection and mitigation problems, such as Support Vector Machines [47], which have reduced the effects of DDoS attacks, and an IPS designed for Software Defined Networks (SDN) [42] to secure the data plane (such as the WedgeTail system [56]).

This work proposes to apply Machine Learning and Deep Learning algorithms, over DDoS attacks in SDN, for the creation of an Intrusion Detection System with at least 95% of accuracy in a deployed network, and integrate it into an Open Network Operating System (ONOS) app, for its future extension with a mitigation strategy to stop the identified attackers. Machine Learning models such as Decision Trees and Random Forests, and Deep Learning models such as Recurrent Neural Networks and Multi-layer Perceptron, are assessed and discussed.

The design of a SDN infrastructure is the best option to deploy our IDS, because these new generation networks are important for data centers, 5G technology, the integration of IoT devices, and have demonstrated effectiveness against cyberattacks [63]. An explanation of what is SDN can be found in chapter 2.

The Figure 1.2 shows a modular SDN infrastructure proposed in [54] for attacks detection and mitigation. We will focus on the IDS module, and provide it as a Service API for the whole network. For development and testing purposes, we will run a simulated environment using Mininet [43] and the ONOS controller. ONOS controller is the leading open source SDN controller for building next-generation SDN solutions, and it gives real-time control for native SDN dataplane devices with OpenFlow [15, 45].

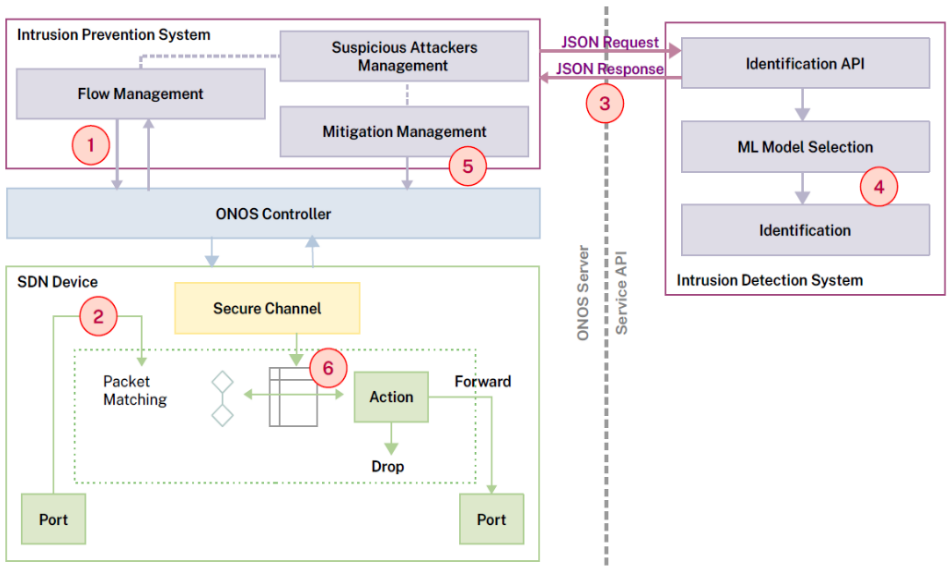


Figure 1.2: A modular SDN architecture proposal, from [54].

The expected behavior of the IDS block from the architectural design in Figure 1.2, is to receive a JSON request from the Flow Management module with the features from the flow, for further inspection using the Machine Learning (ML) or Deep Learning (DL) model specified as one of the object parameters. The IDS then classifies this flow as normal or attack, and identifies if this attack is based on UDP, TCP, or HTTP protocols. At the end of this whole process, a proper JSON response is sent to the IPS on the controller, for blacklisting of the attackers. The SDN Device block is usually a switch, and the devices connected to it and the controller are the interests of protection.

By applying this framework proposal, it is expected to achieve a solution to the rapidly growing DDoS attacks in Software Defined Networks, and significantly save time and money of people and organizations.

The collection of data used for training and testing the novel smart IDS here proposed, is the Bot-IoT dataset [41], published in 2019. This dataset was created with the simulation of IoT sensors emulating a smart home arrangement: a weather station, a smart fridge, motion activated lights, a remotely activated garage door, and a smart thermostat.

This dataset was chosen because it is a state-of-the-art dataset observed in other approaches for protecting IoT devices [36, 65, 37, 55, 29, 32], since it contains realistic normal and attack traffic. Furthermore, this dataset has a subcategory field which enables multiclass classification, and it presents information of DDoS and Denial of Service (DoS) attacks generated using GoldenEye [9] for Application layer protocols (HTTP), and Hping3 [10] for Transport layer protocols (UDP, TCP). The Argus tool [2] was used by the dataset authors, after collecting the pcap files, to generate the network flows and produce the features.

SYN flooding is one of the attack variants contained in the Bot-IoT dataset, and according to the Kaspersky Q3 2021 DDoS attacks report [23], it is the method used in 51.63% of the attacks. Flooding DDoS attacks based on UDP finished second with 38%, and those based on TCP remained third with 8.33%. DDoS attacks based on HTTP finished in fourth place with 1.02%. This distribution of denial of service attacks by type, motivates us to create models suitable for testing in Transport and Application layers, since they represent at least 47% of the whole attack distribution, and when accompanied with the SYN flooding variant from the Bot-IoT dataset we cover near to 99%.

The total amount of records in the Bot-IoT dataset exceeds 72 million [3], whilst 5% of this data (i.e., over 3 million records), is used in [41] by the authors of the dataset. However, the Bot-IoT dataset suffers from severe class imbalance [58], with just a few thousands (about 9,000) normal flows. This problem leads to bias towards a majority class, one of the issues this thesis tackles.

To extend the capacity of our proposed IDS and finally deploy it into a real-time simulated SDN environment, we did a research stay in Colombia with Universidad de Antioquia and in collaboration with Aligo (a cybersecurity company from Medellín) [1]. There, we created a new dataset based on real normal and attack traffic to physical IoT devices: the LATAM-DDoS-IoT dataset. More information about how we created this dataset, its structure and content, is shared in chapter 3.

1.1 Hypothesis and Research Questions

The hypothesis tested is that with a proper Machine Learning and Deep Learning models selection, class balancing technique, and feature set from the Bot-IoT dataset, we can obtain an Intrusion Detection System among the best ones in the state of the art literature (in terms of efficiency and accuracy), for detecting DDoS and DoS attacks to IoT devices in Software Defined Networks.

In particular, we will examine three main research questions:

1. Which class balancing technique can be applied to the Bot-IoT dataset for solving the bias it has towards the majority class?

2. What is the best feature set for class classification using the Bot-IoT dataset to avoid dependencies produced by the Argus flow data generator?

3. What Machine Learning and Deep Learning models selection is good for achieving fidelity and efficiency in detection of DDoS attacks to IoT devices in Software Defined Networks?

1.2 Objectives

The primary objective is the application of Machine Learning and Deep Learning models selection for the design of an Intrusion Detection System with an accuracy of at least 95%, for DDoS attacks to IoT devices in Software Defined Networks.

There are three secondary objectives in this study:

- Analyze the state of the art for a proper Machine Learning and Deep Learning models selection, for the detection of DDoS attacks to IoT devices in Software Defined Networks.

- Create an AI based Intrusion Detection System for the correct identification of DDoS attacks based on UDP, TCP, and HTTP protocols.

- Deploy the resulting Intrusion Detection System in a production simulated environment based on ONOS controller and Mininet, to identify denial of service attacks.

1.3 Contributions

We can summarize the main contributions of this thesis as follows:

- A suitable way to address the Bot-IoT dataset bias problem without adding class weights and without generating synthetic data.
- An extensive evaluation of the classification and time performance of several Machine Learning and Deep Learning models with different feature sets, which led to the discovery that it is not necessary to use the Argus flow data generator for any online implementation based on the Bot-IoT dataset.
- Anomaly detection models that match and exceed the performance of the current state of the art for identifying specific denial of service attacks categories, using different feature sets.
- A deployed IDS on a production simulated environment using ONOS and Mininet, that identifies denial of service attacks.
- A new dataset created with real normal and attack traffic to physical IoT devices.

1.4 Thesis Structure

This thesis consists of 5 chapters, including this introduction. The literature review is presented in chapter 2. Chapter 3 describes the solution proposal for the creation of the novel AI based Intrusion Detection System here proposed. Results analysis and evaluation are shown in chapter 4. Finally, in chapter 5, we present the conclusions and future work.

For the sake of clarity of the reading in chapters 3 and 4, we added appendices A and B, including some tables and figures.

Chapter 2

Literature Review

This chapter is about the efforts around the Bot-IoT dataset for the design of Intrusion Detection Systems based on Artificial Intelligence. We discuss about resampling techniques for tackling the class imbalance problem this dataset has, and different feature sets configurations proposed by other authors. First of all, we define what denial of service attacks are, their classification, and outline the different approaches that can be followed for attacks detection (to determine which fits better for IoT).

2.1 DDoS Attacks: Definition and Classification

Denial of service attacks from a single point (DoS) or distributed points (DDoS), lead to the depletion of memory and processing power on the victim (e.g., a server), so that legitimate traffic cannot be served, and the service becomes unresponsive [62]. They represent the most common and critical attack against IoT devices, Cloud Computing, and 5G communication networks [63], damaging the time and money of people and organizations. The DDoS field is too complex and has reached the point where it is difficult to understand the vast problem space [49].

According to the size of the traffic they generate for exhausting the victim, denial of service attacks can be classified into two main categories: high-rate and low-rate attacks. The behavior of low-rate attacks is extremely inconspicuous since they behave similarly to legitimate traffic and can account for about 10%-20% of the total normal network traffic [66]. Although the average traffic of low-rate attacks is small, they can potentially not only reduce the quality of service of the target, but also stop the service completely [66]. This is achieved by an attacker sending periodically pulsing data, instead of continuous flows [64]. Examples of low-rate denial of service attacks are GoldenEye, Slowloris and R.U.D.Y. (R-U-Dead-Yet?) [54].

On the other side, high-rate DDoS attacks employ an approach of high-rate transmission of packets, where the statistical changes in the behavior can be used to distinguish them from the normal data flows [64]. High-rate attacks violently exhaust the resources and the capacity of the network, making the victim unresponsive in a short period of time [66]. Examples of high-rate denial of service attacks are SYN Flood and UDP Flood [66].

2.2 Intrusion Detection Approaches: Network-based vs Host-based

Due to their nature of low power and low computational strength, IoT devices may not be capable of running encryption algorithms or antivirus software, and because of this, many IoT devices become easy targets of botnets. The large size of IoT makes these botnets be able to perform destructive DDoS attacks [48].

Because of these resource constraints, traditional DDoS defense solutions are not effective in IoT environments. It is mandatory to come up with new proposals, and there has been research about it. A couple examples are the FR-WARD system [48], and DeepPower [33].

In general, malware detection approaches can be divided into two categories [33]:

- Network-based approach: this type of solutions have been commonly used for protecting IoT systems. What these solutions do is to detect malicious traffic patterns in IoT networks.
- Host-based approach: this type of solutions examine the activities executed inside the IoT devices using for instance antivirus software.

There are many IoT devices, and so they are built in a different way, what makes it difficult to come up with one global detection approach. This is why it is impractical to directly use traditional host-based detection solutions.

Mitigation strategies are not in the scope of this thesis, but it is also an interesting and complex problem to mention. Large botnets made up of IoT devices have been a continuous presence in the threat landscape since 2016. Different IoT malware families, such as Mirai, have enabled attackers to launch massive DDoS attacks and other forms of cybercrime. A key aspect observed of IoT botnets, is that they have a short lifetime, so they seem to be highly disposable. This is a huge advantage for the attackers, because it means these botnets are very resistant to takedown with mitigation strategies such as blacklisting, because most IP addresses are used only once [59]. The detection and mitigation tasks for IoT networks are hard to design, but the benefits of these solutions are immense because of the demanded protection tasks.

As previously mentioned in the detection approaches, due to the resource constraints, and different manufacturers of IoT devices, we decided to follow a network-based approach for our detection system, using the Bot-IoT dataset. In the next section we see in detail the testbed used to design this dataset, along the tools used for collecting and parsing the packets into flows and create the features.

2.3 The Bot-IoT Dataset

The Bot-IoT dataset was created in a testbed designed by the University of New South Wales (see Figure 2.1), and includes among others, DDoS and DoS attacks. It consists of four Kali Virtual Machines which act as attackers (Bot Kali_1, Bot Kali_2, Bot Kali_3, and Bot Kali_4), and five IoT devices simulated using Node-RED [14], implemented on an Ubuntu Server, emulating a smart home scenario with a weather station, a smart fridge, motion activated lights, a garage door, and a smart thermostat. For generating normal traffic, the authors used the Ostinato tool [17], along with maintained periodically normal connections between the virtual machines by executing functions such as file transferring.

To dump and analyze the network traffic, the tshark tool [21] was used on the Ubuntu_Tap machine, and the pcap files collected were further parsed offline into network flows to produce the dataset features, using Argus. The total amount of records exceeds 72 million [3].

In the original publication of the Bot-IoT [41], its authors evaluated their work by training three different models: one Machine Learning model based on a Support Vector Machine (SVM) with a linear kernel; two Deep Learning models, using a simple Recurrent Neural Network (RNN), and a Long-Short Term Memory (LSTM) architecture.

The performance of the models was evaluated with two different sets of features: the first of them used the 10 best features (selected from a filter with Correlation Coefficient and Joint Entropy), and the second one used all the 35 features.

For multiclass classification of all the attacks in the Bot-IoT dataset, the best accuracy was of 99.988% with a SVM using all the 35 features. In terms of exclusively DDoS and DoS attacks, the work only reports binary classifications (e.g., Normal flows vs DDoS HTTP), getting the maximum accuracy of 99.999% for Normal flows vs DDoS UDP with a RNN.

Nevertheless, the dataset was unbalanced [58], which may have affected positively the identification of attacks (i.e., the majority class) due to data bias. This is one of the opportunities we address in this research. In the next section we explore the works around the Bot-IoT dataset.

