IoT Attack Detection with Deep Learning Analysis

Riccardo Pecori
Dept. of Engineering
University of Sannio, Benevento, Italy
rpecori@unisannio.it

Amin Tayebi, Armando Vannucci, Luca Veltri
Dept. of Engineering and Architecture
University of Parma, Parma, Italy
amin.tayebi@unipr.it, luca.veltri@unipr.it

***Abstract*—Internet traffic detection and classification has been thoroughly studied in the last decade, but this is still a hot topic as regards the Internet of Things (IoT), a communication paradigm that is going to involve different aspects of our daily life. As a consequence, researchers started applying traditional methods for traffic classification also to the traffic flows coming and addressed to smart devices. In this paper, we created a large integrated dataset of IoT traffic flows, coming from four different network scenarios, in order to have a benchmark for future research. Moreover, we used this dataset to test the effectiveness of a deep learning network model, made of different hidden layers, and we compare its outcomes with the ones obtained through traditional machine learning approaches, demonstrating the superiority of our deep learning architecture in both a binary and multinomial classification.**

***Index Terms*—Internet of Things, Anomaly Detection, Intrusion Detection Systems, Artificial Neural Networks, Deep Learning**

I. Introduction

Nowadays, IoT smart devices are spreading in all aspects of our society, including industry, healthcare, homes, automotive, sport, P2P networks [1], entertainment, and many others [2]. However, this engagement raises serious issues since a lot of network traffic as well as many different traffic classes flow over IoT networks, e.g., those generated from industrial machineries, driverless cars, health sensors, smart homes and other critical devices. As such, the requirements of various IoT applications demand for more security and protection that, in turn, involve accurate classification of network traffic to early detect attacks and enact proper countermeasures. Indeed, given the broad utilization of IoT devices, malicious manipulations could cause deep implications on the security and the strength of the entire Internet. The cyberattack launched by the Mirai malware [3] represents a clear example of the severity caused by instrumenting zombified IoT devices (bots) to launch a larger DDoS attack, and testifies the necessity of secure authentication mechanisms [4], together with proper traffic classification techniques. Therefore, the need for early detection of IoT malicious traffic is a current hot topic, but the methods in the literature, even those based on artificial neural networks, i) do not reach yet the maximum possible accuracy, ii) are based on local ad hoc traffic datasets from one single network scenario, and iii) sometimes do not consider actual IoT traffic at all.

The main contributions of this paper are the following:

- the analysis of recent IoT datasets publicly available;
- the creation of a large dataset of IoT traffic, encompassing different types of IoT attacks, and the computation of a large set of features, to foster new analyses and researches;
- the design and implementation of a deep neural network classifier, capable to achieve very high accuracy, more than all similar contributions in the recent literature, over a dataset made of traffic from different networks.

The structure of the paper is the following: in Sec. II background about Internet traffic classification and deep neural networks is provided. In Sec. III related work about deep learning for IoT attack classification is described, while in Sec. IV both the considered feature model and neural network model are described in detail. Sec. V presents the integrated dataset and the experimental settings, while Sec. VI shows the obtained results as well as a comparison with traditional machine learning approaches. Finally, Sec. VII highlights some possible threats to the validity, while Sec. VIII seals up the paper with some conclusions.

II. Background

A. Traffic Classification

For traffic classification and anomaly detection, usually bidirectional network flows are considered. These are composed of a series of ordered packets, exchanged between two terminal points, and are uniquely identified through the following quintuple: *source IP address*, *destination IP address*, *source port*, *destination port*, *transport protocol*. Source and destination ports and addresses may be pairwise interchangeable and identify the two single main unidirectional sub-flows (from source to destination and vice versa) a flow is made of.

Internet traffic can be captured by using standard network sniffers like tcpdump[1] and Wireshark[2], or network emulators [5]. They permit one to get traffic traces, composed of various packets belonging to different sessions or flows, flowing inside private or public networks.

Historically, the main traffic classification methods can be roughly divided into three categories [6]: Session-based, Content-based and Statistical approaches. The usage of well-known ports belongs to the first category, while the exhaustive packet payload analysis is a proponent of the second category. In this paper, we focus on the third category, which exploits

[1]http://www.tcpdump.org/
[2]http://www.wireshark.org/

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

concepts of statistics, information theory as well as artificial intelligence, and usually does not require any application-level protocol information.

As concerns the inherent nature of statistical traffic classification approaches, they usually perform their tasks at two different levels:

- at a fine-grained level, to detect the particular *application protocol* that generated a certain flow [7];
- at a coarse-grained level, to identify a larger group of protocols (e.g., bulk transfer, mailing, web browsing, etc.), and not a specific protocol.

Whatever the considered granularity, all statistical classification algorithms usually consider transfer-based, time-based and protocol-based features of the packets to characterize a flow [8]. However, the description of flows in terms of numerical features may be carried out at different levels of abstraction [9]. As a matter of fact, there exist i) methods that consider packets as well as their inherent and simplest properties (e.g., size, inter-arrival time, the relative position in the flow, etc.); and ii) methods relying on aggregated statistical features of a flow or sub-flow (e.g., maximum, minimum, mean and standard deviation of the total volume in bytes, the overall duration, etc.).

