electronics

MDPI

Article

Towards Near-Real-Time Intrusion Detection for IoT Devices using Supervised Learning and Apache Spark

Valerio Morfino and Salvatore Rampone *

Department of Law, Economics, Management and Quantitative Methods (DEMM), University of Sannio, I-82100 Benevento, Italy; valerio.morfino@ctcgroup.it

* Correspondence: rampone@unisannio.it

Received: 2 January 2020; Accepted: 5 March 2020; Published: 6 March 2020

Abstract: In the fields of Internet of Things (IoT) infrastructures, attack and anomaly detection are rising concerns. With the increased use of IoT infrastructure in every domain, threats and attacks in these infrastructures are also growing proportionally. In this paper the performances of several machine learning algorithms in identifying cyber-attacks (namely SYN-DOS attacks) to IoT systems are compared both in terms of application performances, and in training/application times. We use supervised machine learning algorithms included in the MLlib library of Apache Spark, a fast and general engine for big data processing. We show the implementation details and the performance of those algorithms on public datasets using a training set of up to 2 million instances. We adopt a Cloud environment, emphasizing the importance of the scalability and of the elasticity of use. Results show that all the Spark algorithms used result in a very good identification accuracy (>99%). Overall, one of them, Random Forest, achieves an accuracy of 1. We also report a very short training time (23.22 sec for Decision Tree with 2 million rows). The experiments also show a very low application time (0.13 sec for over than 600,000 instances for Random Forest) using Apache Spark in the Cloud. Furthermore, the explicit model generated by Random Forest is very easy-to-implement using high- or low-level programming languages. In light of the results obtained, both in terms of computation times and identification performance, a hybrid approach for the detection of SYN-DOS cyber-attacks on IoT devices is proposed: the application of an explicit Random Forest model, implemented directly on the IoT device, along with a second level analysis (training) performed in the Cloud.

Keywords: IoT; cyber-attacks; SYN-DOS; supervised machine learning; Apache Spark; MLlib; cloud environment; hybrid approach

1. Introduction

In recent years, a significant spread of Internet of Things (IoT) devices has been noted. Gartner estimates that the IoT will reach 26 billion units by 2020 [1,2] and a study by Statista reveals that this number will become 75.44 billion worldwide by 2025 [3].

The use of these devices is growing more and more in several application such as mobile health, Internet of Vehicles, smart home, industrial control, and environmental monitoring, extending the scope of mobile communications from interpersonal communications to smart interconnection between things and people, but also between things and things [4]. These devices are more and more often part of everyday life of billions of people, just think to Smart-Tv, smartwatch, IP cameras. Thus, these devices interact with people through the use of sensors and actuators, can open doors, monitor houses, record the heartbeat. But these devices are, almost always, connected to the Internet. So, they are sensitive to cyber-attacks.

Thus, on one hand the IoT devices improves the productivity of companies and enhances the quality of people's lives, but on the other hand the IoT will increase the potential attack surfaces for

cyber criminals [1]. A study by Hewlett Packard revealed that 70% of the most commonly used IoT devices contain serious vulnerabilities [5]. IoT devices have vulnerabilities due to lack of transport encryption, insecure Web interfaces, inadequate software protection, and insufficient authorization. On average, each device contains 25 holes, or risks of compromising the home network [1].

Identifying the number of cyber-attacks on IoT devices and their economic impact is a challenging issue, given the continuous and high-tech changes [6]. However, a survey by Irdeto Global Connected Industries Cybersecurity Survey revealed that cyberattacks targeted at IoT devices could cost the U.S. economy $8.8 billion per year. The research highlight that IoT-focused cyberattacks are alarmingly widespread (80% of interviewed claimed to have experienced an attack in the past 12 months). About 55% of the attacks have caused operational downtime as a result [7]. Gartner forecasts that worldwide spending on IoT security will reach 2,4 billion in 2020 and 3,1 billion in 2021 [8].

On the general Web scenario, according to Cisco Cybersecurity Reports, even though global Web traffic enhances security using encryption techniques, 42% of organizations have been faced a DDoS attack [9]. Namely the SYN-DOS attack is one of the most popular DDoS attack type, widely used because SYN packets are not likely to be rejected by default [10,11]. So, in this work we focus on the SYN-DOS attack, a type of attack that undermines the availability of the network interfaces of devices, exploiting the normal functioning of the TCP/IP protocol.

Machine learning has proven to be very important and effective in identifying and protecting against cyber-attacks [12–16], also specifically for DDOS attacks [17]. For example a 97.4% of identification success for real traffic data has been obtained by D'Angelo et al. [18] using U-BRAIN [19]. In the specific field of IoT devices anomaly and attacks detection Hasan et al. obtained up to 99.4% of identification success using Decision Tree, Random Forest, and Artificial Neural Networks [20].

The application limits of state-of-the-art machine learning algorithms are mostly related to the computational requirements needed for large datasets [21]. This is especially important for IoT devices, which have generally reduced processing capabilities. However, they are often connected or otherwise connectable to the Internet, therefore it is possible to use an approach based on technologies operating in the Cloud environment.