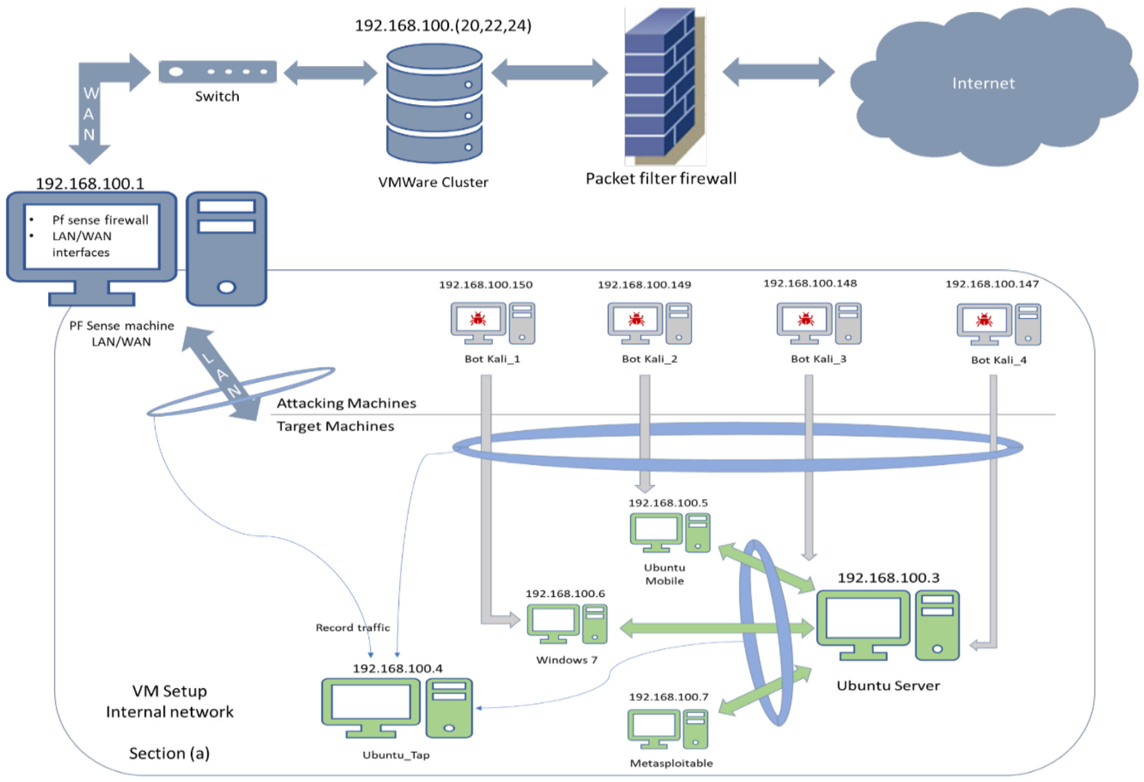


Figure 2.1: Testbed configuration for the Bot-IoT dataset. Image from [41].

2.4 Related Work

The work in [36], used also 5% of the Bot-IoT dataset, and presented 7 different Deep Learning models including RNNs, achieving a maximum sensitivity of 96.868% for Normal flows vs DoS HTTP. With respect to DDoS, the maximum sensitivity was for Normal flows vs DDoS UDP, with 96.666%.

In [65], the authors applied different Random Forest configurations, tuning the depth and the number of trees. The authors proposed 6 different feature sets (from 4 to 8 features, such as IP, port, and timestamp), and compared their accuracies with the 10 best features set and the SVM in the Bot-IoT paper [41]. The accuracy of the SVM with the 10 best features set is 88.372%, while the accuracy of the proposed Random Forest (with the 6 different feature sets) is 100%. Nevertheless, it is important to note that the experiments in [65] not only considered either DDoS or DoS attacks, but also included other types of attacks, such as data ex-filtration and service scanning. No other models were presented by the authors, only Random Forest, with small number of features, which might lead to the loose of information. With respect to time performance, the authors only evaluated the effects of the Random Forest sizes on run-time overheads to classify a single data packet.

In [37], a packet level model based on Deep Learning was proposed, using Feed Forward Neural Networks (FFNN) for binary and multiclass classification with the Bot-IoT dataset. The four categories of attacks were DoS, DDoS, reconnaissance and information theft, in order to differentiate them from the normal traffic. Confusion matrices were generated, and the accuracy, precision, recall, and F1 score metrics were used for performance evaluation. With respect to only DoS and DDoS attacks, the proposed model presented all accuracies above 99% in binary classification (e.g., Normal flows vs DDoS TCP), and an accuracy of 99.414% for multiclass classification. In order to deal with the unbalanced nature of the dataset, class weights were introduced to the training data, so the class with a smaller number of samples got a higher weight value. However, this technique could introduce the risk of over tuning, resulting in weights that may not generalize optimally [37].

In [55], the Bot-IoT dataset was used to validate a new feature selection algorithm based on the Area Under the Curve (AUC) metric. A feature set of 5 variables was selected as the best one, and the mean and the standard deviation of the duration of the aggregated records were two of those features. Only 4 Machine Learning models were applied: Decision Tree, Naive Bayes, Random Forest and SVM. The accuracy, precision, recall, and specificity metrics were used for performance evaluation. In terms of results, Random Forest and Decision Tree showed an accuracy of 100% for HTTP, TCP, and UDP denial of service attacks detection. This paper presented a solution for the problem of selecting effective features for accurate attacks detection in IoT networks. The AUC metric is useful for dealing with imbalanced datasets [38], nevertheless, the research work neither evaluates Deep Learning models nor presents a performance evaluation metric, such as the average of flows per second each model can process, which is relevant to evaluate the feasibility of real-time implementation of their proposed best models.

The work in [29] presented a novel use of Gated Recurrent Units (GRU) in the Bot-IoT dataset. GRUs aim to solve the vanishing gradient problem in a standard RNN [29], by using update and reset gates. The proposed model used only 125,971 samples from the original Bot-IoT dataset, in order to do a fair comparison and to have the same size than the NSL-KDD dataset [60], obtaining an accuracy of 99.76% for Normal vs Attack traffic identification, with no exclusivity for either DDoS or DoS attacks.

In [32], the Bot-IoT dataset was used for conducting binary and multiclass classification tasks, with balanced and unbalanced representations of it, where the class balancing technique used was based on weights, as seen in [37]. As mentioned by the authors in [32], they used the default values of the hyper parameters for each classifier, as provided by Scikit-Learn [53] and Keras [31]. In terms of performance metrics, they present indicators like accuracy and F1 score, however the authors did not present an evaluation of the models feasibility in a real-time scenario (e.g., by evaluating time performance). From the original Bot-IoT dataset that has 35 variables, the authors removed columns with missing values, as well as columns that contain text and columns they considered to be irrelevant. Their complete dataset had 19 variables, where features like the timestamps and the Argus sequence number remained. For training and testing, they applied a data split of 80% and 20% respectively, with no percentage reported for a validation set. For the weighted datasets, the Artificial Neural Network (ANN) was the most outstanding model, with a stable accuracy of 99% for binary classification in DDoS and DoS attacks protocols. For the multiclass classification, the authors presented an overall accuracy with all the attacks types contained in the Bot-IoT dataset, where the ANN kept in first place with an accuracy of 97%. The authors stated they did not train Deep Learning models.

In [58], the authors recognized the need for class balancing in the Bot-IoT dataset. This study showed the majority classes belong to the attack types, while the normal traffic is part of the minority classes with only 9,515 samples (accompanied with information theft, which has 1,587 samples), resulting in a ratio of normal to malicious traffic of 1:7687 [58]. An imbalanced dataset may lead to problems such as poor accuracy and/or bias towards the majority class in the results obtained. Specifically, talking about DDoS and DoS attacks, the normal to attack traffic ratio for DoS is 1:459 (i.e., 9,515 to 33,005,194 flows), and the ratio for normal to DDoS is 1:4038 (i.e., 9,515 to 38,532,480 flows) [58]. Thus, the Bot-IoT dataset seems to be better suited to distinguish between a DoS and a DDoS attack [58], since these categories have similar number of samples (i.e., about 38 million for DDoS, and 33 million for DoS).

Based on this state of the art review, it can be seen that only two works applied class balancing to the Bot-IoT dataset, and not all of them considered Machine Learning and Deep Learning models. Furthermore, time performance evaluation only was contemplated by one of the works. This analysis reflects a need for a more extensive experimentation on the Bot-IoT dataset to design robust anomaly detection models.

Table 2.1 presents a summary comparing the related work with our approach. We present this comparison across 6 relevant criteria to describe the position of our work and how it stands out from the current state of the art. As we can see, our work is the only one that tackles the class balancing problem of the Bot-IoT dataset whilst using different feature sets, evaluating ML and DL models, time performance, and all at a flow-level detection.

Table 2.1: Comparison between our approach and the related work around the Bot-IoT dataset.

	This work	**[41]**	**[36]**	**[65]**	**[37]**	**[55]**	**[29]**	**[32]**
Class balancing	✔	✗	✗	✗	✔	✗	✗	✔
ML models evaluation	✔	✔	✔	✔	✔	✔	✗	✔
DL models evaluation	✔	✔	✔	✗	✔	✗	✔	✗
Feature set(s) proposal	✔	✔	✗	✔	✔	✔	✗	✔
Time performance evaluation	✔	✗	✗	✔	✗	✗	✗	✗
Flow-level detection	✔	✔	✔	✔	✗	✔	✔	✔

2.5 Class Balancing

In order to deal with imbalanced datasets, resampling techniques can be applied to ameliorate this problem. When oversampling, minority class instances are created, either by duplicating elements or by creating new ones synthetically from a similar distribution. The latter technique can be achieved using the Synthetic Minority Oversampling Technique (SMOTE), where depending on the amount of oversampling required, neighbors from the k nearest neighbors are randomly chosen, with one sample generated in each one's direction [30]. When undersampling, samples from the majority class are removed, which can cause loss of information.

We propose to tackle the data bias problem of the Bot-IoT dataset by randomly selecting consecutive flows sections per each DDoS/DoS attack type, to preserve the temporal behavior of the attacks whilst not altering the network traffic collected from the realistic testbed configuration used to design the Bot-IoT dataset.

2.6 Software Defined Networks

The Software Defined Networks are also known as new generation networks. This networking paradigm enables more flexible and manageable environments, by decoupling the control and data planes, which were formerly implemented inside switches and routers [62]. All the network control functions, such as traffic monitoring, take place in a software-based controller [63], which can either be physically centralized or distributed, but logically centralized [26], whilst network switches become forwarding elements that follow rules to dispatch flows [62].

The controller is the most important entity in SDN, however, it is also very sensitive to a broad range of attacks [62]. DDoS attacks are one of these security risks, downgrading the performance, availability, and integrity of the network, by saturating with packets the communication channel between the switches and the controller [62]. This communication channel is also known as the Southbound interface, where OpenFlow [45] and P4 [18] are well-known providers for such API.

In order to fulfill the principles of better administration, transparency, and automation, the Open Networking Foundation [16] proposed a three-layer architecture for SDN (see Figure 2.2), with the control and data planes, but also with the application plane. The application plane includes applications for providing services such as monitoring and routing, such as Intrusion Detection Systems and Intrusion Prevention Systems. These applications communicate to the controller using a Northbound interface, which can be implemented as a REST API [62].

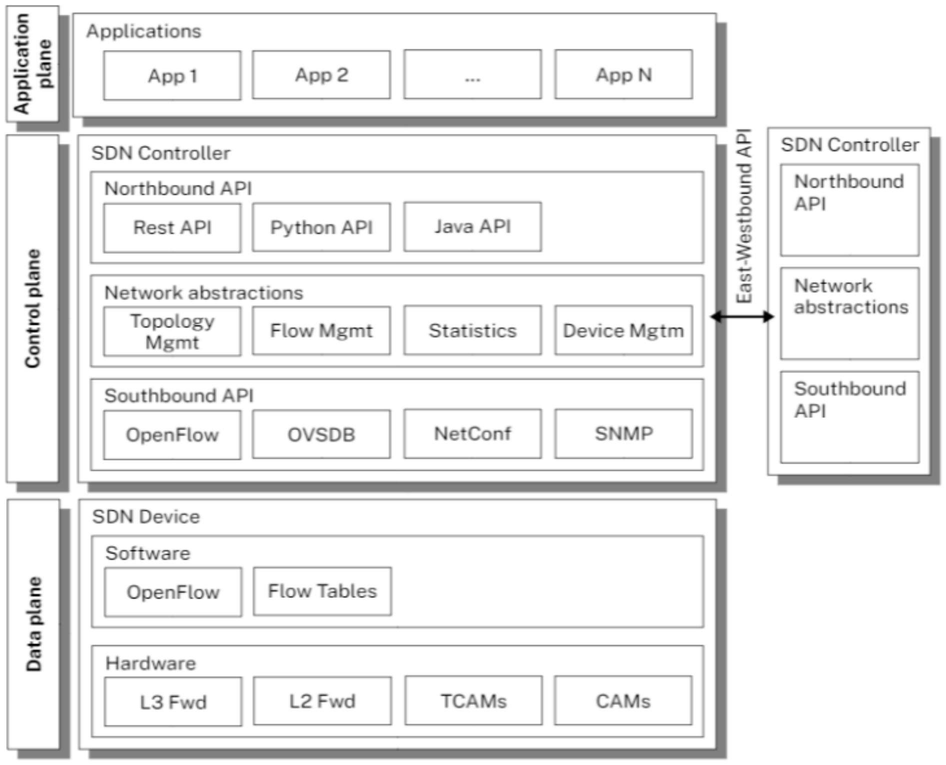


Figure 2.2: Three-layer architecture for SDN, from [62].

This flexibility on global network monitoring and network configuration enables the implementation of detection and mitigation mechanisms against cyberattacks [63]. Simulation of SDN can be done even on a personal computer, using Mininet [43] and controllers such as ONOS [27] and Ryu [19], for research and testing purposes of large-scale networks.

2.7 Conclusions

Based on this review of the current state of the art, we planned to carry out an extensive experimentation specialized in normal flows vs DDoS and DoS attacks, in binary and multiclass classifications, with different feature sets to evaluate how different flows processors could be used either in a simulated or in a real network implementation, such as CICFlowMeter [4] or Flowtbag [8], to avoid dependencies produced by Argus. This experiment design divided in binary and multiclass classifications, allowed us to evaluate the detection and identification of network traffic, respectively, leveraging the categories and subcategories present in the Bot-IoT dataset. Likewise, we planned to compare several ML and DL models for a finer selection of our best algorithms for anomaly detection, under popular metrics such as accuracy and precision, but also time performance to analyze the feasibility of implementation of our smart IDS in a real-time environment.

We decided to train and test Support Vector Machines, Decision Trees, and Random Forests, as Machine Learning models, and Multi-layer Perceptron, Recurrent Neural Networks, LSTMs, and GRUs, as Deep Learning models, due to their strong use in the state of the art for smart intrusion detection.

Chapter 3

Solution Proposal

In this chapter, we describe the methodology we followed to create our balanced version of the Bot-IoT dataset, and also the feature standardization process we applied to help convergence in the different classifiers. Likewise, we present a summary of the ML and DL models parameters to conduct our experiments with the proposed feature sets, and define the different performance metrics to evaluate our results.

