International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 11, No.1 (Jan-2022)

https://dx.doi.org/10.12785/ijcds/110117

A Lightweight Optimized Deep Learning-based Host-Intrusion Detection System Deployed on the Edge for IoT

Idriss Idrissi[1], Mostafa Azizi[1] and Omar Moussaoui[1]

[1]*MATSI Research Lab, ESTO, Mohammed First University, Oujda, Morocco*

Received 18 Apr. 2021, Revised 19 Jun. 2021, Accepted 8 Jul. 2021, Published 9 Jan. 2022

Abstract: The Internet of Things (IoT) is now present in every domain from applications in smart homes, Smart Cities, Industrial Internet of Things (IIoT), such as e-Health, and beyond. The wide use of Internet of Things is making its security a real concern. Techniques based on artificial intelligence (AI) and its subsets machine learning (ML) and deep learning (DL) are commonly used to develop a secure Intrusion Detection System (IDS) for IoT. Researchers and industrialists are commonly using commercial Internet of Things devices, broadly available on the market. In this paper, we present an analysis of the possibility to deploy a Deep Learning-Based Host-Intrusion Detection System (DL-HIDS) on some specific commercial IoT devices. We performed multiple optimizations regarding the types of our used devices to meet their limited hardware specifications. In our conducted analysis, we consider such criteria, as memory consumption and inference timing (attacks prediction timing), to conclude which model fits better to our proposed lightweight DL-HIDS for each studied device, and to anticipate about which IDS we must generate and expectedly deploy based on the characteristics of the devices we possess. The paper also discusses the proposed methodology for such deployment in a real IoT environment. The obtained results about the implementation of our DL-HIDS on different considered devices (up to 99.74% in accuracy and an inference of not more of 1μs for attacks prediction) are promising and prove that we can manage to install a suited IDS for each device, but it should be minutiously supported by a central IDS in fog or cloud layers.

Keywords: IoT, IIoT, IDS, HIDS, Deep Learning, CNN

1. Introduction

The Internet of Things (IoT) continues to hold promise for a larger and more powerful ecosystem of many devices [1]. IoT has applications in every field [2] like Smart homes. These residences use devices connected to the Internet which allow remote monitoring and management of devices and systems, such as lighting and heating. Also, Smart Cities cover from traffic management to water distribution, to waste management, urban security, and environmental monitoring. Industrial IoT (IIoT) that refers to the use of IoT in industrial production under the concept of Industry 4.0. However, complexity and requirements are much higher in IIoT than in IoT, and also in Connected Health; (e-Health, Digital health, Telehealth, or Telemedicine) where IoT wits in healthcare turned into improving healthcare as such with remote monitoring and telemonitoring as the main applications in the wider field of telemedicine especially in the outbreak of epidemics like COVID-19 [3].

With a big demand for IoT devices, and according to statistics estimations [4], IoT connected devices are more than billions of online devices today, they make more than billion dollars in profits, and projects to grow to over trillion dollars by 2026. However, these devices can create a significant security danger if not secured and managed suitably.

The COVID-19 outbreak is leading to a rush in IoT security adoption [5]. Researchers are trying to achieve a good level in for IoT security, among the most used techniques is the use of artificial intelligence (AI) and its subsets machine learning (ML) and deep learning (DL) [6][7]. While these techniques are achieving astonishing results, securing IoT devices doesn't have to be excessively complex or costly. Published papers on IoT security using artificial intelligence [6] are mostly theoretical like proposing and benchmarking Intrusion Detection Systems (IDS) models on different datasets. In this paper we experiment, an optimized DL-based IDS on some lightweight connected devices in the fog computing.

The rest of this paper is structured as follows. In the second section we describe the background, the third section present the related works, fourth section present our proposed method, the fifth section we present and discus the obtained results, and finally a conclusion as a sixth section.

E-mail address: idrissi@ump.ac.ma, azizi.mos@ump.ac.ma, o.moussaoui@ump.ac.ma

2. Background

A. Intrusion Detection System (IDS)

IDS is a software application or a device that defends a system utilizing alarms that alert a security breach, and can acts to block the attacker [8]. There are numerous IDS that can be organized into three main classes: the first is "Host-based" (H-IDS) that monitors vital operating system (OS) files; second is "Network-based" (N-IDS) that examines the network traffic; and the third is "Hybrid" which examines both files in the operating systems and traffic in the network. They can also be organized according to their techniques used into two main classes: "Signature-based" that recognize bad patterns and "Anomaly-based" that differentiate deviations; that are usually based on Machine Learning (ML) [9] or Deep Learning (DL) techniques [10].