In this work we have a dual purpose:

- to value the performances of several machine learning algorithms in identifying SYN-DOS attacks to IoT systems in a Cloud environment, both in terms of application performances, and in training/application times. Namely, we use several general-purpose machine learning algorithms included in the MLlib library of Apache Spark [22], one of the most interesting and used technologies in the big data field, available with an open source license and present in the cloud computing facilities of the main world players [23].
- by using the previous results, to propose a strategy for the sustainable implementation of machine learning algorithms for the detection of SYN-DOS cyber-attacks on IOT devices. Our purpose is to create a hybrid architecture that realizes the training of machine learning models for protecting against DDOS attacks on the cloud and the application of the obtained models directly on the IOT devices.

While there are several application of machine learning algorithms against cyber-attack in a Cloud environment [24–29] or also in a local one, such as Kitsune [30], it seems there is no specific integrated application for IoT.

The remaining of this paper is organized as follows. In Section 2, after a brief introduction to the SYN-DOS attack, we introduce the used datasets, the Apache Spark framework and the MLLIB Spark library for Machine learning. In Section 3, we describe the selected cloud environment, the used datasets, the measured parameters and the experimental results. In the last section, we summarize the results and discuss the work. Some details are reported in the Appendix A.

2. Materials and Methods

2.1. Brief Description of a SYN-DOS Attack

The SYN-DOS (or TCP SYN-DOS or SYN flood) attack, is a type of Distributed Denial of Service (DDoS) attack that exploits the normal three-way handshake of the Transmission Control Protocol (TCP), and can be used to make server processes incapable of answering a legitimate client application's requests for new TCP connections. Any service that binds to and listens on a TCP socket is potentially vulnerable to TCP SYN flooding attacks [10].

According to RFC 793, the normal mechanism of TCP three-way handshake exchanges the following sequence of packets (see Figure 1):

1. Client requests connection by sending SYN (synchronize) message to the server.
2. Server acknowledges by sending SYN-ACK (synchronize-acknowledge) message back to the client.
3. Client responds with an ACK (acknowledge) message, and the connection is established.

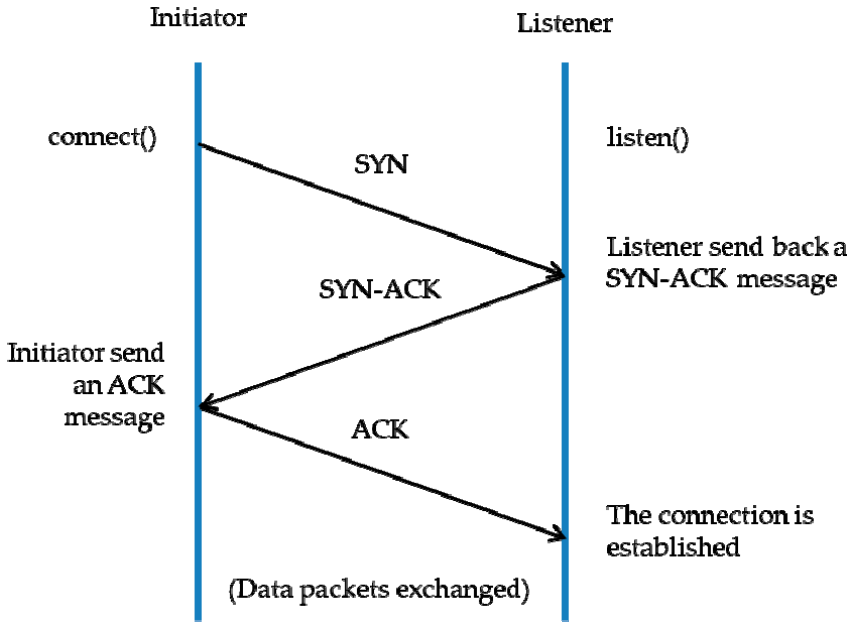

Figure 1. Normal Transmission Control Protocol (TCP) 3-Way Handshake.

In a typical SYN flood attack, a series of SYN packets to the targeted server are sent. The server is unaware of the attack, so it receives multiple, apparently legitimate requests to establish communication. Thus, it responds to each attempt with a SYN-ACK packet.

The malicious client either does not send the expected ACK, or—if the IP address is spoofed—never receives the SYN-ACK. In both cases, the server under attack will wait for an acknowledgement for some time (timeout). During this time the connection remains open. Before the connection time out, another SYN packet arrive. This behavior creates a very large number of connections half-open. Eventually, the server's connection overflow tables fill and the service to legitimate clients will be denied. Finally, the server may even malfunction or crash [31].

Some variations of the attack have been observed. A comprehensive description in presented in [10].

2.2. Attack Data

As attack data we refer to a known data collection [30] containing traffic data of IoT devices, namely surveillance video IP-cameras, assembled in a surveillance network. Several attacks that affect the availability and integrity of the video uplinks are conducted.

Specifically, the work contains 9 different datasets each one for a different kind of attack. For each of these 9 attacks, a dataset of extracted feature vectors was compiled. The features consist of statistics on network traffic which are used to implicitly describe the current state of the channel. These statistics are extracted by a Feature extractor module in the chain. For further details please refer to [30].