In this paper, we focus on the second approach, i.e., the one that regards aggregated statistical features of the flows (and their two main sub-flows). As concerns the granularity of the classification, we will address both a binary classification (distinguishing normal traffic from abnormal one) and a more fine-grained multiclassification (identifying normal traffic and different types of malicious flows).

B. Deep Learning algorithms

Deep Learning (DL) regards a particular branch of machine learning, encompassing techniques that allow the simulation of information processing typical of biological nervous systems [10]. A DL architecture is made of a set of related layers, wherein each layer obtains different inputs from another layer and reorganize the information in a hierarchical fashion, useful to perform feature learning and pattern classification. DL algorithms are usually considered more suitable than other machine learning techniques in contexts featuring a high level of complexity (i.e., several attributes and a great number of data).

The training of a neural network has main two phases:

- *the feed-forward phase*, wherein the activation of the nodes of the network is performed from the input layer, usually containing a number of nodes equal to the number of the considered features, to the output one, usually containing a number of nodes equal to the number of classes, in classification problems. Except for the nodes in the input layer, all subsequent nodes, in the intermediate layers, represent neurons which activate their output according to a proper and ad-hoc activation function (e.g., ReLu) [11].
- *the back-propagation phase*, which allows one to improve the overall network performance by assigning to the connection between the nodes proper and updated weights, as well as bias values if necessary, with the aim of increasing the overall performance of the whole neural network.

III. Related Work

In this section, we summarize some contributions, recently published and surveyed [12], [13], regarding anomaly detection in IoT scenarios with artificial intelligence techniques.

Lopez-Martin et al. [14] proposed an unsupervised anomaly Network Intrusion Detection System (NIDS) for IoT environments, based on a Conditional Variational AutoEncoder (CVAE). Their method is unique due to its ability to carry out feature reconstruction, i.e., it can retrieve missing features from incomplete training datasets. The used dataset was a refined version of the NSL-KDD[3] one with 116 features and 5 possible labels. They proved experimentally that their work is less complex compared to other unsupervised NIDS, with better classification metrics (accuracy, precision, recall and F-measure) than well-known algorithms like random forest, linear SVM, multinomial logistic regression and multi-layer perception, both for binary and multiclassification problems.

Thing [15] analyzed wireless network threats and proposed an anomaly NIDS to detect and classify attacks in IEEE 802.11 networks, based on Stacked Auto-Encoder (SAE), a neural network built by stacking multiple layers of sparse auto-encoders with both two and three hidden layers. The author experienced different activation functions for the hidden neurons. To test his strategy, he used the AWID_CLS_R dataset generated from a lab-emulated Small Office - Home Office (SOHO) infrastructure. He achieved an overall general accuracy of 98.6688% in a 4-class classification (legitimate traffic, flooding attacks, injection attacks and impersonation attacks), but with only a 2-layer neural network.

Diro and Chilamkurti [16] applied Fog Computing principles in IoT environments to detect intrusions. In particular, they equipped edge layer devices with intelligent detection capabilities to improve efficiency and reduce the data transported to the Cloud. The authors proposed a deep learning approach to detect known and unseen intrusion attacks, but without specifying clearly the number of used layers. The distributed parallel deep learning approach gets better results in accuracy than centralized deep learning NIDS and also than shallow machine learning algorithms, but the employed machine learning algorithms are not specified. Diro et al. used the aforementioned NSL-KDD dataset, with some modifications, thus considering 123 features. They performed 4-class detection and achieved 98.27% of overall accuracy, 96.5% detection rate, as well as 2.57% of false alarms rate using the deep learning model, while the shallow machine learning classifiers achieved an accuracy of 96.75%, 93.66% of detection rate and 4.97% of false alarms rate. They also noted an increase in the overall detection accuracy while increasing the number of fog nodes from around 96% to over 99%.

[3]https://www.unb.ca/cic/datasets/nsl.html

Moustafa et al. [17] proposed an Adaboost ensemble method for intrusion detection, based on decision tree, Naive Bayes and Artificial Neural Network, to mitigate particularly botnet attacks against DNS, HTTP and MQTT protocols utilized in IoT networks. A set of 36 features, some typical of a single protocol, are extracted from two datasets, namely UNSW-NB15 and NIMS. Then, a feature selection step is performed to extract the most important ones. This step enables the reduction of the computational cost of the overall system. The ensemble achieved an overall accuracy, for the binary classification, between 98.54% and 98.97% for UNSW-NB15 dataset and between 98.29% and 98.36% for the NIMS dataset, while the performance of the artificial neural network by itself, whose number of layers is not specified, does not pass 96.27%.

In the contribution in [18] the authors used an intelligent system to maximize the recognition rate of network attacks by embedding the temporal behavior of the attacks into a Tap Delay Neural Network (TDNN) structure, a particular type of recurrent neural network. It has only 2 layers and a Tap Line or Tap Delay Line (TDL), which consists of a group of taps that orders the temporal inputs. The system has been compared with SNORT and with the MIT DARPA Intrusion Detection Evaluation system over the old DARPA 1998 dataset with a claimed 100% recognition rate for both port scan and host sweep attacks. No clues are given about the considered features.