First of all, we present the data processing we followed and our proposed feature sets for the Bot-IoT dataset, to then continue with the AI models training and evaluation definition. Then, we present and explain the LATAM-DDoS-IoT dataset creation, which was used for extending our smart IDS based on the Bot-IoT dataset for the goal of deploying our prototype in a real-time SDN testbed. This experimental topology is described in the last section of this chapter.

For the sake of clarity of the reading in this chapter, some tables are in appendix A.

3.1 Data Processing and Proposed Feature Sets

In order to start working with the Bot-IoT dataset, we downloaded the labeled CSV files from [3]. A total of 9,085 samples were extracted for the normal class. We selected items in the majority class (the attacks) by randomly choosing sections of consecutive flows for each DDoS/DoS attack type, in the same proportion to the normal samples to keep a balanced ratio. Figure 3.1, shows we achieved the same number of flows for each of the classes for the multiclass classification, where UDP, TCP, and HTTP, are samples from both DDoS and DoS attacks. See Figure 3.2 for the distribution for binary classification. In the end, the complete dataset size was of 36,340 samples.

In order to design our models, we selected three different feature sets from the original Bot-IoT dataset that has 35 variables. We followed this approach in order to evaluate

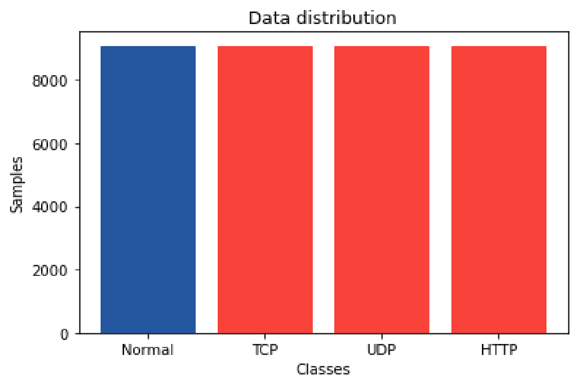


Figure 3.1: Data distribution plot for multiclass classification.

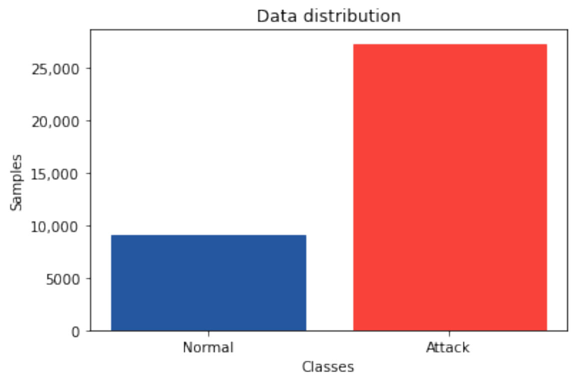


Figure 3.2: Data distribution plot for binary classification.

how the records timestamps affect in the models predictions, and to avoid the dependencies produced by the Argus flow data generator (so that more flows processors could be used in a simulated or real network implementation, such as CICFlowMeter or Flowtbag).

As seen in Table 3.1, all the feature sets share the same statistical variables (i.e., rates, mean, maximum, minimum, etc.). The first feature set we tried is selected to evaluate the impact of the timestamps and the Argus sequence number on the classification results. The second feature set removes the timestamps, because we argue the model could memorize these features which may lead to poor generalization in a real-time scenario, likewise we remove the Argus sequence number to avoid dependencies with this parser. Finally, in the third feature set, we keep the Argus sequence number in agreement with the current state of the art (that use this feature) to evaluate how it affects the classifications excluding only the timestamps.

Table 3.1: Feature sets selected.

Name	**Features**	**Description**
First feature set	stime, pkts, bytes, ltime, seq, dur, mean, stddev, sum, min, max, spkts, dpkts, sbytes, dbytes, rate, srate, drate	Using timestamps, the Argus sequence number, and the statistical variables (i.e., rates, mean, maximum, minimum, etc.).
Second feature set	pkts, bytes, dur, mean, stddev, sum, min, max, spkts, dpkts, sbytes, dbytes, rate, srate, drate	With no timestamps neither the Argus sequence number, only the statistical variables.
Third feature set	pkts, bytes, seq, dur, mean, stddev, sum, min, max, spkts, dpkts, sbytes, dbytes, rate, srate, drate	With the Argus sequence number and the statistical variables.

The three feature sets ranged between 15 and 18 variables, which were selected after dropping columns with missing values and choosing statistical features to capture the traffic behavior. No more feature removal was applied in order to capture the most amount possible of information. See Table 3.2 for the description of the variables. It is relevant to note that 8 of the variables in the 10-best feature set, identified in [41], are included in the first and the third feature sets we proposed, and 7 of those 10-best variables in the second feature set.

Table 3.2: Variables description.

Feature	**Description**
stime	Record start time.
ltime	Record last time.
seq	Argus sequence number.
pkts	Total number of packets in transaction.
bytes	Total number of bytes in transaction.
dur	Record total duration.
mean	Average duration at records aggregate level.
stddev	Standard deviation of the duration at records aggregate level.
sum	Total duration at records aggregate level.
min	Minimum duration at records aggregate level.
max	Maximum duration at records aggregate level.
spkts	Source to destination packets count.
dpkts	Destination to source packets count.
sbytes	Source to destination bytes count.
dbytes	Destination to source bytes count.
rate	Total packets per second in transaction.
srate	Source to destination packets per second.
drate	Destination to source packets per second.

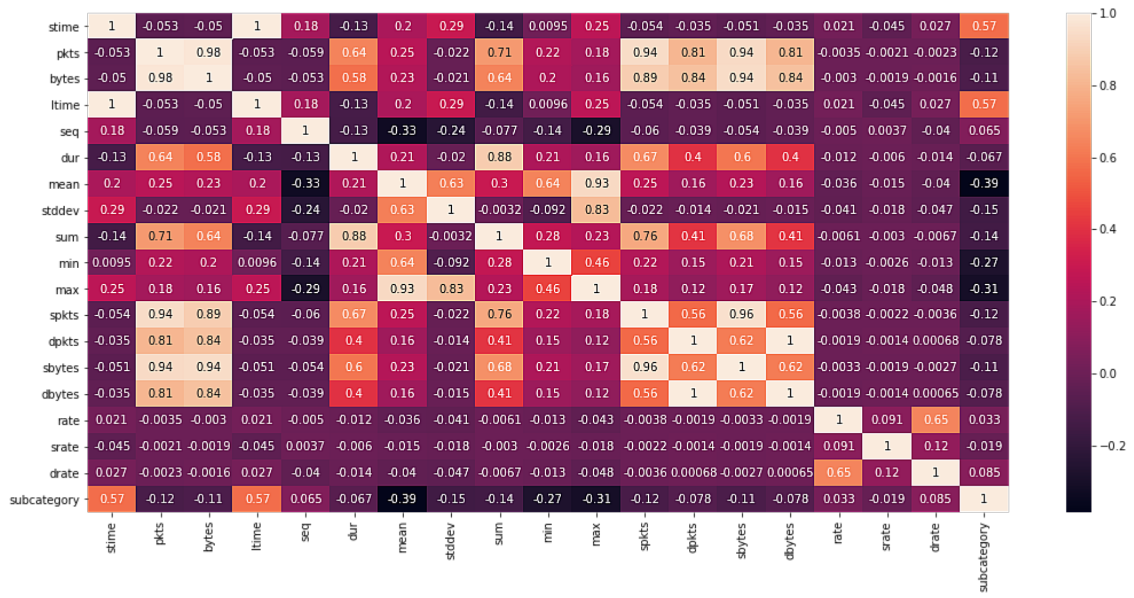


Figure 3.3: Correlation matrix for multiclass classification.

The correlation matrix for the multiclass classification task is shown in Figure 3.3, and for binary classification in Figure 3.4. Here, subcategory represents the class to predict. Since we wanted to see the linear relation between our variables (where all are numerical), we calculated these matrices using the Pearson's correlation coefficient, resulting in values between -1 and 1, positive values indicating a pair of features that increase or decrease together, and negative values indicating that the increase of one variable implies the decrease of another variable (and vice versa).

To help convergence, the features were standardized by subtracting the mean (centering), and dividing by the standard deviation (scaling), resulting in a set of values whose mean is 0, and the standard deviation equals to 1 (see Equation (3.1)).

$$x' = \frac{x - mean}{stddev} \tag{3.1}$$

3.2 AI Models Training and Evaluation

The dataset split for all the Machine Learning and Deep Learning models was 80% for training, 10% for validation (tuning hyper parameters), and 10% for testing. Given our total number of samples, we decided to create separate sets for training, validation and testing, instead of using other alternatives such as k-fold cross-validation.

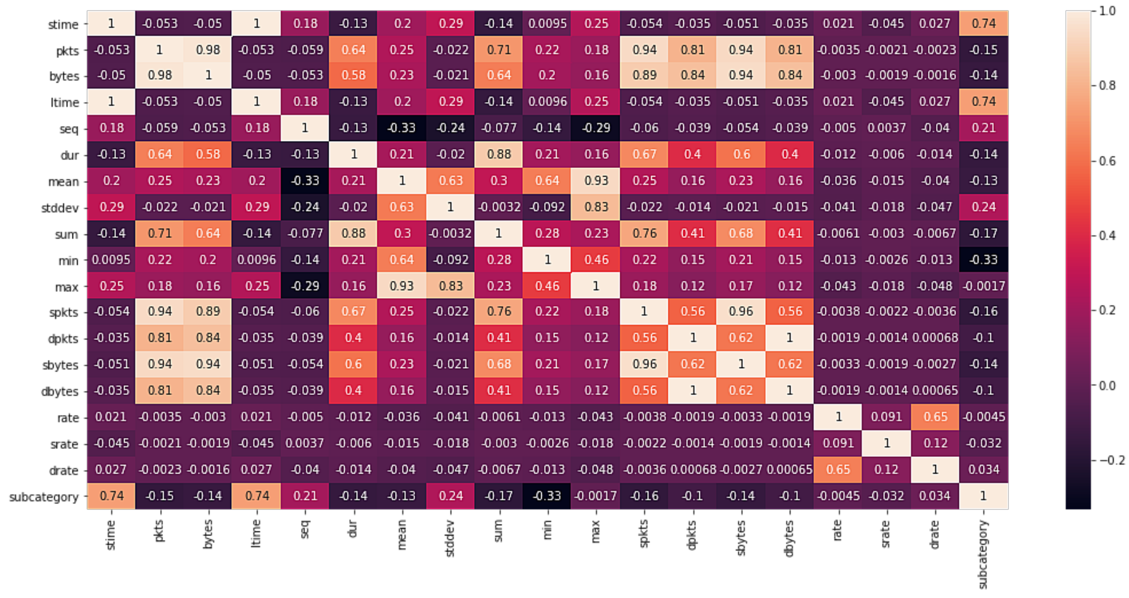


Figure 3.4: Correlation matrix for binary classification.

All the ML models (i.e., Support Vector Machines, Decision Trees, and Random Forests) were built using Scikit-Learn [53], and the DL models (i.e., RNN, GRU, LSTM, and Multi-layer Perceptron [MLP]) using PyTorch [52]. Confusion matrices were generated and accuracy, precision, recall, and F1 score metrics, in addition to time performance as proposed in [25], were used for evaluation and models benchmark. See Equations (3.2), (3.3), (3.4), and (3.5), for these metrics' definitions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.5}$$

Tables A.1, A.2, and A.3, show the parameters for the ML models using the three feature sets. With respect to the DL models, all of them share the characteristics presented in Table A.4, where the input size varies according to the feature set (i.e., 18 for the first set, 15 for the second one, and 16 for the third one). The hyper parameters for both Machine Learning and Deep Learning were chosen after a process of systematic tuning.

This process of systematic tuning for the hyper parameters, consisted in an iterative approach of proving different settings. For instance, when defining the depth of our best decision tree, we selected the tree depth with the best accuracy from a finite range of values; for the SVM we tried the different kernels (linear, polynomial, radial, and sigmoid) to select the best one; for the DL models we defined the best learning rate minimizing the range between 0 and 1 using random decimal numbers.

In this regard, the best obtained max depth for the Decision Tree, in both binary and multiclass classification, was correspondingly used as max depth for the Random Forest sub-estimators. Likewise, we report the number of trees that led us to optimal balance between accuracy and run-time. It should be noted that, the Decision Tree implementation Scikit-Learn uses is an optimized version of the CART (Classification and Regression Trees) algorithm [5].

All our DL models follow a fully connected neural network architecture, where Adam [40] is used as optimizer. Adam is suitable for convex and non-convex optimization problems, with large datasets and a lot of parameters. We decided to choose it because when compared to other optimizers, such as AdaGrad [34] and RMSProp [61], it converges faster (in a less number of epochs) to the best solution. Advantages of Adam are the natural implementation of learning rate annealing (i.e., adaptive learning rates), and its hyper parameters require little tuning (in fact, we used the default values provided by PyTorch for β_1, β_2, and ϵ).

3.3 The LATAM-DDoS-IoT Dataset

For extending our smart IDS based on the Bot-IoT dataset, we did a research stay in Colombia, with Universidad de Antioquia and in collaboration with the cybersecurity company Aligo, where we built and implemented a hybrid testbed for DDoS attacks. See Figure 3.5 for the testbed configuration.

The victims of the attacks were four physical IoT devices and one simulated device. The four physical IoT devices were two Google Home Mini, one smart power strip, and one smart light bulb, connected via an access point. The simulated device using Node-RED was a thermostat, running on a container in a virtual machine with Fedora Linux.

We launched DDoS and DoS attacks based on UDP and TCP (using Hping3), and based on HTTP (using GoldenEye). The attackers were two Kali virtual machines, running the previously mentioned tools, but also tcpdump [20] for capturing the network traffic, and Nmap [13] for port scanning to identify open services running on the victims.