The full dataset contains a total of 2,771,276 instances, of which 2,764,238 contain regular traffic and 20,000 malicious traffic. Each row of the dataset has 115 features in numeric double format, describing the state of the channel.

2.3. Apache Spark

Apache Spark is a high-performance, general-purpose distributed computing system. It enables the process of large quantities of data, beyond what can fit on a single machine, with a high-level APIs, which is accessible in Java, Scala, Python and R programming languages. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

Spark allows users to write programs on a cluster computing system that can perform operations on very big amount of data in parallel. A large dataset is rep-resented using a distributed data structure called RDD — Resilient, Distributed Dataset—which is stored in a distributed way in the executors (i.e., slave nodes). The objects that comprise RDDs are called partitions. They may (but not must) be computed on different nodes of a distributed system.

Spark evaluates RDDs lazily. Thus, RDD transformations are computed only when the final RDD data needs to be computed. Spark can keep an RDD loaded in-memory on executor nodes throughout the life of a Spark application for faster access in case of repeated computations. RDDs are immutable: transforming an RDD returns a new RDD and the old one can be trashed by a garbage collector. The paradigms of lazy evaluation, in-memory storage, and immutability make Spark fault-tolerant, scalable, efficient and easy to use [23]. In more detail Spark can warrant resilience: when any node crashes in the middle of any operation, one other node has reference to the crashed one, thanks to mechanism called lineage. In case of a crash, the cluster manager assigns the job to another node, which will operate on the particular partition of the RDD and will perform the operations that it has to execute without data loss [32].

2.4. Machine learning algorithms

MLlib is Spark's machine learning (ML) library [23]. It provides tools such as machine learning algorithms for classification, regression, clustering, and collaborative filtering, and others (Featurization, Pipelines, Persistence, and Utilities).

In choosing the machine learning algorithms we consider our data are labeled (attack or not), so we face a supervised learning problem, where the expected output of the model is a binary classification. To this aim we consider the following algorithms from Apache Spark MLlib standard library: Logistic Regression (LR) [33]; Decision Tree (DT) [34]; Random Forest (RF) [35]; Gradient Boosted Tree (GBT) [36]; Linear Support Vector Machine (SVM) [37].

Since the aim of the work is to find algorithms that can be easily implemented on IoT devices, preference have been given to tree algorithms, which can be easily implemented with IF-THEN-ELSE programming structures even on devices with little processing capacity. Linear Regression and Support Vector Machine have been included in the comparison because they have been used in related works.

Logistic Regression (LR) is a linear method commonly used for classification. The method combines each of individual input (i.e., the features) with specific weight, generated during the training process, that are combined to get a probability to belonging to a particular class. The weight represents the feature importance. Thus, if a feature has a large weight can be assumed that a variation in the feature have a significant effect on the outcome [38]. In Spark.ML, logistic regression can be used to predict a binary outcome by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial logistic regression [39].

Decision Trees (DTs) are one of the most friendly and interpretable models for performing classification because they are pretty similar to decision models that humans use quite often. In the training phase, the Decision Tree creates a structure with all the inputs and follows a set of branches to make a prediction. This behavior makes this algorithm a good starting point model, because it is rather easy to reason about and easy to inspect, furthermore it makes very few assumptions on the structure of data. In other words, rather than trying to train coefficient in order to model a function, it creates a big tree of decisions to follow at prediction time. This model supports both binary and multi-class classification. The big issue of Decision Tree is that it can overfit data extremely quickly, because it creates a pathway from the start based on every single training example, even if there are some way to limit this issue, such as limiting the height of the tree [38]. In Spark.MLlib, decision trees are supported for binary and multiclass classification and for regression, using both continuous and categorical features. The implementation partitions data by rows, allowing distributed training with millions of instances [40]. This feature makes the Spark implementation able to work efficiently on very big datasets.

Random Forest (RF) and Gradient-Boosted Tree (GBT) are both extensions of Decision Tree. Rather training one tree on all data, multiple trees on varying subset of data are trained. Thus, the various decision trees will become "expert" in a specific domain. By combining these various experts, a "wisdom of crowd effect" is obtained, where the group's performance exceeds any individual. This method can also help to prevent the overfitting, a big problem for Decision Tree. Random Forest and Gradient-Boosted Tree use two different methods for combining decision trees. On the one hand in Random Forest simply several trees are trained and then average of responses are averaged to make a prediction; on the other hand, in Gradient-Boosted Tree each tree makes a weighted prediction, so some tree have more predictive power for some classes than others. The decision trees are iteratively trained in order to minimize a loss function [38].

The Spark.ML implementation supports Random Forest for binary and multiclass classification and for regression, using both continuous and categorical features. Gradient-Boosted Tree are supported for binary classification and for regression, using both continuous and categorical features. To the current version of Spark (2.4.4.), multiclass classification is not supported. Both Random Forest and Gradient-Boosted Tree of Spark.MLlib use the Decision Tree implementation, therefore the same considerations on efficiency are applied [41].