Vinayakumar et al. [19] proposed a highly scalable and hybrid Dense Neural Network framework called scale-hybrid-IDS-AlertNet (SHIA IDS), which can be used in real-time to effectively monitor network traffic and host-level events to proactively alert possible cyber-attacks. The authors considered both a multiclassification, trying to detect each different attack, and a binary classification by combining all attacks together and labeling them as "abnormal" traffic. They tested their model in different public datasets such as CICIDS-2017, NSL-KDD, KYOTO, WSN-DS, KDDCup-99, UNSW-NB15. They evaluated deep learning architectures of up to 5 hidden layers performing also experiments with a reduced set of features and some comparisons with traditional machine learning algorithms. They achieved a best overall accuracy over all classes, in the mutliclassification experiments, of 87.3% on the CICIDS-2017 dataset and of 93.57% on the UNSW-NB15 dataset.

Finally, the recent work in [20] considered two datasets, namely ISCX-IDS-2012 and CIC-IDS-2017, to evaluate the performance of a proposed Hybrid Neural Network method. Like the contribution in [19], they tested their model in two scenarios, namely multiclassification and binary classification, and they considered different types of features of a traffic flow (sequential, statistical and environmental). The achieved overall accuracy in the binary classification is 99.57% on the CIC-IDS-2017 dataset and 99.58% on the ISCX-IDS-2012 dataset, while the values are 99.35% and 99.61%, respectively, in the multiclassification scenario.

IV. Approach

In this section, we first describe the proposed feature model and then we describe the used deep learning neural network.

A. Feature model

Similarly to the anomaly detection on traditional Internet traffic, also in case of IoT traffic we consider bi-directional network flows composed of a series of ordered packets, exchanged between two terminal points. The set of features considers all the aspects that characterize a flow as well as its two main sub-flows, one for each direction of the communication (forward, Fwd, or backward, Bwd). As a consequence a flow F_i, identified through the 5-tuple described in Sec. II, can be considered as a series of features (f_j, $j = 1...n$) mirroring an instance as follows:

$$F_i = \{f_1, f_2, f_3, \ ...f_n\} \quad i = 1...M \tag{1}$$

where n is the number of features per flow, and M the number of considered flows.

Specifically, we extracted from the considered traffic flows a set of 70 statistical features, as summarized in Table I: these features have been already successfully adopted within the traffic classification schemes discussed in [9], [21]. Table I lists, for each feature, a brief description, as well as the unit of measurement (UoM) of the feature itself (where μs and B stand for micro-seconds and bytes, respectively). The features encompass mainly size- and time-related characteristics, as well as their maximum, minimum, average and standard deviation values, but they also comprise information about the flags, the number of packets with certain characteristics or further sub-flows inside the two main sub-flows.

In order to make a flow a full instance, we had to label each flow according to the type of traffic, adding the so called *class* as the $(n+1)^{th}$ feature. In this paper we considered both a binary classification, where the flows were label only as BENIGN or ATTACK, and a multinomial classification, where we considered the particular type of attack. In particular, we considered the following attacks:

- DOS: a Denial of Service attack, performed also in a distributed fashion.
- MIRAI: an attack launched by a Mirai bot.
- MITM: a Man-in-the-Middle attack.
- SCANNING: comprising both OS scanning and service scanning attacks.

B. Deep learning model

The considered deep learning architecture was inspired by the work in [22], its main components are presented in Figure 1, and they are described in the following:

- one *Input layer*: the entry point of the network, encompassing a number of nodes equal to the number of considered features (70 as described in Subsec. IV-A);
- a *Batch Normalization layer*: useful to improve the training of the neural network, since it increases the speed of training and permits one to adopt higher learning rates as

TABLE I
THE LIST OF THE 70 CONSIDERED FEATURES USED TO CHARACTERIZE A FLOW.