B. Deep Learning (DL)

DL is a Machine Learning (ML) based on the artificial neural networks (ANN); it is considered as a computing system inspired by the information processing and distributed communication nodes in biological brain where the machine can learn from several data instances, letting it to classify other instances [11]. Deep learning is widely used in various research fields, it is well known by its detecting capacity for the perfect features in raw data over consecutive nonlinear transformations, with each adjustment attaining further high level of complexity and abstraction [12][13]. Some of the most known deep learning algorithms are CNN (Convolutional Neural Networks), RNN (Recurrent Neural Networks), LSTM (Long Short-Term Memory), and GRU (Gated Recurrent Unit) [14].

C. Convolutional Neural Networks (CNN or ConvNets)

CNN are a class of deep neural networks that are used in many fields [15]. They are specifically a class of neural networks that uses the convolution and the pooling layers instead of the fully connected hidden layers [16]. Contrariwise to the other machine learning algorithms CNN has the ability to learn automatically the better features and categorize the traffic [17].

Moreover, CNN can achieve better classification and learn further features with more traffic data for the reason that it shares the same convolution matrix (mask), that would decrease the number of parameters and calculation summation of training significantly (see Figure 1) [18].

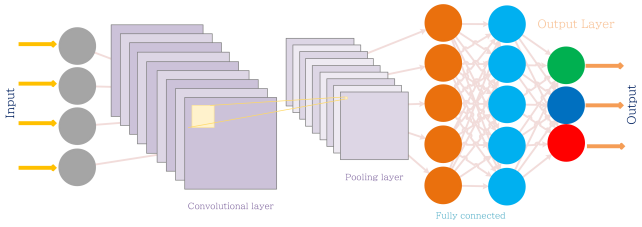

Figure 1. CNN architecture

D. Fog computing

Fog computing [19][20] is a new computing model that exists near the IoT devices and extends the cloud-based computing, storage and networking facilities. It acts as a layer intermediating between the IoT devices and the Cloud datacenters. Though, the computational nodes are distributed and heterogeneous (see Figure 2).

E. Edge computing

Edge computing [21] is a paradigm in where the data processing happens in the network edge so that computing occurs near data sources, rather than the cloud or the fog computing. An edge device is a computing or networking resource placed in between the data sources and the cloud nodes (see Figure 2) [10].

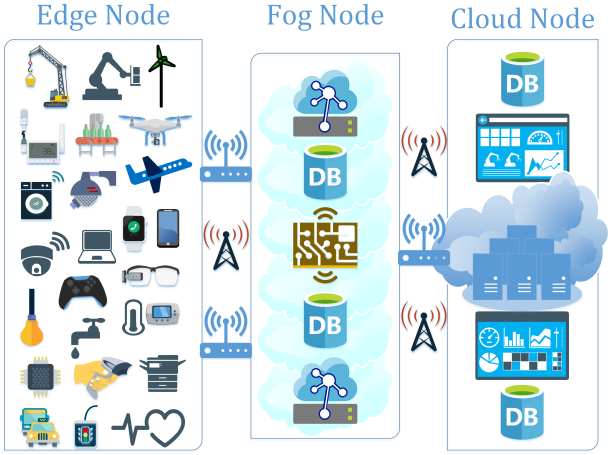

Figure 2. Edge, Fog, and Cloud nodes

F. Dataset

Dataset is a set of collected and managed data in a specific context for a specific purpose. Datasets are important ingredient for deep learning, they are base knowledge for model training. For the cybersecurity there are many datasets available some for general purposes and other specific for IoT, in our experiment we worked with the MQTT-IOT-IDS2020 dataset "uniflow features" version [22]. This dataset was generated with a simulated MQTT network architecture that contains 12 sensors, a broker, a camera (simulated), and an attacker, the dataset holds five labels, four of them are attacks; aggressive scan, UDP scan, Sparta SSH brute-force, and MQTT brute-force attack, and the fifth label is for normal traffic [23].

G. TensorFlow Lite

TensorFlow Lite [24] is Google's open-source software stack specifically for edge computing. It is a cross-platform deep learning framework that converts TensorFlow pre-trained models into an optimized format in speed and storage, that can be deployed on edge devices like Linux based embedded such as Raspberry Pi or Microcontrollers, and also on Android or iOS devices to make the inference at the Edge.