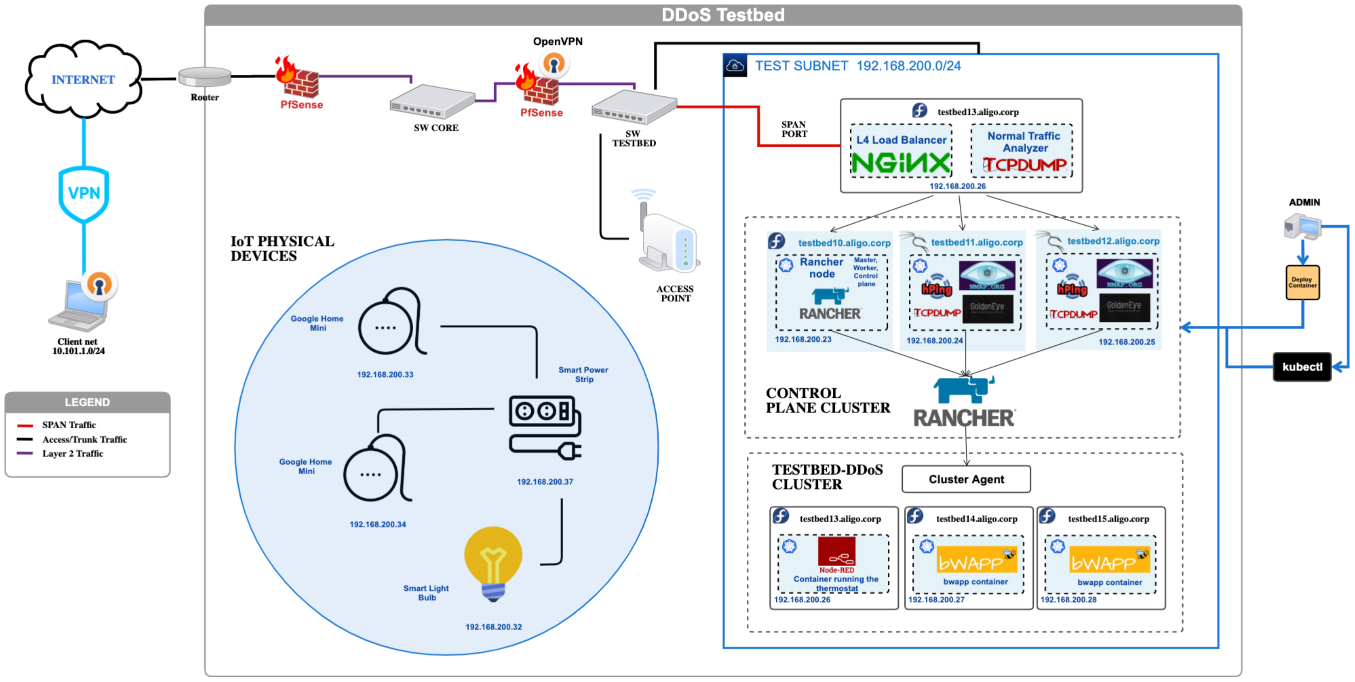


Figure 3.5: Testbed configuration for the LATAM-DDoS-IoT dataset.

The normal traffic was captured from the *testbed13.aligo.corp* node, connected to a span port, where allowed real activities from the Aligo customers were collected. The time window for capturing this normal traffic was of 50 minutes, whilst for each victim attack was of 10 minutes. Each time window was run after the previous finished, to avoid any bias towards the timestamps. The pcap files collected were then offline processed using Argus for getting data at flow-level and the features we previously discussed from our different feature sets. See Table 3.3 for the total number of flows collected for the DoS version of our dataset, and Table 3.4 for the DDoS version.

The data was labeled following the same convention from the Bot-IoT dataset, with a category column (where 0 means normal, and 1 means attack traffic), and a subcategory column (with values from 0 to 3, for Normal, UDP, TCP, and HTTP classes, respectively). This labeling allows the implementation of supervised learning methods (as we have done so far), and binary and multiclass classification tasks for the detection and identification of denial of service attacks.

The total number of samples for the LATAM-DoS-IoT dataset was 30,662,911 flows with 20 columns, and for the LATAM-DDoS-IoT dataset was 49,666,991 flows with the same number of columns. The correlation matrix for the DoS version is shown in Figure

Table 3.3: Total flows for the LATAM-DoS-IoT dataset.

Victim	Via	Time	Flows
Google Home Mini (192.168.200.33)	hping3 TCP	10 minutes	8,002,589
Google Home Mini (192.168.200.34)	hping3 UDP	10 minutes	7,667,921
Smart Light Bulb	hping3 UDP	10 minutes	7,610,420
Smart Power Strip	hping3 UDP	10 minutes	6,573,253
Normal Traffic	Span port	50 minutes	799,187
Thermostat	GoldenEye (HTTP)	10 minutes	9,541

Table 3.4: Total flows for the LATAM-DDoS-IoT dataset.

Victim	Via	Time	Flows
Google Home Mini (192.168.200.34)	hping3 UDP	10 minutes	14,804,835
Smart Light Bulb	hping3 UDP	10 minutes	13,053,593
Smart Power Strip	hping3 UDP	10 minutes	12,632,022
Google Home Mini (192.168.200.33)	hping3 TCP	10 minutes	8,343,462
Normal Traffic	Span port	50 minutes	799,187
Thermostat	GoldenEye (HTTP)	10 minutes	33,892

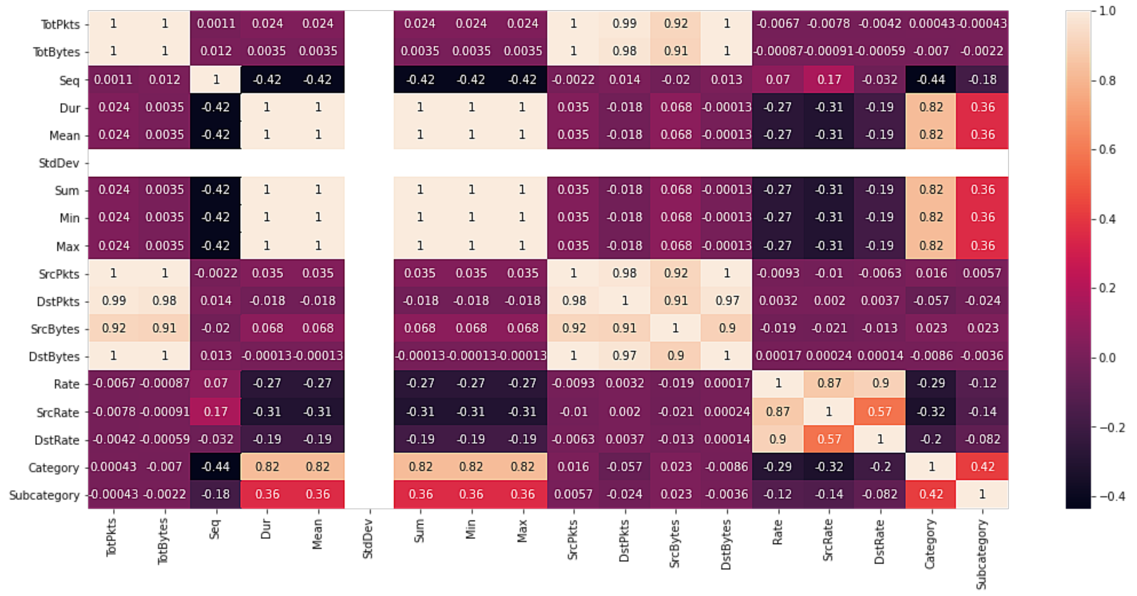


Figure 3.6: Correlation matrix for the LATAM-DoS-IoT dataset.

3.6, and for the DDoS version in Figure 3.7 (both calculated using the Pearson's correlation coefficient).

As stated by the Bot-IoT dataset authors [41], they did not have access to real IoT devices during their experiments. This is one of the advantages of using our proposed dataset, where our scenario is diverse and represents a realistic configuration. Also, another advantage is that we present a vast amount of real normal traffic, generated by real customers consuming real services, which can be used to model one-class classifiers [39] for zero-day attacks [28] detection.

Our denial of service attacks based on TCP and UDP protocols had a flooding behavior. We successfully made the two Google Home Mini fail, where the Google service remained processing the voice requests (with the status lights on) but it could not respond to them. Also, we got to successfully stress the smart power strip, because it did not resist the DDoS attack for more than 8 minutes. For the smart light bulb, the light remained on, but it could not change of color. Hping3 seems to be a strong attack tool for denial of service, since GoldenEye (used against the simulated thermostat) did not interrupt the victim. That can be explained because GoldenEye is a low-rate denial of service tool so it requires more time to stop the victim. See Table 3.5 for the commands used to launch the attacks. The difference between the LATAM-DoS-IoT and LATAM-DDoS-IoT datasets, is that the same attack commands were launched from either one or two Kali virtual machines at the

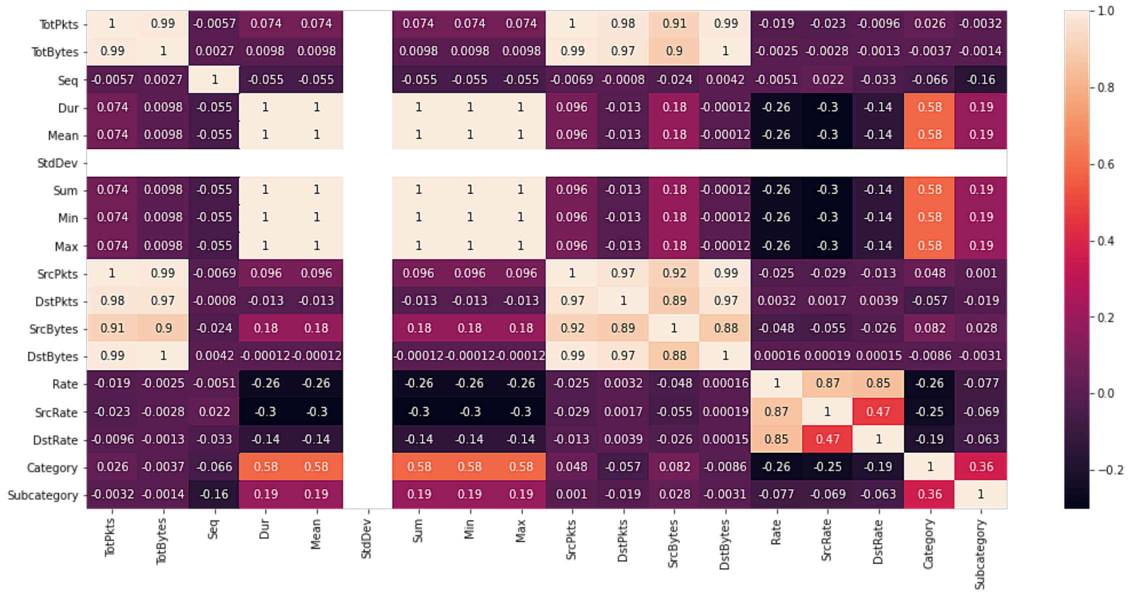


Figure 3.7: Correlation matrix for the LATAM-DDoS-IoT dataset.

same time, respectively.

Table 3.5: Attacks commands for both LATAM-DoS-IoT and LATAM-DDoS-IoT datasets.

Victim	IP	Port	Command
Smart Power Strip	192.168.200.37	6668	*hping3 -S –udp –flood -d 100 -p 6668 192.168.200.37*
Google Home Mini	192.168.200.34	8008	*hping3 -S –udp –flood -d 100 -p 8008 192.168.200.34*
Google Home Mini	192.168.200.33	8008	*hping3 -S –flood -d 100 -p 8008 192.168.200.33*
Smart Light Bulb	192.168.200.32	6668	*hping3 -S –udp –flood -d 100 -p 6668 192.168.200.32*
Thermostat	192.168.200.26	80	*goldeneye.py http://192.168.200.26:80 -m post -s 75 -w 1*

From Table 3.5 we can see three distinct commands patterns:

1. *hping3 -S –udp –flood -d 100 -p 6668 192.168.200.37*: this command means to launch a SYN UDP attack sending packets as fast as possible, of body size 100 bytes, against destination port 6668 of the IP 192.168.200.37.

2. *hping3 -S –flood -d 100 -p 8008 192.168.200.33*: this command means to launch a SYN TCP attack sending packets as fast as possible, of body size 100 bytes, against destination port 8008 of the IP 192.168.200.33.

3. *goldeneye.py http://192.168.200.26:80 -m post -s 75 -w 1*: this command means to launch an HTTP attack against the port 80 of the IP address http://192.168.200.26, using the HTTP method POST, 75 concurrent sockets and 1 concurrent worker.

From the datasets we created, we decided to apply the same random selection of consecutive flows sections for each attack type, in the same proportion to the normal samples to keep a balanced ratio. See Figure 3.8 for the data distribution for the LATAM-DoS-IoT dataset, and Figure 3.9 for the LATAM-DDoS-IoT dataset. In the end, the complete dataset size for training the AI models was of 2,407,102 and 2,431,453 samples, for DoS and DDoS versions respectively. The difference between both distributions is basically the number of GoldenEye flows, which is less than the other classes due to the low-rate behavior of this tool.

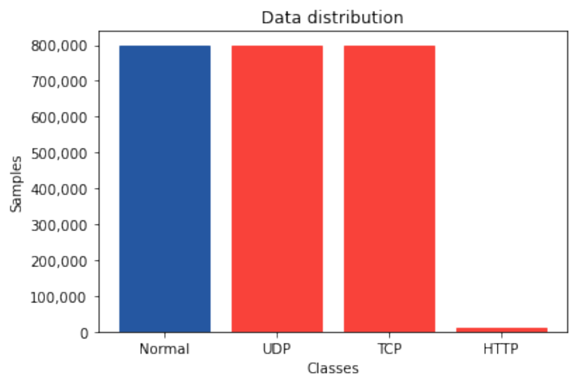


Figure 3.8: Data distribution plot for the LATAM-DoS-IoT dataset for training the AI models.

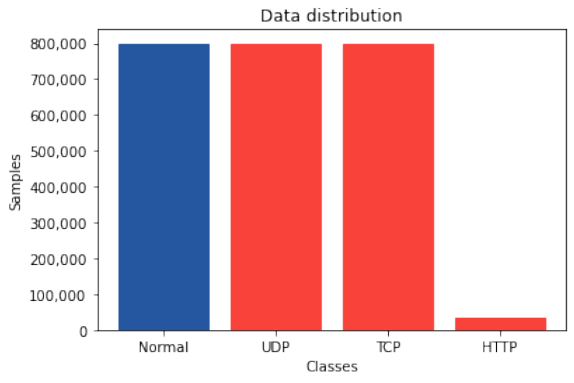


Figure 3.9: Data distribution plot for the LATAM-DDoS-IoT dataset for training the AI models.

Once having these balanced versions, we conducted different experiments with the best Machine Learning and the best Deep Learning models from our tests with the Bot-IoT dataset, namely Decision Tree and Multi-layer Perceptron. These new experiments included applying transfer learning with our Bot-IoT dataset balanced version, and another experiment concatenating datasets. In chapter 4 we present and discuss the obtained results.