Feature	Description	UoM
Flow duration	Duration of the flow in microseconds	μ s
Total Fwd Packet	Total packets in the forward direction	pck
Total Bwd packets	Total packets in the backward direction	pck
Total Length of Fwd Packet	Total size of packets in forward direction	B
Total Length of Bwd Packet	Total size of packets in backward direction	B
Fwd Packet Length Min	Minimum size of packets in forward direction	B
Fwd Packet Length Max	Maximum size of packets in forward direction	B
Fwd Packet Length Mean	Mean size of packets in forward direction	B
Fwd Packet Length Std	Standard deviation size of packets in forward direction	B
Bwd Packet Length Min	Minimum size of packets in backward direction	B
Bwd Packet Length Max	Maximum size of packets in backward direction	B
Bwd Packet Length Mean	Mean size of packets in backward direction	B
Bwd Packet Length Std	Standard deviation size of packets in backward direction	B
Flow Byte Rate	Number of flow bytes per second	B/s
Flow Packets Rate	Number of flow packets per second	pck/s
Flow IAT Mean	Mean time between two packets sent in the flow	μ s
Flow IAT Std	Standard deviation time between two packets sent in the flow	μ s
Flow IAT Max	Maximum time between two packets sent in the flow	μ s
Flow IAT Min	Minimum time between two packets sent in the flow	μ s
Fwd IAT Min	Minimum time between two packets sent in the forward direction	μ s
Fwd IAT Max	Maximum time between two packets sent in the forward direction	μ s
Fwd IAT Mean	Mean time between two packets sent in the forward direction	μ s
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction	μ s
Fwd IAT Total	Sum of all IATs in the forward direction	μ s
Bwd IAT Min	Minimum time between two packets sent in the backward direction	μ s
Bwd IAT Max	Maximum time between two packets sent in the backward direction	μ s
Bwd IAT Mean	Mean time between two packets sent in the backward direction	μ s
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction	μ s
Bwd IAT Total	Sum of all IATs in the backward direction	μ s
Fwd PSH flag	Number of times the PSH flag was set in packets traveling in the forward direction (0 for UDP)	-
Bwd PSH Flag	Number of times the PSH flag was set in packets traveling in the backward direction (0 for UDP)	-
Fwd URG Flag	Number of times the URG flag was set in packets traveling in the forward direction (0 for UDP)	-
Bwd URG Flag	Number of times the URG flag was set in packets traveling in the backward direction (0 for UDP)	-
Fwd Header Length	Total bytes used for headers in the forward direction	B
Bwd Header Length	Total bytes used for headers in the backward direction	B
Fwd Packet Rate	Number of forward packets per second	pck/s
Bwd Packets Rate	Number of backward packets per second	pck/s
Min Packet Length	Minimum length of a packet in the whole flow	B
Max Packet Length	Maximum length of a packet in the whole flow	B
Packet Length Mean	Mean length of a packet in the whole flow	B
Packet Length Std	Standard deviation length of a packet in the whole flow	B
Packet Length Variance	Variance length of a packet in the whole flow	B
FIN Flag Count	Number of packets with FIN	pck
SYN Flag Count	Number of packets with SYN	pck
RST Flag Count	Number of packets with RST	pck
PSH Flag Count	Number of packets with PUSH	pck
ACK Flag Count	Number of packets with ACK	pck
URG Flag Count	Number of packets with URG	pck
CWR Flag Count	Number of packets with CWE	pck
ECE Flag Count	Number of packets with ECE	pck
Down/Up Ratio	Download and upload ratio	-
Average Packet Size	Average size of packets in the whole flow	B
Avg Fwd Segment Size	Average size of a segment observed in the forward direction	B
Avg Bwd Segment Size	Average size of a segment observed in the backward direction	B
Subflow Fwd Packets	Average number of packets in a sub flow in the forward direction	pck
Subflow Fwd Bytes	Average number of bytes in a sub flow in the forward direction	B
Subflow Bwd Packets	Average number of packets in a sub flow in the backward direction	pck
Subflow Bwd Bytes	Average number of bytes in a sub flow in the backward direction	B
Init Win bytes forward	Total number of bytes sent in initial window in the forward direction	B
Init Win bytes backward	Total number of bytes sent in initial window in the backward direction	B
Act data pkt forward	Count of packets with at least 1 byte of TCP data payload in the forward direction	pck
min seg size forward	Minimum segment size observed in the forward direction	B
Active Min	Minimum time a flow was active before becoming idle	μ s
Active Mean	Mean time a flow was active before becoming idle	μ s
Active Max	Maximum time a flow was active before becoming idle	μ s
Active Std	Standard deviation time a flow was active before becoming idle	μ s
Idle Min	Minimum time a flow was idle before becoming active	μ s
Idle Mean	Mean time a flow was idle before becoming active	μ s
Idle Max	Maximum time a flow was idle before becoming active	μ s
Idle Std	Standard deviation time a flow was idle before becoming active	μ s

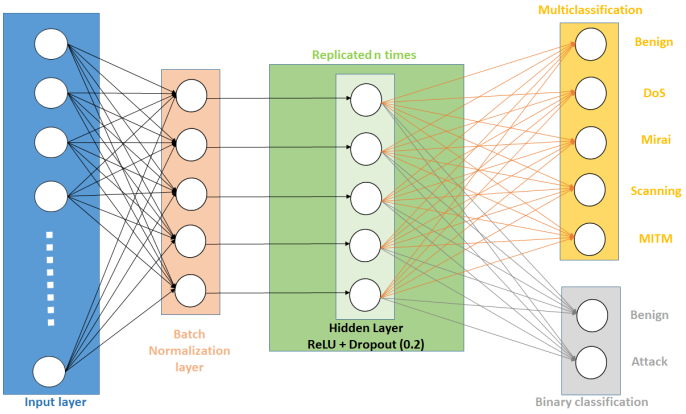

Fig. 1. The used neural network model for both the binary (gray) and multi- (orange) classification.

well as to saturate possible nonlinearities. This usually results into a higher accuracy on both validation and test, thanks to a stable gradient propagation within the network itself [23].