Linear Support Vector Machine (SVM) is the MLlib Spark implementation of Support Vector Machine, a class of algorithms widely used for classification and regression analysis. Given a set of training examples, each belonging to one of two class labels, an SVM algorithm builds a model that assigns new examples into one label or another. A linear SVM described finds linear boundaries in the input feature space. The SVM model resulting from the training stage is a representation of the examples belonging to the training set as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples, in the application stage, are mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. More formally, an SVM constructs a hyperplane to separate data points belonging to two class labels in feature space. A good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class. Maximizing the separability between the two classes reduces the generalization error of the classifier [42]. The current version of spark (2.4.4.) Spark.ML implementation supports only binary classification [43].

3. Experiments and Results

3.1. Execution Environment

The execution environment used is Databricks community edition [44]. It provides a just in time platform on top of Apache Spark that empowers to build and deploy advance analytic solution. It is orchestrated with open source Spark Core with underlying general execution engine which supports a

wide variety of application, Java, Scala and Python API for the ease of development. It had integrated workspace in the form of notebooks and dashboards.

Summary of Databricks processing resources (Databricks community cloud):

- 8 cores (shared)
- 6 GB RAM
- Apache Spark 2.4.0, Scala 2.11

3.2. Datasets

The datasets used in this work are subsets of the data collection defined in [22]. We focus on dataset called "SYN DoS", related to a Syn Flood attack able to create a DoS (Denial of Service).

We created a python script, based on [45], to extract 5 datasets containing different number of instances.

The datasets characteristics (Dimension, number of Training Instances, number of Testing Instances, Total number of Instances, and ratio between regular traffic instances and malicious traffic instances) are described in Table 1. Data sets are spitted using the ratio of 70% training set and 30% testing set.

Table 1. Datasets used in the experiments.

Dataset name	Dim (Gbyte)	Training Instances	Testing Instances	Total Instances	Normal/Malicious Instances
SYNDOS10K	0.0317	7698	3302	11,000	10,000/1000
SYNDOS100K	0.266	67,871	29,167	97,038	90,000/7038
SYNDOS300K	0.825	210,245	89,793	300,038	293,000/7038
SYNDOS1M	2.68	700,192	299,846	1,000,038	993,000/7038
SYNDOS2M	5.39	1,405,057	601,981	2,007,038	2,000,000/7038

Each row of the dataset has 115 features in numeric double format, describing the state of the channel and one label, containing "F" for normal packet and "T" for malicious packet.

3.3. Evaluation Parameters

For each algorithm we value the following parameters [46]:

(1) The Accuracy on the testing set (*ACC*):

$$ACC = (TP + TN)/(P + N) = (TP + TN)/(TP + TN + FP + FN)$$

where *TP* (True Positive) is the number of malicious packets correctly identified, *TN* (True Negative) is the number of normal instances correctly identified, *FP* (False Positive) are normal instances incorrectly identified, *FN* (False Negative) is the number of undetected malicious packets.

(2) The Error Rate on the testing set (*ERR*):

$$ERR = 1 - ACC$$

(3) The Total number of errors on the testing set.

(4) The Training time, i.e., the time in seconds required to train each machine learning method (Logistic Regression (LR); Decision Tree (DT); Random Forest (RF); Linear Support Vector Machine (SVM); Gradient Boosted Tree (GBT])) in the selected execution environment (3.1).

(5) The application time, in seconds, of the rule/solution obtained from each method on the testing set in the selected execution environment (see Section 3.1).

3.4. Classification Performance Evaluation

To evaluate the classification performance, the k-fold cross validation technique was used. Cross validation splits the dataset into a set of folds which are used as separate training and test datasets. In the experiments we used k = 10. Thus, 10 (training, test) dataset pair have been generated. The validation was supported by Spark.MLlib class CrossValidator that helps to automate the process and offers tools for tuning the hyperparameters [47].

3.5. Results

In Table 2 we report the Accuracy on the testing set (*ACC*) of the algorithms adopted in this study.

Table 2. Accuracy on the testing set (*ACC*) of the adopted machine learning algorithms in the MLlib Spark Environment. LR, DT, RF, GBT, and SVM stand for Linear Regression, Decision Tree, Random Forest, Gradient Boosted Tree, Linear Support Vector Machine, respectively.

ACC					
Dataset	**LR**	**DT**	**RF**	**GBT**	**SVM**
SYNDOS10K	1.0000	1.0000	1.0000	1.0000	1.0000
SYNDOS100K	1.0000	0.9998	1.0000	1.0000	0.9999
SYNDOS300K	1.0000	1.0000	1.0000	0.9999	0.9800
SYNDO1M	0.9999	0.9999	1.0000	0.9999	0.9945
SYNDOS2M	0.9973	0.9999	1.0000	0.9999	0.9999

In Table 3 we report the Error Rate on the testing set (*ERR*), and in Table 4 we report the absolute number of errors on the testing set.

Table 3. Error rate (*ERR*) on the testing set of algorithms used in the MLlib Spark Environment applied on the testing set. LR, DT, RF, GBT, and SVM stand for Linear Regression, Decision Tree, Random Forest, Gradient Boosted Tree, Linear Support Vector Machine, respectively.