3.4 SDN Testbed for IDS Deployment

In order to deploy our smart IDS we simulated a SDN infrastructure using Mininet and ONOS as network controller. This testbed is a hybrid architecture, where we have ONOS installed and running as a Linux service in an Ubuntu 18.04.6 LTS machine, but we also have simulated devices using Mininet. The network diagram is depicted in Figure 3.10.

Our testbed includes one physical Ubuntu computer where the ONOS controller is directly running as a Linux service. Also, the architecture is structured with two switches, where each one of them has four nodes with an IP address in the subnet 10.0.0.x. We used iPerf [12] for generating normal traffic from the normal clients, and also for implementing two servers listening on TCP port 5001. The attacker nodes generate high-rate denial of service attacks based on TCP using Hping3. We can list the different commands, per type of node from the Figure 3.10, as follows:

1. *iperf -s*: this command allows running iPerf in server mode. We ran this command for implementing the server nodes (i.e., the 10.0.0.1 and 10.0.0.8 IP addresses).

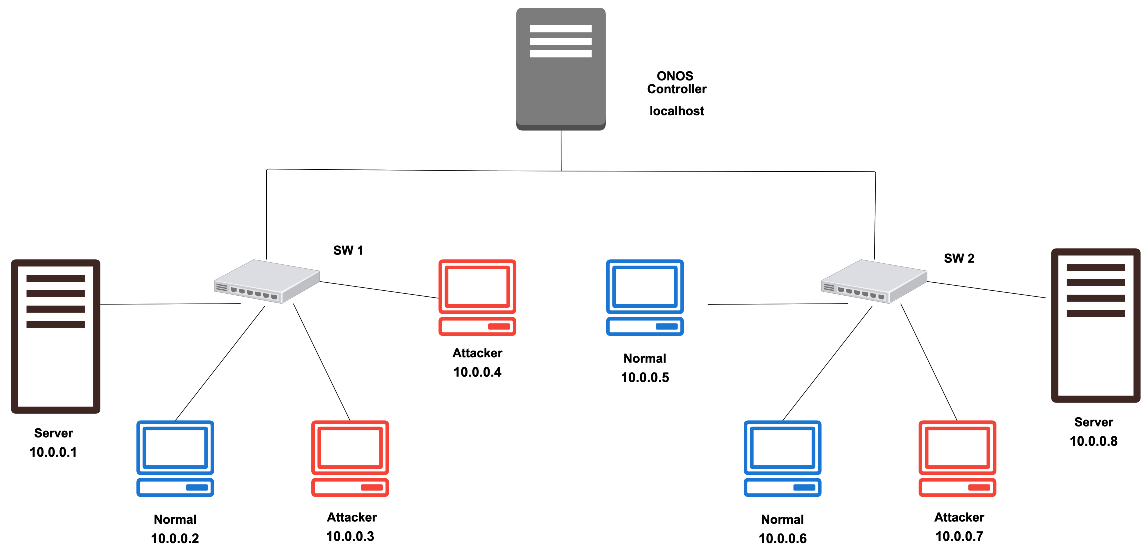


Figure 3.10: SDN testbed for deploying our smart IDS.

2. *iperf -c 10.0.0.1 -p 5001 -t 60 -p 3*: this command allows running iPerf in client mode, and creates three TCP simultaneous connections to the server 10.0.0.1 in the port 5001, during 60 seconds.

3. *hping3 -S –faster -d 100 -p 5001 10.0.0.1*: this command means to launch a SYN TCP attack sending 100 packets per second, of body size 100 bytes, against destination port 5001 of the IP 10.0.0.1.

Naming the hosts from the IP address 10.0.0.1 up to the IP address 10.0.0.8, as h1 up to h8, we can explain from which node to what node the communications occur in the SDN testbed. The host h2 generates legitimate traffic to h8; h5 and h6 legitimate traffic to h1; h3 and h4 attack h1; and h7 attacks h8. We generated legitimate traffic during 60 seconds from our normal hosts at the same time, and in the second 50 we started launching denial of service attacks from all the attacker nodes for 10 seconds. This was enough for running a ping reachability test in the network and see 98% of the communications dropped.

Currently our Java prototype is limited to work only for TCP traffic, but can be further enhanced to capture also UDP traffic, in order to exploit the most out of our AI models. This is the reason why we only launch TCP traffic from the attackers. HTTP also runs over TCP, but due to time constraints and since it only represents 1.02% of the attacks registered in the third quarter of 2021 (as mentioned in chapter 1), we decided to leave that experiment as future work.

The Java app mentioned above is based on [11], and was developed for capturing the network traffic, parse the packets into flows, and then extract the features. It uses a Java implementation of Flowtbag (originally written in Go), and we extended the code for connecting to our cloud-hosted REST API written in Flask [7] (where our logic for the flows classification is running), and to structure the JSON request with the variables we needed. See Algorithm 1 for the procedure we followed for the IDS classification process inside our API.

Algorithm 1 IDS Classification Process

```
Input: JSON request with the flow parameters.
Output: JSON response with the flow classification.
1: Normalize data applying standardization.
2: Select model specified in JSON request.
3: Predict flow.
4: Return JSON response with the flow prediction, specifying the name of the model
   used, the classification result, and the corresponding class id.
```

The results we got and the discussion we made from this experimental topology, can also be read in chapter 4.

Chapter 4

Experimental Results and Discussion

In this chapter we present the obtained results with our balanced version of the Bot-IoT dataset, and we make a comparison with the related work presented in chapter 2. We also show and discuss the results we got using the LATAM-DDoS-IoT dataset, with transfer learning and concatenating the Bot-IoT dataset. Finally, we present the performance of our smart IDS after being deployed in the SDN experimental topology described in chapter 3.

In order to improve the reading of this chapter some results are in appendix B.

4.1 Bot-IoT Dataset Results

The results for multiclass and binary classification for the first feature set are presented in Tables B.1 and B.2, respectively; for the second feature set, we display the results in Tables B.3 and B.4; finally, Tables B.5 and B.6 show the results for the third feature set. From these results, it can be seen that Decision Tree and Random Forest have the best performance for both classification tasks in the three distinct feature sets, outperforming the DL models. On the other hand, SVM is the poorest performing model (in agreement with previous work in [41], with the 10-best feature set).

Our results show that Machine Learning models such as Random Forest and Decision Tree have strong performance, that is marginally better than that presented from Deep Learning models. Since all the results show a similar order of magnitude, we argue that given the relative small amount of features in all our datasets, Decision Tree methods show a robust performance that do not learn to depend on one particular feature, thus generalizing better. This also shows that traditional ML models are reliable and should not be discarded without proper evaluation, such as the one we carried out here, and particularly when using tabular data.

Our sampling methodology allowed us to use standard cost functions without weighting techniques, whilst addressing the balancing problem in the Bot-IoT dataset, in contrast to what is commonly done in the current state of the art (e.g., in [37]), which may lead to over tuning. Likewise, our feature sets included more characteristics compared to [65], capturing more information, whilst reducing the amount of manual feature engineering, in agreement with the ethos of current Machine Learning practices. With this, we presented extensive tests with both Machine Learning and Deep Learning models (compared, for instance, to [32], where only Machine Learning models were presented by the authors).

Given the importance of hardware real-time implementations, we consider it is relevant to evaluate the time performance of each model for classifying network traffic. As proposed in [25], we calculate the average number of flows per second our anomaly detection methods can classify. This experimentation was conducted in a MacBook Pro with Apple M1 Chip and 16 GB RAM for both the multiclass and the binary classification models, with the three feature sets. See Tables B.7 and B.8 for the first feature set; Tables B.9 and B.10 for the second feature set; and Tables B.11 and B.12 for the third feature set.

From the real world scenario tested in [25], on regular days around 500 flows/second passed through the network collector, while in dense traffic situations it achieved peaks of a maximum of 1,681 flows/second. Then, for the first feature set, from Tables B.7 and B.8, all the models, except for Random Forest in binary classification, are capable of analyzing the amount of flows/second required on heavy traffic days; whilst for the second feature set (Tables B.9 and B.10), all the models, except for SVM in both classification tasks, achieve the maximum peak. Finally, results for the third feature set (Tables B.11 and B.12) show that all the models, except for SVM and Random Forest in multiclass and binary classification, achieve the maximum amount of flows/second discussed.

From the results, it can be seen that Decision Tree is the best anomaly detection method for the proposed IDS, as shown in the results for accuracy, precision, recall, F1 score, and time performance, outperforming all the other models in the three feature sets, see Figure 4.1. We consider the third feature set as the most appropriate one for our novel IDS, since it shows stable results for Machine Learning and Deep Learning models (similar to results in the state of the art), both in multiclass and binary classifications, whilst not using timestamps as learnable features (which can lead to poor performance in a real-time real-world scenario). In addition, results in the literature use the Argus seq as one feature they feed in their models, as our third feature set does. See Figure B.1 and Figure B.2 for the Decision Tree confusion matrices using this feature set.

Not using neither timestamps nor the Argus sequence number (as in the second feature set), caused the Deep Learning models to have accuracies around 96% and 97% for both binary and multiclass classification, which is lower than the performance achieved by standard ML models. Although initially this result may appear surprising, we argue

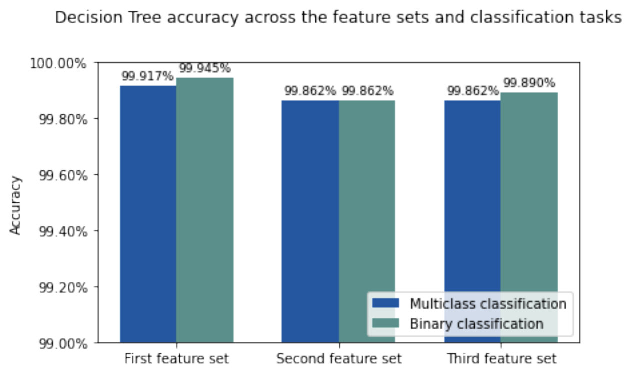


Figure 4.1: Decision Tree accuracy, as the best model, across the three different feature sets for binary and multiclass classifications.

this is due to the fact that the DL models learn to depend heavily on these particular features. Given the recurrent nature of the neural networks we assessed, these features (as provided in the original dataset) may display strong temporal dependencies (with strong correlations to the categories our models are classifying), once again strengthening the network dependence on these features, leading to poor generalization when implemented with online data. Nonetheless, Random Forest and Decision Tree still show the strongest performance when trained on this feature set, achieving results above 99.8%, which can be explained due to the random nature of these models, that allows overcoming dependencies on temporal data. It should also be noted, that we did not find other studies that use a similar set of features as those proposed in our second one, so we cannot establish a fair comparison to other works. In addition, the models trained on this feature set are totally independent of temporal characteristics such as timestamps and, particularly, Argus generated sequence numbers, which make strong generalization models suitable for online IDS implementations.

In addition to all these experiments, binary classification for Normal flows vs each DDoS/DoS protocol were performed. Tables B.13, B.14, and B.15, show the best anomaly detection models regarding accuracy, precision, recall, and F1 score, for each of those combinations. It can be seen that Decision Tree and Random Forest are the strongest models, achieving 100% across all the metrics in several combinations.

4.2 Comparison with Previous Works

Unlike previous works, this study addresses the class imbalance problem of the Bot-IoT dataset without adding class weights (which can lead to poor generalization, as seen in

[37, 32]), and without generating synthetic data. With this, we carried out an extensive experimentation of normal flows vs denial of service attacks, in binary and multiclass classifications. Under these analyses, three different feature sets were selected from the original dataset (with larger size and solving the problem of missing information, compared to [65]). It is also discussed why different flows processors could be used in a real-world scenario, and the importance of learning different features.

Likewise, this work shows an extensive evaluation of 7 distinct Machine Learning and Deep Learning models different to [65, 55, 29, 32], where only either ML or DL models are assessed by the authors. Also, we applied a systematic tuning of our models hyper parameters and dedicated 10% of our data to validation, in contrast to the process followed in [32]. With respect to performance evaluation, we not only presented confusion matrices and popular metrics (i.e., accuracy, precision, recall, and F1 score), but also added the time performance measurement to show the IDS feasibility of implementation in production networks, demonstrating that the best resulting models here presented are a realistic solution, this is in contrast to all the related works reviewed in chapter 2 that use the Bot-IoT dataset (except for [65]).

Our results match and exceed the current state of the art, with an average accuracy greater than 99% across our three different feature sets, and 100% across several combinations of Normal flows vs the DDoS/DoS subcategories. These results do not present bias towards a majority class. Compared to the works in our review that deal with class balancing, in [37] the accuracy for multiclass classification for normal flows vs DDoS and DoS attacks is 99.414%, whilst our best results for the same classification is 99.945% for the first feature set, 99.89% using the second feature set, and 99.917% using the third feature set. In addition, compared to [32], where the stable accuracy was 99% for binary classification in DDoS and DoS attacks protocols, and 97% for multiclass classification, we obtain accuracies greater than 99.85% using our three different feature sets for binary classification, and a best accuracy of at least 99.945% for the multiclass classification on our three feature sets.

4.3 LATAM-DDoS-IoT Dataset Results

As demonstrated in section 4.1 with the performance metrics, the best Machine Learning model was Decision Tree and the best Deep Learning model was Multi-layer Perceptron. Now, to extend our smart IDS we decided to conduct experiments using the LATAM dataset (for short) and the second feature set we proposed (the one with no timestamps neither the Argus sequence number). This second feature set, as stated, is the best for online IDS implementations, and removes any dependency with the Argus flow data generator (which, as mentioned, was originally used for the Bot-IoT dataset). Since our Java prototype uses Flowtbag, we now have a suitable data configuration to test for different parsers.

First of all, we conducted binary and multiclass classifications using the balanced version of the LATAM dataset, previously presented in chapter 3, for both the DoS and the DDoS versions. See Table B.16 for the hyper parameters setting for the best AI models using the LATAM-DoS-IoT dataset, and Table B.17 for the LATAM-DDoS-IoT dataset. This hyper parameters tuning was using the same systematic approach explained in chapter 3. The classification results are shown in Tables B.18 and B.19 for the LATAM-DoS-IoT dataset, and in Tables B.20 and B.21 for the LATAM-DDoS-IoT dataset.

The time performance results for the LATAM-DoS-IoT dataset are in Tables B.22 and B.23. For the LATAM-DDoS-IoT dataset see Tables B.24 and B.25. The presented results for the LATAM dataset indicate Decision Tree is marginally better than MLP, and both anomaly detection models still outperform the maximum peak of 1,681 flows per second discussed in [25]. In both dataset versions, binary classification is a bit better than multiclass classification, and the LATAM-DoS-IoT dataset classification results are slightly better than the results using the LATAM-DDoS-IoT dataset.