- a variable number of *Hidden layers*: they are made of artificial perceptrons and output a weighted sum of their inputs, passed through a $ReLu$ activation function. We made experiments with a different number of hidden layers to evaluate the differences in the overall performance.
- a *Dropout layer*: we considered this layer tightly coupled with the aforementioned one and immediately following it. As a matter of fact, in the above mentioned experiments, we replicated different times the pair hidden layer-dropout layer. The dropout layer helps to prevent overfitting by means of a regularization technique that turns off randomly several neurons in a layer according to a probability p drawn from a Bernoulli distribution. We considered the same probability for each node of the coupled hidden layer and we set its value to 0.2.
- one *Output layer*: this layer produces the final classification outcome and is composed of a number of nodes equal to the number of classes. In Figure 1, we show 2 different output layers, since they refer to the binary classification and multi-classification problem, respectively. However, when we ran our experiments we considered for each problem only one of the two depicted output layers at a time, which are a dense layer using a $softmax$ as activation function.

V. Experiment Description

In this section, we present the application of the deep neural network architecture, described in Section IV, to a large IoT dataset we constructed by merging together traffic traces coming from different sources.

First of all, we describe the four datasets we used as well as the statistical characteristics of the whole merged large dataset, secondly, we detail the parameters we used as well as the considered metrics, and finally, we report the evaluation results together with a brief discussion.

A. Dataset

The literature review, reported in Sec. III, shows that the studies about IoT attacks and traffic classification mainly employ *ad hoc* built datasets to evaluate particular and specific attacks or traffic flows. Some of the limits of these datasets are that they are i) small, so not apt for being used with deep learning techniques, ii) with a few attacks or with difficulties in separate clearly attacks from benign traffic, iii) not directly comparable and scarcely usable to evaluate and compare different approaches, and iv) usually coming from the same network scenario, where flows exhibit the same easy to learn pattern across the considered features.

Stemming from these considerations, we decided to build a large and integrated dataset by merging together four different IoT datasets, with different dimensions and different types of attacks and of traffic, in order to validate the proposed deep neural network model. The integration procedure is composed of the following steps: i) datasets selection, ii) datasets transformation; iii) datasets cleaning, and iv) datasets merging.

The selection of the datasets regarded finding out in the Web useful and recent datasets, built in the last few years and containing a sufficient number of instances regarding both benign and malicious IoT traffic.

The transformation step concerned the creation of proper CSV files, with more features than the 70 ones described in Sec. IV-A, from raw .pcap files, as well as the consistent labeling of the flow instances by exploiting the information each single dataset was endowed with. These tasks have been performed by using the CICFLOWMETER tool[4] [24], which is a Java network traffic flow generator allowing for flexibility in terms of choosing the features to compute and a control of the duration of the flow timeout.

The cleaning step regarded the removal of instances containing inconsistent values such as NaN, Infinity, and the like, and it has been performed using a proper Python script. This step involved also the reduction of the initial available features, in order to remove non-computable features or constant features throughout all the instances. This step led to have the surviving 70 features described in Sec. IV-A.

Finally, the merging step, carried out through a Python script as well, provided the integration of the considered datasets into a unique large dataset, whose statistics, together with the ones of the 4 datasets it is composed of, are summarized in Table II.

Dataset D1[5] has been released on September 2019 and built considering two typical smart home devices, i.e., SKT NUGU (NU 100) and EZVIZ Wi-Fi Camera (C2C Mini O Plus 1080P), as well as some laptops and some smartphones, connected to the same wireless network. All attacks except

[4]https://github.com/ahlashkari/CICFlowMeter

[5]https://ieee-dataport.org/open-access/iot-network-intrusion-dataset

TABLE II
STATISTICS OF THE CONSIDERED DATASETS.

Dataset ID	No. instances	Benign	DoS	MITM	Mirai	Scanning
D1	26246	496	3273	378	18623	3476
D2	794767	-	-	-	-	794767
D3	437322	437322	-	-	-	-
D4	546720	-	-	-	546720	-
Whole	1805053	437817	3273	378	565343	798243

Mirai Botnet category contain packets captured while simulating attacks using tools such as Nmap. In the case of Mirai Botnet attack, the packets were generated on a laptop and then manipulated to make it appear as if they originated from the IoT device. For dataset D2[6], created by designing a realistic network environment in the Cyber Range Lab of UNSW of Canberra, we considered only 5% of the entire dataset and only scanning attacks in order to make the whole merged dataset unbalanced and test our DL architecture in these conditions. For dataset D3[7], whose data were collected for IEEE TMC 2018 [25], we only considered benign traffic, in order to increase the number of instances of this type of traffic up to the order of magnitude of scanning attacks and, at the same time, to vary the network scenario from which traffic traces are captured. Indeed, these traces contain traffic from a great variety of IoT devices, such as Amazon Echo, Netatmo Welcome, TP-Link Day Night Cloud camera, Samsung SmartCam, etc.

Finally, dataset D4[8] contains mainly IoT malware traffic of type "Mirai" captured at the Stratosphere IPS laboratory at the Czech Technical University in 2018 and 2019, and we exploited it partly to achieve an order of magnitude of Mirai instances, in the whole dataset, equal to the one of benign traffic and scanning attacks.