ERR					
Dataset	**LR**	**DT**	**RF**	**GBT**	**SVM**
SYNDOS10K	0	0	0	0	0
SYNDOS100K	0	0.0002	0	0	0.0001
SYNDOS300K	0	0	0	0.0001	0.0200
SYNDO1M	0.0001	0.0001	0	0.0001	0.0055
SYNDOS2M	0.0027	0.0001	0	0.0001	0.0001

Table 4. Absolute number of errors of algorithms used in the MLlib Spark Environment applied on the testing set. LR, DT, RF, GBT, and SVM stand for Linear Regression, Decision Tree, Random Forest, Gradient Boosted Tree, Linear Support Vector Machine, respectively.

TOTAL ERRORS					
Dataset	**LR**	**DT**	**RF**	**GBT**	**SVM**
SYNDOS10K	0	0	0	0	0
SYNDOS100K	0	4	0	0	1
SYNDOS300K	0	0	0	2	1635
SYNDO1M	1540	2	0	1	1646
SYNDOS2M	1601	3	0	1	1655

The training time reported in Table 5 is referred to training set, meanwhile the application time, reported in Table 6, is referred to the inference on the test set.

Table 5. Training time (in seconds) of the adopted machine learning algorithms on Databriks community cloud environment. LR, DT, RF, GBT, and SVM stand for Linear Regression, Decision Tree, Random Forest, Gradient Boosted Tree, Linear Support Vector Machine, respectively.

TRAINING TIME					
Dataset	**LR**	**DT**	**RF**	**GBT**	**SVM**
SYNDOS10K	6.54	1.65	2.31	23.34	4.41
SYNDOS100K	8.09	2.65	10.27	29.15	72.46
SYNDOS300K	20.10	8.07	30.45	41.96	179.83
SYNDO1M	94.66	10.57	92.88	144.83	705.54
SYNDOS2M	118.64	23.22	215.82	212.76	1412.29

Table 6. Application time (in seconds) of the adopted machine learning algorithms on Databricks community cloud environment. LR, DT, RF, GBT, and SVM stand for Linear Regression, Decision Tree, Random Forest, Gradient Boosted Tree, Linear Support Vector Machine, respectively.

APPLICATION TIME					
Dataset	**LR**	**DT**	**RF**	**GBT**	**SVM**
SYNDOS10K	0.05	0.09	0.10	0.09	0.09
SYNDOS100K	0.05	0.09	0.10	0.09	0.09
SYNDOS300K	0.06	0.11	0.11	0.09	0.10
SYNDO1M	0.06	0.11	0.11	0.11	0.13
SYNDOS2M	0.06	0.12	0.13	0.14	0.14

The Random Forest appears to have the best performance. Figures 2 and 3 depict the training time of RF as function of the dataset dimension in GBytes and of the number of instances, respectively.

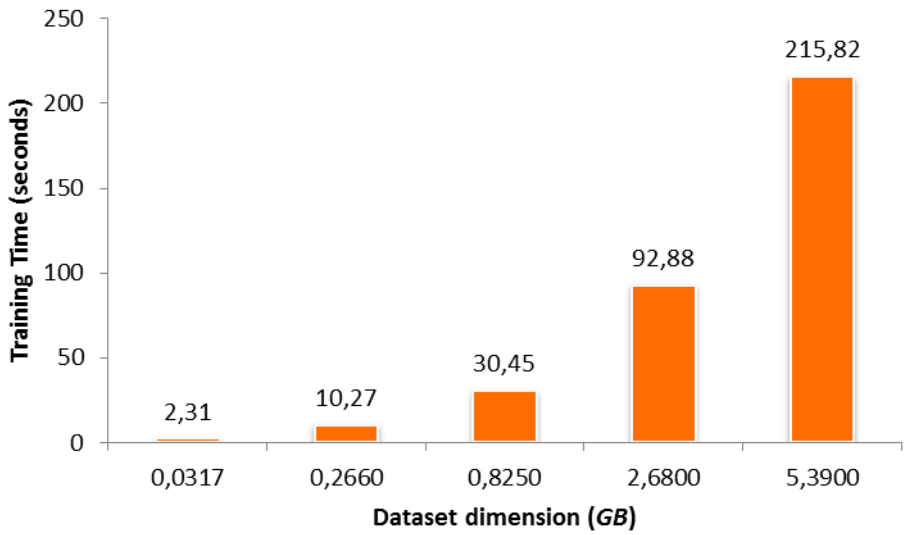

Figure 2. Training time (*y*-axis) vs. Dataset dimension (*x*-axis) for RF.