For combining the LATAM dataset with the Bot-IoT dataset in order to extend our smart IDS with real traffic from real customers and physical IoT devices, we conducted two experiments: one applying transfer learning, and another one concatenating both datasets. More details can be found in the followed subsections.

4.3.1 Transfer Learning

Transfer learning is an AI technique focused on transferring knowledge from a source domain to a target domain, when both of them share similarities [67]. We want to transfer knowledge from the LATAM dataset to our balanced version of the Bot-IoT dataset, because we believe more data (millions vs thousands records, respectively) can positively indicate more generalization of the problem domain (i.e., anomaly detection in network flows).

Transfer learning helped us to initialize the weights and biases of our new MLP models, instead of applying random initializations. To accomplish that, we took our neural networks trained with the LATAM dataset, and applied freezing to all their linear layers except for the last one, in order to train our new and final linear layer with our balanced version of the Bot-IoT dataset, but now in much less time because we accelerated the whole learning process. For the Decision Tree we tried to implement some kind of incremental learning, with partial fits, but so far Scikit-Learn does not include that API implementation for this model. See Tables B.32 and B.33 for the transfer learning results with the adjusted MLP models.

Our recall results from both tables, for binary classification, indicate the models detected above 95% of the attacks. For multiclass classification in both tables, precision was the highest metric, indicating flows identified as attacks were in their majority correctly classified. We can argue that the normal flows heavily affected our accuracies, because the testbed designed for the Bot-IoT dataset incorporates synthetic normal traffic generated using Ostinato, which is different to our normal traffic collected from real customers consuming real services. This aspect conducts to a negative transfer [51] between both domains, which can be slightly improved with a fine-tuning to find a better network setting that improves our detection metrics, for instance freezing different layers (not all of them).

4.3.2 Datasets Concatenation

Other experiments we conducted were concatenating our balanced version of the Bot-IoT dataset with each balanced version of the LATAM dataset, which took more time for training but allowed the models to learn all data from both domains from scratch, leading to best classification results. The resulting dataset size for the concatenation of our balanced version of the Bot-IoT dataset with the balanced version of the LATAM-DoS-IoT dataset was 2,443,442 samples; for our balanced version of the Bot-IoT dataset concatenated with the balanced version of the LATAM-DDoS-IoT dataset was 2,467,793 samples. From now on, for easy reading, we will call LATAM-Bot-DoS-IoT our Bot-IoT + LATAM-DoS-IoT combination, and LATAM-Bot-DDoS-IoT our Bot-IoT + LATAM-DDoS-IoT combination.

The hyper parameters setting (applying the same systematic tuning approach we have followed so far) for the LATAM-Bot-DoS-IoT dataset is in Table B.26, and for the LATAM-Bot-DDoS-IoT dataset in Table B.27. See Tables B.28 and B.29 for the classification results with the LATAM-Bot-DoS-IoT dataset, and Tables B.30 and B.31 for the LATAM-Bot-DDoS-IoT dataset.

Tables B.34 and B.35 show the time performance of the anomaly detection models obtained from the LATAM-Bot-DoS-IoT dataset. See Tables B.36 and B.37 for the time performance with the LATAM-Bot-DDoS-IoT dataset.

These performance results indicate that learning all data from scratch conducts to better classification results, and now our anomaly detection models have successfully learned information from both domains: the Bot-IoT dataset and the LATAM dataset. Since Decision Tree outperforms the MLP in time performance and both classification tasks, we decided to choose it to conduct a final experiment deploying our smart IDS in the SDN topology we designed and previously explained in chapter 3. We will use learning from both domains (to increase generalization), and since we will launch DDoS attacks, the Decision Tree obtained from the LATAM-Bot-DDoS-IoT dataset is going to be evaluated.

4.4 IDS Deployment Results in SDN Testbed

When deploying our smart IDS in the SDN topology we created, we wanted to know the classification results, but also measure how much time it takes for our Java app to process the packets, and how much waiting time is required to free all the connections in the network. Since our Java prototype currently works only for TCP traffic, we evaluated the Decision Tree for binary classification from the LATAM-Bot-DDoS-IoT dataset.

After three runs of the SDN experiment described in chapter 3, we got the results from Table 4.1. It took an average of 11 minutes for the network to free all the connections, which was the average waiting time required between experiments to run them; an average of 0.0025 seconds for processing each TCP packet (i.e., adding the packet into a new or existing flow, and classify the flow if its connection is closed); and we got an average classification accuracy of 93.181%, and a best accuracy of 94.608%. See Figure 4.2 for the confusion matrix of this best binary classification, and Table 4.2 for the other classification metrics.

Table 4.1: IDS results in SDN testbed.

Run	Time to free the connections	Avg time to process packets	Accuracy
1	10:25.13 min	0.0025 s	92.118%
2	11:40.9 min	0.0026 s	92.818%
3	10:58.32 min	0.0025 s	94.608%

Table 4.2: Best binary classification results in the SDN testbed.

Model	Accuracy	Precision	Recall	F1 score
Decision Tree	94.608%	100%	91.406%	95.510%

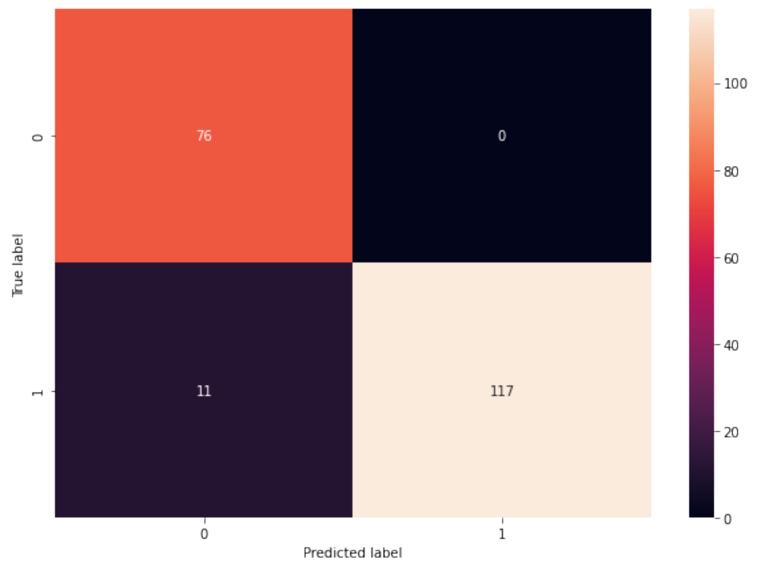
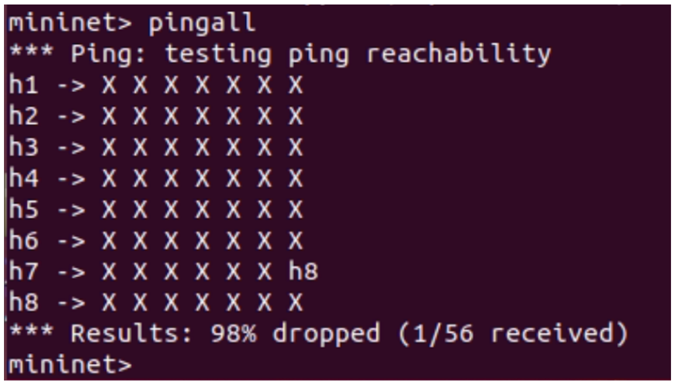


Figure 4.2: Best confusion matrix for Decision Tree binary classification in the SDN testbed. The numbers in the axes mean 0 for Normal class, and 1 for Attack class.

From Table 4.2, we can see that 100% of the flows the smart IDS identified as attacks were correctly classified, and 91.406% of the attack flows were detected. The F1 score is 95.51%, indicating a good balance between the precision and recall. Since our results do not show misclassifying of legitimate traffic, and show detection of above 90% of the attacks, we can argue our smart IDS is a suitable defense system against denial of service attacks in SDN environments.

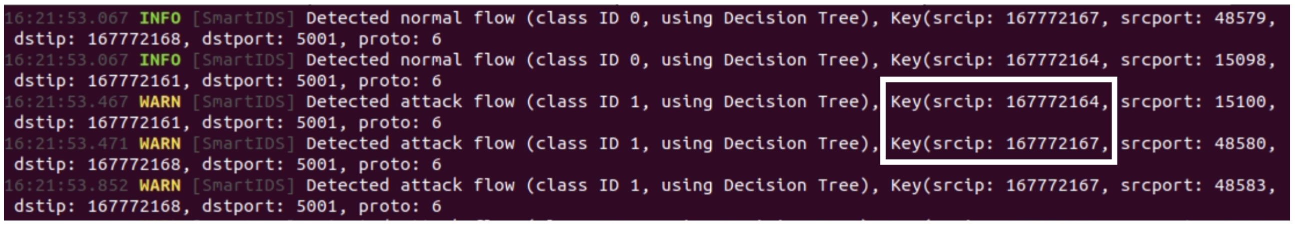
To take a look at how devastating our attacks were, we show a ping reachability test from Mininet in Figure 4.3, where we can see 98% of the communication was dropped. This test was executed just after the 60 seconds of launching all the traffic in one of the experiment runs, and the only ping that remained working was between the attack host h7 and its victim (i.e., the server h8). That can be explained because 10 seconds were not enough for a single attacker to completely overflow its victim. In this experiment run, an average of 860,016 packets were transmitted by each attacker node in just 10 seconds, to the victim nodes h1 and h8 (as presented in chapter 3).

Figure 4.4 shows a fragment of our smart IDS logs while classifying traffic in ONOS: we log the classification results, the model the administrator specified for detecting the traffic, and the tuple that conforms the flow (source IP, source port, destination IP, destination port, and protocol). The white square in the image shows our smart IDS is able to detect attacks from different source IPs. These source IPs values correspond to the



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X X X
h2 -> X X X X X X X
h3 -> X X X X X X X
h4 -> X X X X X X X
h5 -> X X X X X X X
h6 -> X X X X X X X
h7 -> X X X X X X h8
h8 -> X X X X X X X
*** Results: 98% dropped (1/56 received)
mininet>
```

Figure 4.3: Ping reachability test using Mininet.