The integrated whole dataset comprises a total of 1805053 flow instances, 437817 of normal traffic and 1367236 of abnormal traffic (attacks). The figures of the four considered attacks (DoS, Mirai, MITM and Scanning) are detailed in Table II. As one can see, the instances for DoS and MITM are quite small compared to the other types of traffic, and this was done purposely to test the robustness of the proposed DL model against unbalanced data, as it will be described in Section VI.

B. Evaluation Settings

The assessment has been performed on the aforementioned integrated dataset in order to identify i) benign and malicious traffic, and ii) normal traffic as well as 4 different attacks. The evaluation is performed using both the deep learning architecture described in Subsec. IV-B and by using some traditional machine learning algorithm, namely Hoeffding Tree (HT), useful for the classification of streaming data from IoT devices, and Naive Bayes (NB), with the aim to demonstrate that, for the considered IoT attack scenario, a machine learning approach is overcome by simple deep neural networks with several hidden layers. The considered deep neural network architecture has been implemented using the Python programming language, in particular Tensorflow 2.1.0[9], an open source software library for high-performance numerical computation, and Keras 2.3.1[10], a Python-based high-level neural networks API, cable to run on top of TensorFlow and to simplify the creation of an artificial neural network. The deep neural network model, described in Section IV-B, was trained by using categorical cross-entropy [26] as a loss function, and stochastic gradient descent (SGD), with learning rate equal to 0.1, momentum equal to 0.09, decay of $1e^{-6}$ for optimizing the loss function. Moreover, SGD has been integrated with Nesterov accelerated gradient (NAG) correction to avoid excessive changes in the parameter space [27]. The metrics that we used to evaluate the classification results have been the following: Precision, Recall, Accuracy and F-measure. As a matter of fact, the considered datasets is somehow unbalanced with respect to certain attack classes as it has been described in Subsec. V-A, so only the overall accuracy would have not been a significant parameter. Precision has been evaluated as the proportion of samples that truly belong to a given attack (or normal flow) among all those which were assigned to it. It is computed as the ratio of the number of relevant detected samples (true positive) to the sum of irrelevant detected samples (false positives) and relevant detected samples (true positives):

$$Precision = \frac{true\,positives}{true\,positives + false\,positives}. \quad (2)$$

On the other hand, the recall has been evaluated as the proportion of samples assigned to a given attack (or normal flow), among all the samples that truly belong to the attack (or normal traffic) itself. It is computed as the ratio of the number of relevant detected samples (true positive) to the total number of relevant samples (the sum of true positives and false negatives):

$$Recall = \frac{true\,positives}{true\,positives + false\,negatives}. \quad (3)$$

The F-measure, or F-score, is the weighted harmonic mean of precision and recall, and is computed according to the following formula:

$$F - score = 2\frac{PR}{P+R}, \quad (4)$$

where P and R are precision and recall, respectively. While precision and recall can be computed both per class and on average, the accuracy is an overall metric and has been computed as the ratio of the sum of true positives and true negatives to the total number of samples:

$$Accuracy = \frac{tp+tn}{tp+fn+tn+fp}, \quad (5)$$

[6]https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php

[7]https://iotanalytics.unsw.edu.au/iottraces.html

[8]https://www.stratosphereips.org/datasets-iot

[9]https://www.tensorlow.org/

[10]https://keras.io/

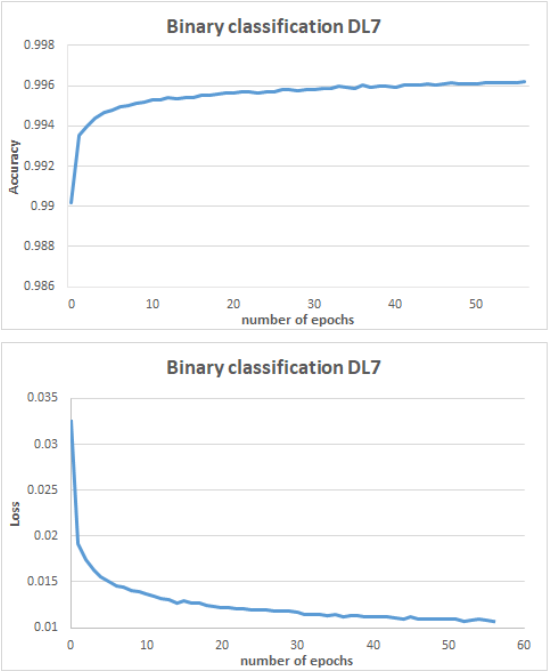

Fig. 2. Accuracy and loss trends for the binary classification and DL7.

where tp means true positives, tn means true negatives, fn means false negatives, and fp means false positives.

The experiments have been run on an Intel Core i7 7^{th} gen machine, equipped with 1 GPU and 16GB of RAM.

VI. Results

In this section we present some of the obtained results, as regards both the training phase of the considered DL model as well as the test phase and the comparison with traditional machine learning algorithms. All the results have been obtained considering a 5-fold cross validation process.