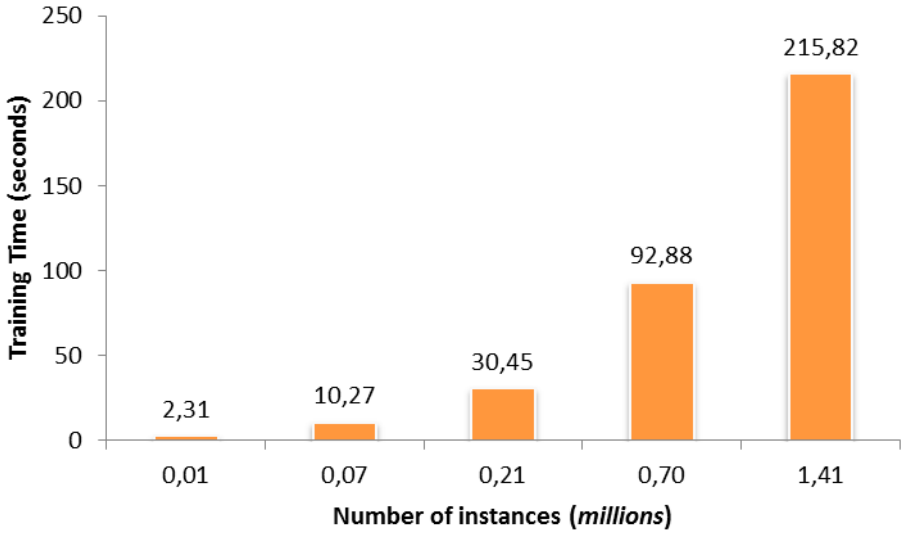

Figure 3. Training time (*y*-axis) vs. Number of instances (*x*-axis) for RF.

3.6. Hybrid Architecture

Our purpose is to create a hybrid architecture that realizes the training of machine learning models for protecting against DDOS attacks on the cloud and the application of the obtained models directly on the IOT devices.

From the previous analysis the Random Forest results to be the best performer, reporting an accuracy rate of 1. It is worth to be noted that the resulting RF model (see Appendix A) is composed of a chain of IF-THEN-ELSE instructions. This chain can be easily implemented using any high- or low-level programming language, resulting in a very high-performance run-time code, thanks to simplicity of instructions. In particular, this code could be easily implemented on any IOT device with very limited CPU and memory resources.

4. Discussion

In this work we have applied some general-purpose supervised machine learning algorithms included in the MLlib library of Apache Spark on the problem of identifying a SYN-FLOOD DDOS attack for IOT devices (Web cams). For the experiments we used 5 different datasets extracted from a public dataset. The datasets have a cardinality from 10,000 up to 2 Million of elements. A cloud environment, Databricks, has been used for training the models.

The analysis of results shows that all the algorithms of MLLib achieved a very high level of accuracy (up to 1) with both a very short training time (23.22 seconds for Decision Tree on dataset SYNDOS2M) and a minimum application time (less than of 0.14 seconds for all the algorithms). The best performing algorithm was Random Forest, which achieved an accuracy of 1 in all the experiments, a training time of 215.82 seconds with the SYNDOS2M dataset and an application time of 0.13 seconds.

These results appear consistent and improve on the results in the literature. Othman et al. in [48] tested four algorithms of Apache Spark MLlib, Support Vector Machine (SVM), Naïve Bayes, Decision Tree and Random Forest, on UNSW-NB15 dataset. Random Forest resulted the best performer with an accuracy of 97.5%.

A very similar result was archived by Belouch et al. in [49]. They evaluated the performance of four well-known classification algorithms SVM, Naïve Bayes, Decision Tree and Random Forest using Apache Spark using UNSW-NB15 dataset for network intrusion detection. The paper shows an important advantage for Random Forest classifier.

Gupta et al. [26] implemented a Spark-based intrusion detection framework with two feature selection algorithms: correlation-based feature selection and Chi-squared feature selection, based

on Spark's batch processing features. They used five Machine Learning algorithms, Logistic Regression, SVM, Naïve Bayes, Random Forest and GB Tree, on NSL-KDD and DARPA 1999 dataset. The best performing algorithm results Random Forest, meanwhile the fastest in the application phase Naïve Bayes.

Furthermore, Random Forest and Decision Tree generate explicit models consisting of a chain of simple IF-THEN-ELSE statements. These conditions can be easily implemented on IoT devices, even if they have limited memory and CPU resources.

The short training times in a cloud environment and the possibility of applying the inferred rules directly on the IoT device thanks to a simple and fast code implementation, leads us to propose a novel approach to SYN-DOS attacks mitigation, creating an architecture that includes training and retraining of machine learning models on the Cloud and the application of the resulting models for protecting against DDOS attacks directly on the IOT devices, leveraging the simple implementation of the Random Forest algorithm on low resources IOT devices.

This kind of approach seems to be supported by a recent report [50] evidencing that the major cloud service vendors have IoT services, that exchange protocols are consolidated and that attention to security is increased.

We currently plan to define a Cloud-based hybrid architecture in a more general context, extending the experiments to other types of attacks, and this will be the subject of future work.