```
16:21:53.067 INFO [SmartIDS] Detected normal flow (class ID 0, using Decision Tree), Key(srcip: 167772167, srcport: 48579,
 dstip: 167772168, dstport: 5001, proto: 6
16:21:53.067 INFO [SmartIDS] Detected normal flow (class ID 0, using Decision Tree), Key(srcip: 167772164, srcport: 15098,
 dstip: 167772161, dstport: 5001, proto: 6
16:21:53.467 WARN [SmartIDS] Detected attack flow (class ID 1, using Decision Tree), Key(srcip: 167772164, srcport: 15100,
 dstip: 167772161, dstport: 5001, proto: 6
16:21:53.471 WARN [SmartIDS] Detected attack flow (class ID 1, using Decision Tree), Key(srcip: 167772167, srcport: 48580,
 dstip: 167772168, dstport: 5001, proto: 6
16:21:53.852 WARN [SmartIDS] Detected attack flow (class ID 1, using Decision Tree), Key(srcip: 167772167, srcport: 48583,
 dstip: 167772168, dstport: 5001, proto: 6
```

Figure 4.4: Logs of our smart IDS running in ONOS.

attackers h4 and h7.

In the next chapter we conclude this thesis and present future work that can be done for exploring new directions and extending what we have accomplished.

Chapter 5

Conclusions and Future Work

In this last chapter we enclose the conclusions from our research, and we propose future work that can be done to further extend what we have made.

5.1 Conclusions

Anomaly detection is a vast problem in Artificial Intelligence. Its application in network traffic leads to the creation of defense systems against cyberattacks that can be devastating for people and organizations, so their level of protection is crucial. In this thesis, we focused on DDoS and DoS attacks, which represent the most common threat against and from technologies such as IoT and 5G communication networks.

For one set of experiments we used the Bot-IoT dataset, a state-of-the-art collection of data for protecting IoT networks. The methodology proposed addresses the class imbalance problem of the original dataset (without neither adding synthetic data nor class weights) leading to the creation of a novel IDS based on AI models which focuses on DDoS and DoS attacks based on UDP, TCP and HTTP protocols. The proposed IDS presents results without biases towards a majority class, achieving an average accuracy greater than 99% with our three distinct feature sets, where the Decision Tree is the outstanding anomaly detection model, whilst being feasible for implementation in real-time production environments, with a remarkable time performance for heavy traffic days (evaluating more than 1,681 flows/second). Also, we achieved 100% across accuracy, precision, recall, and F1 score metrics, with the Decision Tree and the Random Forest for several combinations of Normal flows vs the DDoS/DoS protocols.

We extended the capacity of this smart IDS adding real normal traffic from real clients consuming real services, and also real attack traffic to physical IoT devices, creating a new dataset called: the LATAM-DDoS-IoT. First of all, to leverage the categories and subcategories present in it, we conducted binary and multiclass classifications with its balanced DoS and DDoS versions, getting an average accuracy of 99.967% and 98.872%, respectively. Then, we combined our balanced version of the Bot-IoT dataset applying transfer learning, showing how different our datasets were from each other. Also, in another experiment we concatenated both datasets to get a higher level of generalization from both domains, achieving high results such as the 99.99% of accuracy obtained from the Decision Tree in binary classification for DoS.

Since one of our initial objectives was to deploy our resulting IDS in a production SDN simulated environment, we created a hybrid architecture using ONOS and Mininet, and coded a Java app for communicating with our cloud-hosted Flask REST API. We can say our smart IDS behaves strongly, where 100% of the flows identified as attacks were correctly classified, and above 90% of the attack flows were detected. Our defense system does not misclassify legitimate traffic, and presents an average time performance of even more than 30,000 flows per second.

Part of the results of this thesis have been published in the paper "Transport and Application Layer DDoS Attacks Detection to IoT Devices by Using Machine Learning and Deep Learning Models" [24].

5.2 Future Work

This work can be extended creating and deploying an Intrusion Prevention System as mitigation management strategy, which integrates and communicates with our Intrusion Detection System. The resulting architecture will not only allow to detect attacks (as we have done), but also to stop the identified attackers, diminishing the network damage.

Another direction for further improvement, is to test our IDS in a fully physical SDN architecture. We moved from a 100% simulated architecture (as the virtualized simulation in [54]), to a hybrid architecture running ONOS as a Linux service instead of using Docker [6]. Now, the next component to remove would be Mininet, to avoid virtualized hosts and implement real physical devices.

So far, our Java functional prototype only works for capturing TCP traffic. Naturally, an area of improvement would be to make it suitable for capturing also UDP traffic, in order to take the most out of our AI models.

We also proposed a new dataset, with DoS and DDoS versions, that we globally called the LATAM-DDoS-IoT dataset. It will be interesting to see what kind of experiments derive from its use, as one-class classifiers, since their real traffic from real clients and physical IoT devices make it convenient for real production environments.

Appendix A

Chapter 3 Appendix

Table A.1: Summary of ML models parameters for the first feature set.

Model	**Binary classification**	**Multiclass classification**
SVM	• Kernel: Radial Basis Function • Max iterations: 70,000	• Kernel: Linear • Max iterations: 70,000
Decision Tree	• Max depth: 11 • Entropy criterion	• Max depth: 10 • Entropy criterion
Random Forest	• Max depth: 11 • Entropy criterion • Trees: 12	• Max depth: 10 • Entropy criterion • Trees: 9

Table A.2: Summary of ML models parameters for the second feature set.

Model	**Binary classification**	**Multiclass classification**
SVM	• Kernel: Radial Basis Function • Max iterations: 50,000	• Kernel: Radial Basis Function • Max iterations: 50,000
Decision Tree	• Max depth: 7 • Entropy criterion	• Max depth: 8 • Entropy criterion
Random Forest	• Max depth: 7 • Entropy criterion • Trees: 2	• Max depth: 8 • Entropy criterion • Trees: 9

Table A.3: Summary of ML models parameters for the third feature set.

Model	**Binary classification**	**Multiclass classification**
SVM	• Kernel: Radial Basis Function • Max iterations: 50,000	• Kernel: Radial Basis Function • Max iterations: 70,000
Decision Tree	• Max depth: 8 • Entropy criterion	• Max depth: 7 • Entropy criterion
Random Forest	• Max depth: 8 • Entropy criterion • Trees: 11	• Max depth: 7 • Entropy criterion • Trees: 21

Table A.4: Summary of DL models parameters for the three feature sets.

Model	**Binary classification**	**Multiclass classification**
RNN, LSTM, GRU, MLP	• Classes: 2 • Batch size: 128 • Input size: 18, 15, and 16 • Hidden size: 128 (512 for MLP) • Layers: 3 (4 for MLP) • Sequence length: 1 (None for MLP) • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU	• Classes: 4 • Batch size: 128 • Input size: 18, 15, and 16 • Hidden size: 128 (512 for MLP) • Layers: 3 (4 for MLP) • Sequence length: 1 (None for MLP) • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU

Appendix B

Chapter 4 Appendix

Table B.1: Multiclass classification results for the first feature set.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Random Forest	99.945%	99.945%	99.945%	99.945%
Decision Tree	99.917%	99.918%	99.917%	99.917%
LSTM	99.862%	99.862%	99.864%	99.863%
GRU	99.862%	99.861%	99.865%	99.863%
MLP	99.862%	99.861%	99.865%	99.863%
RNN	99.807%	99.806%	99.811%	99.808%
SVM	94.056%	94.661%	94.056%	94.122%

Table B.2: Binary classification results for the first feature set.

Model	Accuracy	Precision	Recall	F1 score
Random Forest	99.972%	99.973%	99.972%	99.972%
Decision Tree	99.945%	99.945%	99.945%	99.945%
RNN	99.862%	99.889%	99.926%	99.908%
MLP	99.862%	99.889%	99.926%	99.908%
GRU	99.835%	99.852%	99.926%	99.889%
LSTM	99.807%	99.816%	99.926%	99.871%
SVM	98.404%	98.431%	98.404%	98.388%

Table B.3: Multiclass classification results for the second feature set.

Model	Accuracy	Precision	Recall	F1 score
Random Forest	99.890%	99.890%	99.890%	99.890%
Decision Tree	99.862%	99.863%	99.862%	99.862%
MLP	96.340%	96.372%	96.375%	96.354%
GRU	96.230%	96.276%	96.250%	96.249%
LSTM	96.010%	96.042%	96.042%	99.022%
RNN	95.019%	95.126%	95.027%	95.049%
SVM	75.482%	78.481%	75.482%	75.218%

Table B.4: Binary classification results for the second feature set.

Model	Accuracy	Precision	Recall	F1 score
Decision Tree	99.862%	99.862%	99.862%	99.862%
Random Forest	99.835%	99.835%	99.835%	99.835%
GRU	97.111%	97.797%	98.339%	98.067%
LSTM	96.753%	97.437%	98.228%	97.831%
MLP	96.505%	97.498%	97.822%	97.66%
RNN	96.147%	97.381%	97.453%	97.417%
SVM	81.205%	84.721%	81.205%	76.825%

Table B.5: Multiclass classification results for the third feature set.

Model	Accuracy	Precision	Recall	F1 score
Random Forest	99.917%	99.918%	99.917%	99.917%
Decision Tree	99.862%	99.863%	99.862%	99.862%
GRU	99.697%	99.693%	99.702%	99.697%
MLP	99.642%	99.638%	99.646%	99.641%
LSTM	99.560%	99.557%	99.565%	99.561%
RNN	99.560%	99.556%	99.566%	99.560%
SVM	89.351%	89.603%	89.351%	89.347%

Table B.6: Binary classification results for the third feature set.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	99.890%	99.890%	99.890%	99.890%
Random Forest	99.835%	99.835%	99.835%	99.835%
MLP	99.697%	99.742%	99.852%	99.797%
GRU	99.642%	99.595%	99.926%	99.760%
RNN	99.615%	99.595%	99.889%	99.742%
LSTM	99.615%	99.559%	99.926%	99.742%
SVM	94.194%	94.347%	94.194%	94.246%

Table B.7: Multiclass classification time performance for the first feature set.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	29,453	790.687
MLP	8,306	537.827
SVM	4,283	139.935
RNN	4,158	59.906
GRU	2,497	51.750
LSTM	2,388	20.823
Random Forest	1,813	65.692

Table B.8: Binary classification time performance for the first feature set.

Model	Avg flows/second	Stddev flows/second
Decision Tree	29,452	716.966
MLP	9,411	38.543
SVM	4,956	25.011
RNN	4,375	77.826
GRU	2,661	8.712
LSTM	2,610	5.094
Random Forest	1,350	81.339

Table B.9: Multiclass classification time performance for the second feature set.

Model	Avg flows / second	Stddev flows / second
Decision Tree	30,362	681.989
MLP	9,319	48.970
RNN	4,742	49.485
GRU	2,864	17.051
LSTM	2,702	33.465
Random Forest	1,954	15.106
SVM	651	8.033

Table B.10: Binary classification time performance for the second feature set.

Model	Avg flows/second	Stddev flows/second
Decision Tree	29,940	523.611
MLP	9,177	142.993
RNN	4,697	27.281
Random Forest	4,571	60.758
GRU	2,763	49.491
LSTM	2,687	22.446
SVM	866	7.232

Table B.11: Multiclass classification time performance for the third feature set.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	33,094	378.595
MLP	9,934	257.982
RNN	4,823	99.721
GRU	2,918	51.754
LSTM	2,877	65.451
SVM	1,171	17.393
Random Forest	994	22.931

Table B.12: Binary classification time performance for the third feature set.

Model	**Avg flows/second**	**Stddev flows/second**
Decision Tree	32,607	151.361
MLP	10,017	101.060
RNN	4,883	123.540
GRU	2,996	34.989
LSTM	2,864	79.405
Random Forest	1,668	89.448
SVM	1,422	10.979

Table B.13: Binary classification results for Normal flows vs DDoS/DoS subcategories (protocols), using the first feature set.

Classes	**Best Model(s)**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Normal vs DDoS	Random Forest	99.956%	99.956%	99.956%	99.956%
Normal vs DDoS UDP	Decision Tree and Random Forest	99.853%	99.853%	99.853%	99.853%
Normal vs DDoS HTTP	Decision Tree and Random Forest	100%	100%	100%	100%
Normal vs DDoS TCP	Decision Tree and Random Forest	100%	100%	100%	100%
Normal vs DoS	Random Forest	99.956%	99.956%	99.956%	99.956%
Normal vs DoS UDP	All models, except for SVM	100%	100%	100%	100%
Normal vs DoS HTTP	Decision Tree	100%	100%	100%	100%
Normal vs DoS TCP	All models, except for SVM	100%	100%	100%	100%

Table B.14: Binary classification results for Normal flows vs DDoS/DoS subcategories (protocols), using the second feature set.

Classes	**Best Model(s)**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Normal vs DDoS	Decision Tree and Random Forest	99.956%	99.956%	99.956%	99.956%
Normal vs DDoS UDP	Decision Tree and Random Forest	99.853%	99.853%	99.853%	99.853%
Normal vs DDoS HTTP	Decision Tree and Random Forest	100%	100%	100%	100%
Normal vs DDoS TCP	Decision Tree and Random Forest	100%	100%	100%	100%
Normal vs DoS	Random Forest	99.868%	99.868%	99.868%	99.868%
Normal vs DoS UDP	All models, except for SVM	100%	100%	100%	100%
Normal vs DoS HTTP	Decision Tree	100%	100%	100%	100%
Normal vs DoS TCP	Decision Tree and Random Forest	100%	100%	100%	100%

Table B.15: Binary classification results for Normal flows vs DDoS/DoS subcategories (protocols), using the third feature set.

Classes	**Best Model(s)**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Normal vs DDoS	Random Forest	99.956%	99.956%	99.956%	99.956%
Normal vs DDoS UDP	Random Forest	99.853%	99.853%	99.853%	99.853%
Normal vs DDoS HTTP	Decision Tree and Random Forest	100%	100%	100%	100%
Normal vs DDoS TCP	Random Forest	100%	100%	100%	100%
Normal vs DoS	Random Forest	99.868%	99.868%	99.868%	99.868%
Normal vs DoS UDP	All models, except for SVM	100%	100%	100%	100%
Normal vs DoS HTTP	Decision Tree	100%	100%	100%	100%
Normal vs DoS TCP	All models, except for SVM	100%	100%	100%	100%

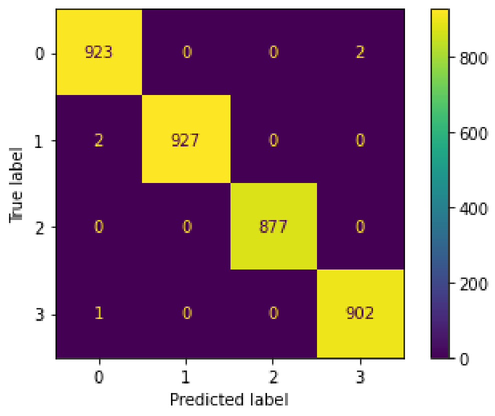


Figure B.1: Confusion matrix for Decision Tree multiclass classification, using the best feature set. The numbers in the axes mean 0 for Normal class, 1 for UDP class, 2 for TCP class, and 3 for HTTP class.

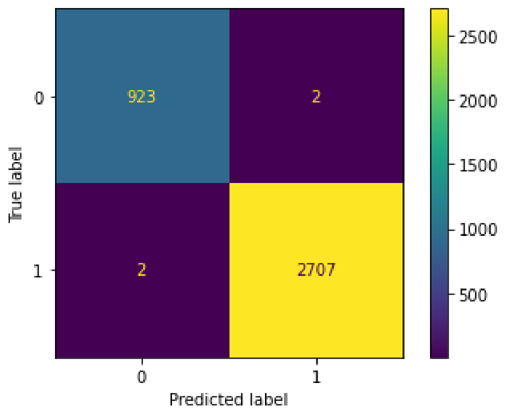


Figure B.2: Confusion matrix for Decision Tree binary classification, using the best feature set. The numbers in the axes mean 0 for Normal class, and 1 for Attack class.

Table B.16: Summary of AI models parameters for the LATAM-DoS-IoT Dataset.

Model	**Binary classification**	**Multiclass classification**
Decision Tree	• Max depth: 11 • Entropy criterion	• Max depth: 12 • Entropy criterion
MLP	• Classes: 2 • Batch size: 128 • Input size: 15 • Hidden size: 512 • Layers: 4 • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU	• Classes: 4 • Batch size: 128 • Input size: 15 • Hidden size: 512 • Layers: 4 • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU

Table B.17: Summary of AI models parameters for the LATAM-DDoS-IoT Dataset.

Model	**Binary classification**	**Multiclass classification**
Decision Tree	• Max depth: 14 • Entropy criterion	• Max depth: 14 • Entropy criterion
MLP	• Classes: 2 • Batch size: 128 • Input size: 15 • Hidden size: 512 • Layers: 4 • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU	• Classes: 4 • Batch size: 128 • Input size: 15 • Hidden size: 512 • Layers: 4 • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU

Table B.18: Binary classification results for the LATAM-DoS-IoT dataset.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	99.996%	99.996%	99.996%	99.996%
MLP	99.940%	99.999%	99.912%	99.955%

Table B.19: Multiclass classification results for the LATAM-DoS-IoT dataset.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	99.995%	99.995%	99.995%	99.995%
MLP	99.938%	99.938%	99.940%	99.937%