Deep learning networks are usually trained for several epochs, a hyper-parameter defining the number of times the learning algorithm will present the entire training dataset to the network under training. To make the network reaching its best performance during the test phase, it is important to set the number of epochs to a value corresponding to the point when the network accuracy vs the numbers of epochs is not increasing anymore. The loss function is usually inversely proportional to the accuracy and when the accuracy increases, the network is learning and the loss trend is decreasing. In a symmetric way to what said about the accuracy, the network does not learn anymore when the loss function reaches a constant trend.

Figure 2 shows the trends of accuracy and loss versus an increasing number of epochs for the considered DL model with a hidden number of layers equal to 7 (DL7), the configuration reaching the best performance in terms of accuracy. As one can see, after about 50 epochs both the accuracy and the loss trends stabilize. Similarly, Figure 3 presents how accuracy and loss vary during the epochs when considering the multinomial

TABLE III
Classification results on the test set.

	Binary classification				Multiclassification			
Alg.	**Acc.**	**P**	**R**	**F**	**Acc.**	**P**	**R**	**F**
HT	0.9930	0.993	0.993	0.993	0.9736	0.974	0.974	0.974
NB	0.8723	0.877	0.872	0.859	0.6702	0.879	0.670	0.748
DL4	0.9968	0.9935	0.9935	0.935	0.9942	0.9875	0.9854	0.9864
DL5	0.9969	0.9936	0.9936	0.9936	0.9943	0.9884	0.9864	0.9874
DL6	0.9970	0.9938	0.9938	0.9938	**0.9973**	0.9886	0.9867	0.9877
DL7	**0.9975**	0.9937	0.9937	0.9937	0.9937	0.9874	0.9854	0.9864

classification and the training of the proposed DL model with 6 hidden layers (DL6), the configuration achieving the top accuracy in this type of classification. As one can see, the trend is less smooth than in the case of the binary classification, and this is possibly due to the major difficulty in separating more classes, but the number of epochs when the trend stabilizes is more or less the same.

As concerns the test phase, we report in Table III the values of overall accuracy, precision, recall and F-measure for both some traditional machine learning techniques and the proposed DL model. As stated in Subsec. V-B, we considered as machine learning techniques HT and NB and their Weka[11] implementation, while for the proposed DL model we report the results for different hidden layers, from 4 to 7. As one can see, the best accuracy is reached with a DL architecture made of 7 layers for the binary classification, while it is achieved by a DL architecture made of 6 layers for the multiclassification. In general, the DL approach outperforms the two considered traditional machine learning techniques and can reach the best results compared to the related works surveyed in Sec. III. As

[11]https://www.cs.waikato.ac.nz/ml/weka/

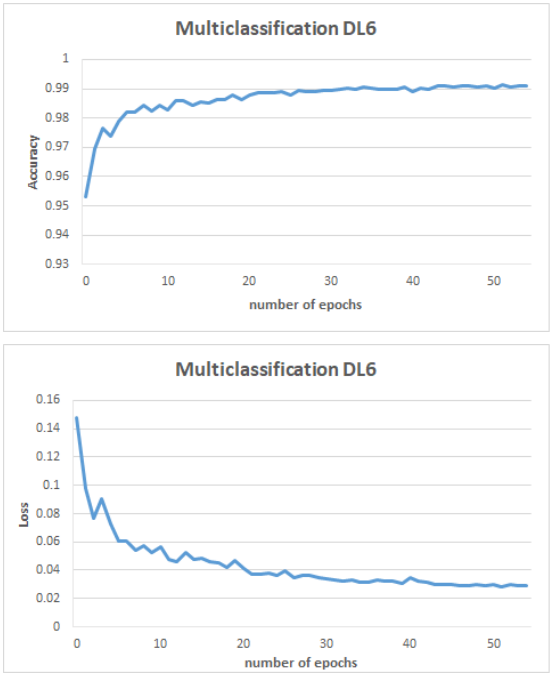

Fig. 3. Accuracy and loss trends for the multiclassification and DL6.

concerns the introduced unbalance in the dataset, we can see that for the binary classification the effect is negligible, given that the averaged precision, recall and F-measure are always greater than 0.99, while it becomes somewhat relevant in the multiclassification, even if the considered metrics are always better than those of traditional ML techniques.

VII. Threats to validity

As concerns the construct validity threats, some inaccuracies and omissions can be due to the reliability of the captured traffic traces and of the tools used to extract or compute the features. In order to limit this threat, we have considered four different datasets, from four different network scenarios, as well as a very renown and used tool like CICFLOWMETER.

Moreover, regarding the internal validity, if the adopted datasets are not correctly labeled or are obtained with a non-rigorous process, we could have classification errors. This risk is strongly mitigated because the used datasets are well documented and referenced in papers already published in reputable venues.

Finally, threats to external validity may involve the generalization of the discussed findings. We have evaluated our approach on a great number of flows from four existing datasets having different sizes and characteristics. However, in the future, it is possible to integrate more datasets with many more IoT flows.