Author Contributions: Conceptualization, V.M. and S.R.; methodology, V.M. and S.R.; software, V.M.; validation, S.R.; resources, V.M.; data curation, V.M.; writing—original draft preparation, V.M.; writing—review and editing, S.R.; visualization, V.M.; supervision, S.R.; project administration, S.R.; funding acquisition, S.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Università del Sannio—POR CAMPANIA FESR 2014/2020—"Distretti ad Alta Tecnologia, Aggregazioni e Laboratori Pubblico Privati per il rafforzamento del potenziale scientifico e tecnologico della Regione Campania"—Distretto Aerospaziale della Campania (DAC)S.C.A.R.L.—in the framework of the TABASCO project B43D18000220007.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

The explicit model resulted by the training of a Random Forest looks like:

Tree 0 (weight 1.0):
If (feature 70 <= 1698312.4645730546)
If (feature 31 <= 740.337133167518)
Predict: 0.0
Else (feature 31 > 740.337133167518)
If (feature 58 <= 68.39295337141556)
Predict: 0.0
Else (feature 58 > 68.39295337141556)
Predict: 1.0
Else (feature 70 > 1698312.4645730546)
Predict: 0.0
Tree 1 (weight 1.0):
If (feature 64 <= -4.608276837544893E-8)
If (feature 75 <= 322414.471725536)
Predict: 1.0
Else (feature 75 > 322414.471725536)
Predict: 0.0
Else (feature 64 > -4.608276837544893E-8)
Predict: 0.0

...

For brevity only two branches of the tree have been reported.

References

1. Lee, I.; Lee, K. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Bus. Horizons* **2015**, *58*, 431–440. [CrossRef]
2. Gartner Says the Internet of Things Will Transform the Data Center. Available online: https://www.gartner.com/en/newsroom/press-releases/2014-05-01-gartner-says-iot-security-requirements-will-reshape-and-expand-over-half-of-global-enterprise-it-security-programs-by-2020 (accessed on 19 December 2019).
3. Internet of Things (Iot) Connected Devices Installed Base Worldwide From 2015 to 2025 (In Billions). Available online: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (accessed on 19 December 2019).
4. Liu, G.; Jiang, D. 5G: Vision and Requirements for Mobile Communication System towards Year 2020. *Chin. J. Eng.* **2016**, *2016*, 5974586. [CrossRef]
5. HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack. Available online: http://www8.hp.com/us/en/hp-news/press-release.html?id=1744676#.VOTykPnF-ok (accessed on 10 November 2019).
6. Radanliev, P.; De Roure, C.; Cannady, S.; Montalvo, R.M.; Nicolescu, R.; Huth, M. Economic impact of IoT cyber risk-analysing past and present to predict the future developments in IoT risk analysis and IoT cyber insurance. In *Living in the Internet of Things: Cybersecurity of the IoT*; Institution of Engineering and Technology: London, UK, 2018. [CrossRef]
7. Irdeto Global Connected Industries Cybersecurity Survey-Full Report. Available online: https://go.irdeto.com/thank-you-download-connected-industries-survey-report/ (accessed on 10 November 2019).
8. Gartner Says Worldwide IoT Security Spending Will Reach $1.5 Billion in 2018. Available online: https://www.gartner.com/en/newsroom/press-releases/2018-03-21-gartner-says-worldwide-iot-security-spending-will-reach-1-point-5-billion-in-2018 (accessed on 13 September 2019).
9. Cisco Cybersecurity Reports. 2018. Available online: https://www.cisco.com/c/en/us/products/security/security-reports.html (accessed on 24 August 2019).
10. Defenses Against TCP SYN Flooding Attacks-The Internet Protocol Journal-Volume 9, Number 4. Available online: https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-34/syn-flooding-attacks.html (accessed on 24 August 2019).
11. Ngo, D.-M.; Pham-Quoc, C.; Thinh, T.N. An Efficient High-Throughput and Low-Latency SYN Flood Defender for High-Speed Networks. *Secur. Commun. Networks* **2018**, *2018*, 9562801. [CrossRef]
12. García-Teodoro, P.; Diaz-Verdejo, J.; Maciá-Fernández, G.; Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.* **2009**, *28*, 18–28. [CrossRef]
13. Harjinder, K.; Gurpreet, S.; Jaspreet, M. A review of machine learning based anomaly detection techniques. *arXiv* **2013**, arXiv:1307.7286.
14. Buczak, A.L.; Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 1153–1176. [CrossRef]
15. Xiao, L.; Wan, X.; Lu, X.; Zhang, Y.; Wu, D. IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security? *IEEE Signal Process. Mag.* **2018**, *35*, 41–49. [CrossRef]
16. Machaka, P.; Nelwamondo, F. Data mining techniques for distributed denial of service attacks detection in the internet of things: A research survey. In *Securing the Internet of Things: Concepts, Methodologies, Tools, and Applications*; IGI Global: Hershey, PA, USA, 2020; pp. 561–608.
17. Nooribakhsh, M.; MollaMotalebi, M. A review on statistical approaches for anomaly detection in DDoS attacks. *Inf. Secur. J.* **2020**, *29*, 118–133. [CrossRef]
18. D'Angelo, G.; Palmieri, F.; Ficco, M.; Rampone, S. An uncertainty-managing batch relevance-based approach to network anomaly detection. *Appl. Soft Comput.* **2015**, *36*, 408–418. [CrossRef]
19. Rampone, S.; Russo, C. A fuzzified BRAIN algorithm for learning DNF from incomplete data. *Electron. J. Appl. Stat. Anal. (EJASA)* **2012**, *5*, 256–270.