Table B.20: Binary classification results for the LATAM-DDoS-IoT dataset.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	98.911%	98.942%	98.911%	98.915%
MLP	98.834%	99.978%	98.287%	99.125%

Table B.21: Multiclass classification results for the LATAM-DDoS-IoT dataset.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	98.908%	98.857%	98.908%	98.605%
MLP	98.834%	98.793%	98.839%	98.456%

Table B.22: Binary classification time performance for the LATAM-DoS-IoT dataset.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	31,967	480.296
MLP	9,669	124.086

Table B.23: Multiclass classification time performance for the LATAM-DoS-IoT dataset.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	31,070	564.739
MLP	9,623	59.003

Table B.24: Binary classification time performance for the LATAM-DDoS-IoT dataset.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	32,349	215.802
MLP	9,990	39.523

Table B.25: Multiclass classification time performance for the LATAM-DDoS-IoT dataset.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	32,320	257.284
MLP	9,764	108.798

Table B.26: Summary of AI models parameters for the LATAM-Bot-DoS-IoT dataset.

Model	**Binary classification**	**Multiclass classification**
Decision Tree	• Max depth: 12 • Entropy criterion	• Max depth: 12 • Entropy criterion
MLP	• Classes: 2 • Batch size: 128 • Input size: 15 • Hidden size: 512 • Layers: 4 • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU	• Classes: 4 • Batch size: 128 • Input size: 15 • Hidden size: 512 • Layers: 4 • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU

Table B.27: Summary of AI models parameters for the LATAM-Bot-DDoS-IoT dataset.

Model	**Binary classification**	**Multiclass classification**
Decision Tree	• Max depth: 13 • Entropy criterion	• Max depth: 14 • Entropy criterion
MLP	• Classes: 2 • Batch size: 128 • Input size: 15 • Hidden size: 512 • Layers: 4 • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU	• Classes: 4 • Batch size: 128 • Input size: 15 • Hidden size: 512 • Layers: 4 • Epochs: 100 • Optimizer: Adam • Loss function: Cross Entropy • Learning rate: 0.0011 • Device: CPU

Table B.28: Binary classification results for the LATAM-Bot-DoS-IoT dataset.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	99.990%	99.990%	99.990%	99.990%
MLP	99.801%	99.859%	99.843%	99.851%

Table B.29: Multiclass classification results for the LATAM-Bot-DoS-IoT dataset.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	99.989%	99.989%	99.989%	99.989%
MLP	99.793%	99.791%	99.789%	99.789%

Table B.30: Binary classification results for the LATAM-Bot-DDoS-IoT dataset.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	98.884%	98.916%	98.884%	98.888%
MLP	98.655%	99.840%	98.156%	98.991%

Table B.31: Multiclass classification results for the LATAM-Bot-DDoS-IoT dataset.

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
Decision Tree	98.867%	98.868%	98.867%	98.625%
MLP	98.654%	98.658%	98.661%	98.376%

Table B.32: Transfer learning results using the LATAM-DoS-IoT dataset as source domain and the Bot-IoT dataset as target domain.

Model	**Classification**	**Accuracy**	**Precision**	**Recall**	**F1 score**
MLP	Binary	83.187%	84.081%	95.533%	89.442%
MLP	Multiclass	78.481%	79.734%	78.517%	78.255%

Table B.33: Transfer learning results using the LATAM-DDoS-IoT dataset as source domain and the Bot-IoT dataset as target domain.

Model	**Classification**	**Accuracy**	**Precision**	**Recall**	**F1 score**
MLP	Binary	87.342%	87.272%	97.195%	91.966%
MLP	Multiclass	82.223%	84.295%	82.355%	81.885%

Table B.34: Binary classification time performance for the LATAM-Bot-DoS-IoT dataset.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	31,334	344.921
MLP	9,781	33.618

Table B.35: Multiclass classification time performance for the LATAM-Bot-DoS-IoT dataset.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	32,785	179.086
MLP	9,998	62.752

Table B.36: Binary classification time performance for the LATAM-Bot-DDoS-IoT dataset.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	31,485	142.967
MLP	9,592	181.967

Table B.37: Multiclass classification time performance for the LATAM-Bot-DDoS-IoT dataset.

Model	**Avg flows / second**	**Stddev flows / second**
Decision Tree	32,101	264.161
MLP	9,756	170.605

Bibliography

[1] Aligo. https://aligo.com.co. Accessed on 14 May 2022.

[2] Argus. https://openargus.org. Accessed on 10 December 2021.

[3] The bot-iot dataset. https://research.unsw.edu.au/projects/bot-iot-dataset. Accessed on 26 January 2021.

[4] Cicflowmeter. https://github.com/ahlashkari/CICFlowMeter. Accessed on 13 December 2021.

[5] Decision trees - scikit-learn. https://scikit-learn.org/0.24/modules/tree.html. Accessed on 9 April 2022.

[6] Docker. https://www.docker.com. Accessed on 3 April 2022.

[7] Flask. https://palletsprojects.com/p/flask. Accessed on 29 March 2022.

[8] Flowtbag. https://github.com/DanielArndt/flowtbag. Accessed on 13 December 2021.

[9] Goldeneye. https://github.com/jseidl/GoldenEye. Accessed on 5 November 2021.

[10] hping. http://www.hping.org. Accessed on 5 November 2021.

[11] Http ddos detector. https://github.com/jatj/httpDetector. Accessed on 29 March 2022.

[12] iperf. https://iperf.fr. Accessed on 29 March 2022.

[13] Nmap. https://nmap.org. Accessed on 29 March 2022.

[14] Node-red. https://nodered.org/. Accessed on 19 March 2022.

[15] Open network operating system (onos) sdn controller for sdn/nfv solutions. https://opennetworking.org/onos/. Accessed on 3 November 2020.

[16] Open networking foundation. https://opennetworking.org/. Accessed on 13 December 2021.

[17] Ostinato. https://ostinato.org/. Accessed on 19 March 2022.

[18] P4. https://p4.org/. Accessed on 13 December 2021.

[19] Ryu. https://ryu-sdn.org/. Accessed on 19 March 2022.

[20] tcpdump. https://www.tcpdump.org. Accessed on 29 March 2022.

[21] Tshark network analysis tool. https://www.wireshark.org/. Accessed on 19 March 2022.

[22] Understanding denial-of-service attacks. https://us-cert.cisa.gov/ncas/tips/ST04-015. Accessed on 3 September 2021.

[23] ALEXANDER GUTNIKOV, OLEG KUPREEV, Y. S. Ddos attacks in q3 2021 — securelist. https://securelist.com/ddos-attacks-in-q3-2021/104796/, 2021. Accessed on 15 November 2021.

[24] ALMARAZ-RIVERA, J. G., PEREZ-DIAZ, J. A., AND CANTORAL-CEBALLOS, J. A. Transport and application layer ddos attacks detection to iot devices by using machine learning and deep learning models. *Sensors 22*, 9 (2022).

[25] ASSIS, M. V., CARVALHO, L. F., LLORET, J., AND PROENÇA, M. L. A gru deep learning system against attacks in software defined networks. *Journal of Network and Computer Applications 177* (2021), 102942.

[26] BANNOUR, F., SOUIHI, S., AND MELLOUK, A. Distributed sdn control: Survey, taxonomy, and challenges. *IEEE Communications Surveys Tutorials 20*, 1 (2018), 333–354.

[27] BERDE, P., GEROLA, M., HART, J., HIGUCHI, Y., KOBAYASHI, M., KOIDE, T., LANTZ, B., O'CONNOR, B., RADOSLAVOV, P., SNOW, W., AND PARULKAR, G. Onos: Towards an open, distributed sdn os. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking* (New York, NY, USA, 2014), HotSDN '14, Association for Computing Machinery, p. 1–6.

[28] BILGE, L., AND DUMITRAŞ, T. Before we knew it: An empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (New York, NY, USA, 2012), CCS '12, Association for Computing Machinery, p. 833–844.

[29] BISWAS, R., AND ROY, S. Botnet traffic identification using neural networks. *Multimedia Tools and Applications 80* (2021), 24147–24171.

[30] CHAWLA, N. V., BOWYER, K. W., HALL, L. O., AND KEGELMEYER, W. P. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research 16*, 1 (jun 2002), 321–357.

[31] CHOLLET, F., ET AL. Keras. https://keras.io, 2015. Accessed on 5 November 2021.

[32] CHURCHER, A., ULLAH, R., AHMAD, J., UR REHMAN, S., MASOOD, F., GOGATE, M., ALQAHTANI, F., NOUR, B., AND BUCHANAN, W. J. An experimental analysis of attack classification using machine learning in iot networks. *Sensors 21*, 2 (2021).

[33] DING, F., LI, H., LUO, F., HU, H., CHENG, L., XIAO, H., AND GE, R. Deeppower: Non-intrusive and deep learning-based detection of iot malware using power side channels. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security* (New York, NY, USA, 2020), ASIA CCS '20, Association for Computing Machinery, p. 33–46.

[34] DUCHI, J., HAZAN, E., AND SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*, 61 (2011), 2121–2159.

[35] FARO, C. Ddos attacks in q3 2021 — kaspersky. https://usa.kaspersky.com/about/press-releases/2021_kaspersky-finds-ddos-attacks-in-q3-grow-by-24-become-more-sophisticated, 2021. Accessed on 15 November 2021.

[36] FERRAG, M. A., MAGLARAS, L., MOSCHOYIANNIS, S., AND JANICKE, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications 50* (2020), 102419.

[37] GE, M., FU, X., SYED, N., BAIG, Z., TEO, G., AND ROBLES-KELLY, A. Deep learning-based intrusion detection for iot networks. In *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)* (2019), pp. 256–25609.

[38] HUANG, J., AND LING, C. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering 17*, 3 (2005), 299–310.

[39] KHAN, S. S., AND MADDEN, M. G. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review 29*, 3 (2014), 345–374.

[40] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980, 2014. Accessed on 16 March 2022.

[41] KORONIOTIS, N., MOUSTAFA, N., SITNIKOVA, E., AND TURNBULL, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems 100* (2019), 779–796.

[42] KREUTZ, D., RAMOS, F. M. V., VERÍSSIMO, P. E., ROTHENBERG, C. E., AZODOLMOLKY, S., AND UHLIG, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE 103*, 1 (2015), 14–76.

[43] LANTZ, B., HELLER, B., AND MCKEOWN, N. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (New York, NY, USA, 2010), Hotnets-IX, Association for Computing Machinery.

[44] LI, H., WEI, F., AND HU, H. Enabling dynamic network access control with anomaly-based ids and sdn. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization* (New York, NY, USA, 2019), SDN-NFVSec '19, Association for Computing Machinery, p. 13–16.

[45] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev. 38*, 2 (mar 2008), 69–74.

[46] MCMILLEN, D. Internet of threats: Iot botnets drive surge in network attacks. https://securityintelligence.com/posts/internet-of-threats-iot-botnets-network-attacks/, 2021. Accessed on 17 December 2021.

[47] MEHR, S. Y., AND RAMAMURTHY, B. An svm based ddos attack detection method for ryu sdn controller. In *Proceedings of the 15th International Conference on Emerging Networking EXperiments and Technologies* (New York, NY, USA, 2019), CoNEXT '19, Association for Computing Machinery, p. 72–73.

[48] MERGENDAHL, S., SISODIA, D., LI, J., AND CAM, H. Source-end ddos defense in iot environments. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy* (New York, NY, USA, 2017), IoTS&P '17, Association for Computing Machinery, p. 63–64.

[49] MIRKOVIC, J., AND REIHER, P. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev. 34*, 2 (Apr. 2004), 39–53.

[50] OLEG KUPREEV, EKATERINA BADOVSKAYA, A. G. Ddos attacks in q2 2020 — securelist. https://securelist.com/ddos-attacks-in-q2-2020/98077/, 2020. Accessed on 15 November 2021.

[51] PAN, S. J., AND YANG, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering 22*, 10 (2010), 1345–1359.

[52] PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., DESMAISON, A., KOPF, A., YANG, E., DEVITO, Z., RAISON, M., TEJANI, A., CHILAMKURTHY, S., STEINER, B., FANG, L., BAI, J., AND CHINTALA, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[53] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[54] PÉREZ-DÍAZ, J. A., VALDOVINOS, I. A., CHOO, K.-K. R., AND ZHU, D. A flexible sdn-based architecture for identifying and mitigating low-rate ddos attacks using machine learning. *IEEE Access 8* (2020), 155859–155872.

[55] SHAFIQ, M., TIAN, Z., BASHIR, A. K., DU, X., AND GUIZANI, M. Corrauc: A malicious bot-iot traffic detection method in iot network using machine-learning techniques. *IEEE Internet of Things Journal 8*, 5 (2021), 3242–3254.

[56] SHAGHAGHI, A., KAAFAR, M. A., AND JHA, S. Wedgetail: An intrusion prevention system for the data plane of software defined networks. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*

(New York, NY, USA, 2017), ASIA CCS '17, Association for Computing Machinery, p. 849–861.

[57] SINHA, S. State of iot 2021. https://iot-analytics.com/number-connected-iot-devices/, 2021. Accessed on 7 January 2022.

[58] SRINIVASA GOPALAN, S. Towards effective detection of botnet attacks using bot-iot dataset. https://scholarworks.rit.edu/theses/10698, 2021. Accessed on 19 August 2021.

[59] TANABE, R., TAMAI, T., FUJITA, A., ISAWA, R., YOSHIOKA, K., MATSUMOTO, T., GAÑÁN, C., AND VAN EETEN, M. Disposable botnets: Examining the anatomy of iot botnet infrastructure. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (New York, NY, USA, 2020), ARES '20, Association for Computing Machinery.

[60] THOMAS, R., AND PAVITHRAN, D. A survey of intrusion detection models based on nsl-kdd data set. In *2018 Fifth HCT Information Technology Trends (ITT)* (2018), pp. 286–291.

[61] TIELEMAN, T., HINTON, G., ET AL. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning 4*, 2 (2012), 26–31.

[62] VALDOVINOS, I. A., PÉREZ-DÍAZ, J. A., CHOO, K.-K. R., AND BOTERO, J. F. Emerging ddos attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. *Journal of Network and Computer Applications 187* (2021), 103093.

[63] YUNGAICELA-NAULA, N. M., VARGAS-ROSALES, C., AND PEREZ-DIAZ, J. A. Sdn-based architecture for transport and application layer ddos attack detection by using machine and deep learning. *IEEE Access* (2021), 1–1.

[64] ZHANG, C., CAI, Z., CHEN, W., LUO, X., AND YIN, J. Flow level detection and filtering of low-rate ddos. *Computer Networks 56*, 15 (2012), 3417–3431.

[65] ZHANG, Y., XU, J., WANG, Z., GENG, R., CHOO, K., PEREZ-DIAZ, J., AND ZHU, D. Efficient and intelligent attack detection in software defined iot networks. In *2020 IEEE International Conference on Embedded Software and Systems (ICESS)* (Los Alamitos, CA, USA, dec 2020), IEEE Computer Society, pp. 1–9.

[66] Zhijun, W., Wenjing, L., Liang, L., and Meng, Y. Low-rate dos attacks, detection, defense, and challenges: A survey. *IEEE Access 8* (2020), 43920–43943.

[67] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE 109*, 1 (2021), 43–76.

Curriculum Vitae

Josué Genaro Almaraz Rivera was born in Nuevo León, México, on January 27, 2000. He received the B.S. degree in Computer Science from Universidad Autónoma de Nuevo León in 2020, and in summer of the same year he was accepted in the M.S. degree in Computer Science at Tecnológico de Monterrey, Monterrey Campus. He expects to receive this degree in mid-2022.

He has worked at companies like Apple and Meta. Recently, in summer 2021, he interned as Data Engineer in WhatsApp. Also, he was TEDx speaker in two events at Universidad Autónoma de Nuevo León, during 2017 and 2018, with the Social Hacking and Artificial Intelligence topics.

His current research interests include the development of new Machine Learning and Deep Learning methods for anomaly detection, and the application of Artificial Intelligence models in cybersecurity.

This document was typed in using LaTeX 2_ε[1] by Josué Genaro Almaraz Rivera.

[1]The template `MCCi-DCC-Thesis.cls` used to set up this document was prepared by the Research Group with Strategic Focus in Intelligent Systems of Tecnológico de Monterrey, Monterrey Campus.