VIII. Conclusions

In this paper, we have analyzed a large dataset of IoT benign and malicious flows which we built by integrating four different recent datasets. The analysis has been carried out using both traditional machine learning algorithms and a proper DL architecture that resulted to achieve very good results and to outperform traditional machine learning techniques, in terms of overall accuracy, precision, recall and F-measure, for both a binary and multi-classification.

In future work, we will focus on integrating much more datasets and instances to apply the proposed DL model with much more layers, and on techniques of feature selection to verify whether all the considered features are useful or not for the classification tasks we considered. A further future improvement may also regard the consideration of different types of neural networks, such as LSTM, and different architectures of the neural network itself.

References

[1] R. Pecori. A PKI-free key agreement protocol for P2P VoIP applications. In *2012 IEEE International Conference on Communications (ICC)*, pages 6748–6752, 2012.

[2] Hamid Tahaei et al. The rise of traffic classification in IoT networks: A survey. *J. Net. and Comp. Apps.*, 154:102538, 2020.

[3] G. Perrone, M. Vecchio, R. Pecori and R. Giaffreda. The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices. In *Proc. 2nd Int. Conf. on Internet of Things, Big Data and Security - Vol. 1: IoTBDS,*, pages 246–253. INSTICC, SciTePress, 2017.

[4] M. Calabretta, R. Pecori, and L. Veltri. A Token-based Protocol for Securing MQTT Communications. In *2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6, 2018.

[5] L. Veltri et al. Nemo: A flexible and highly scalable network emulator. *SoftwareX*, 10:100248, 2019.

[6] R. Pecori and L. Veltri. A statistical blind technique for recognition of Internet traffic with dependence enforcement. In *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 328–333, Aug 2014.

[7] P. Ducange, G. Mannará, F. Marcelloni, R. Pecori, and M. Vecchio. A novel approach for Internet traffic classification based on multi-objective evolutionary fuzzy classifiers. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, July 2017.

[8] José Camacho et al. A generalizable dynamic flow pairing method for traffic classification. *Computer Networks*, 57(14):2718 – 2732, 2013.

[9] Yu Wang, Yang Xiang, Jun Zhang, Wanlei Zhou, and Bailin Xie. Internet traffic clustering with side information. *Journal of Computer and System Sciences*, 80(5):1021 – 1036, 2014.

[10] Li et al. Deng. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[11] M. W. Gardner and S.R. Dorling. Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14):2627–2636, 1998.

[12] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Communications Surveys Tutorials*, 21(3):2671–2701, 2019.

[13] Elrawy, M. F. and Awad, A. I. and Hamed, H. F.A. Intrusion Detection Systems for IoT-based Smart Environments: A Survey. *J. Cloud Comput.*, 7(1):123:1–123:20, December 2018.

[14] Lopez-Martin, et al. Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT. *Sensors*, 17(9):1967, Aug 2017.

[15] V. L. L. Thing. IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, March 2017.

[16] A. A. Diro and N. Chilamkurti. Distributed attack detection scheme using deep learning approach for internet of things. *Future Generation Computer Systems*, 82:761 – 768, 2018.

[17] N. Moustafa, B. Turnbull, and K. R. Choo. An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet of Things Journal*, 6(3):4815–4830, June 2019.

[18] O. Al-Jarrah and A. Arafat. Network Intrusion Detection System using attack behavior classification. In *2014 5th Int. Conf. on Information and Communication Systems (ICICS)*, pages 1–6, April 2014.

[19] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman. Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, 7:41525–41550, 2019.

[20] C. Ma, X. Du, and L. Cao. Analysis of Multi-Types of Flow Features Based on Hybrid Neural Network for Improving Network Anomaly Detection. *IEEE Access*, 7:148363–148380, 2019.

[21] J. Zhang and X. Chen and Y. Xiang and W. Zhou and J. Wu. Robust Network Traffic Classification. *IEEE/ACM Transactions on Networking*, 23(4):1257–1270, Aug 2015.

[22] M. L. Bernardi et al. Keystroke analysis for user identification using deep neural networks. In *2019 Int. Joint Conf. on Neural Networks (IJCNN)*, pages 1–8, July 2019.

[23] Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. 32nd Int. Conf.on Machine Learning - Vol. 37*, ICML'15, pages 448–456. JMLR.org, 2015.

[24] A. H. Lashkari et al. Characterization of Tor Traffic using Time based Features. In *Proc. 3rd Int. Conf. on Information Systems Security and Privacy - Vol. 1: ICISSP,*, pages 253–262. INSTICC, SciTePress, 2017.

[25] A. Sivanathan et al. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, Aug 2019.

[26] Mannor, Shie and Peleg, Dori and Rubinstein, Reuven. The Cross Entropy Method for Classification. In *Proc. 22nd Int. Conf. on Machine Learning*, ICML '05, pages 561–568, New York, NY, USA, 2005. ACM.

[27] Sutskever, Ilya et al. On the Importance of Initialization and Momentum in Deep Learning. In *Proc. 30th Int. Conf. on Machine Learning - Vol. 28*, ICML'13, pages III–1139–III–1147. JMLR.org, 2013.