20. Hasan, M.; Islam, M.M.; Zarif, M.I.I.; Hashem, M.M.A. Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. *Internet Things* **2019**, *7*. [CrossRef]
21. L'Heureux, A.; Grolinger, K.; Elyamany, H.F.; Capretz, M.A.M. Machine learning With Big Data: Challenges and approaches. *IEEE Access* **2017**, *5*, 7776–7797. [CrossRef]
22. Apache Spark Home Page. Available online: http://spark.apache.org/ (accessed on 13 December 2019).
23. Zaharia, M.; Xin, R.S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Ghodsi, A. Apache spark: A unified engine for big data processing. *Commun. ACM* **2016**, *59*, 56–65. [CrossRef]
24. Chang, T.Y.; Hsieh, C.J. Detection and Analysis of Distributed Denial-of-service in Internet of Things—Employing Artificial Neural Network and Apache Spark Platform. *Sens. Mater.* **2018**, *30*, 857–867.
25. Pallaprolu, S.C.; Sankineni, R.; Thevar, M.; Karabatis, G.; Wang, J. Zero-day attack identification in streaming data using semantics and Spark. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 121–128.
26. Gupta, G.P.; Kulariya, M. A Framework for Fast and Efficient Cyber Security Network Intrusion Detection Using Apache Spark. *Procedia Comput. Sci.* **2016**, *93*, 824–831. [CrossRef]
27. Hafsa, M.; Jemili, F. Comparative Study between Big Data Analysis Techniques in Intrusion Detection. *Big Data Cogn. Comput.* **2018**, *3*, 1. [CrossRef]
28. Manzoor, M.A.; Morgan, Y. Real-time support vector machine based network intrusion detection system using Apache Storm. In Proceedings of the 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 13–15 October 2016; pp. 1–5.
29. Abusitta, A.; Bellaiche, M.; Dagenais, M.; Halabi, T. A deep learning approach for proactive multi-cloud cooperative intrusion detection system. *Futur. Gener. Comput. Syst.* **2019**, *98*, 308–318. [CrossRef]
30. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *arXiv* **2018**, arXiv:1802.09089.
31. What is a SYN Flood Attack. Available online: https://www.imperva.com/learn/application-security/syn-flood/ (accessed on 24 August 2019).
32. Karau, H.; Warren, R. *High Performance Spark: Best Practices for Scaling and Optimizing Apache Spark*, 1st ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2017; pp. 10–19.
33. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.
34. Safavian, S.; Landgrebe, D. A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man Cybern.* **1991**, *21*, 660–674. [CrossRef]
35. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
36. Scholkopf, B.; Smola, A.J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press: Cambridge, MA, USA, 2001.
37. Friedman, H.J. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378. [CrossRef]
38. Chambers, B.; Zaharia, M. *Spark: The Definitive Guide: Big Data Processing Made Simple*; O'Reilly Media, Inc.: New York, NY, USA, 2018.
39. Apache Spark, Classification and Regression. Available online: https://spark.apache.org/docs/latest/ml-classification-regression.html (accessed on 22 February 2019).
40. Apache Spark, Decision Tree. Available online: https://spark.apache.org/docs/latest/mllib-decision-tree.html (accessed on 22 February 2019).
41. Apache Spark, Ensembles-RDD-based API. Available online: https://spark.apache.org/docs/latest/mllib-ensembles.html (accessed on 22 February 2019).
42. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 417–421.
43. Apache Spark, MLlib-Linear Methods. Available online: https://spark.apache.org/docs/1.0.0/mllib-linear-methods.html (accessed on 22 February 2019).
44. Databricks Home Page. Available online: https://databricks.com/ (accessed on 18 December 2019).
45. Yahalom, R.; Steren, A.; Nameri, Y.; Roytman, M. Small versions of the extracted features datasets for 9 attacks on IP camera and IoT networks generated by Mirskey et al. (2018). *Mendeley Data v1* **2018**. [CrossRef]
46. Mitchell, T.M. *Machine Learning*, 1st ed.; McGraw-Hill, Inc.: New York, NY, USA, 1997.
47. Apache Spark, ML Tuning: Model Selection and Hyperparameter Tuning. Available online: https://spark.apache.org/docs/latest/ml-tuning.html (accessed on 22 February 2019).

48. Othman, S.M.; Ba-Alwi, F.M.; Alsohybe, N.T.; Al-Hashida, A.Y. Intrusion detection model using machine learning algorithm on Big Data environment. *J. Big Data* **2018**, *5*, 34. [CrossRef]
49. Belouch, M.; El Hadaj, S.; Idhammad, M. Performance evaluation of intrusion detection based on machine learning using Apache Spark. *Procedia Comput. Sci.* **2018**, *127*, 1–6. [CrossRef]
50. Key Trends from the Iot Developer Survey. 2018. Available online: https://blog.benjamin-cabe.com/2018/04/17/key-trends-iot-developer-survey-2018 (accessed on 20 February 2019).