3. Proposed Method

Our proposed Deep Learning-based Host-based Intrusion Detection System will be deployed in a lightweight IoT device at the fog node (Figure 3) with much less

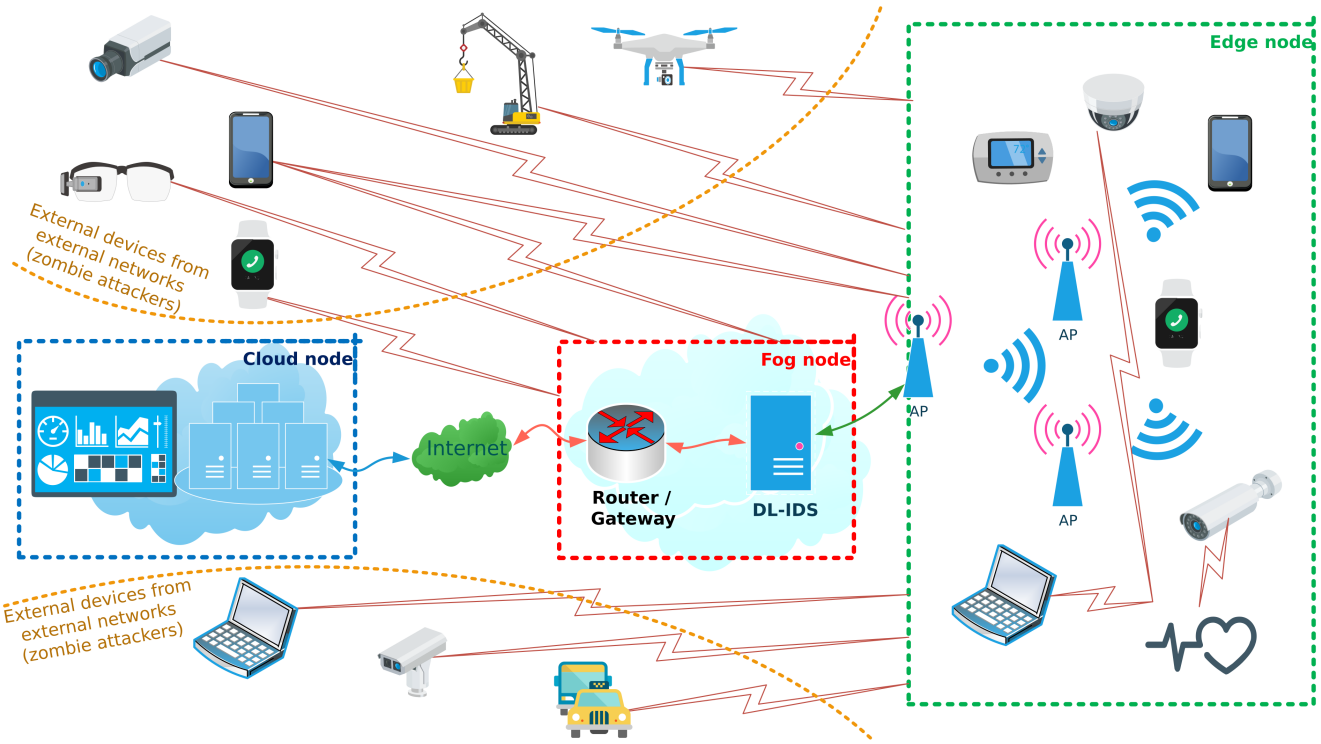

Figure 3. DL-IDS proposed deployment in real IoT environment (Fog node scenario)

computation power than classical servers or firewalls. It is planned to be placed in a real IoT environment at the edge on enabled-AI chips and preferably connected directly to a power source like cameras. At the Edge node, our DL-HIDS can have a faster inference regardless of the network connectivity, as the reasons are made on the Edge device, a data traffic transferred from and to the cloud server will be eliminated, making inferences faster, and in the security side of our DL-HIDS, there is no data leakage of any sort. The deployed model needs to take the minimum space possible, and runs on just a little memory to make a faster inference though improves the latency, but the outdraw of optimization is that there is a balance between the model size and its accuracy.

Our proposed DL-HIDS will analyze only inbound traffic, where we are assuming that this device is legit and trustworthy, otherwise, a second Network IDS "DL-NIDS" needs to be deployed on the fog node to reinforce the security on a device with more computation power preferably connected to a lightweight hardware accelerator for deep neural networks (like the Intel Neural Compute Stick 2 [25], Google Coral edge TPU [26], or Nvidia jetson nano [27]), where it can analyze the inbound traffic to/from and especially when the network is compromised from the devices both inside and outside the network in real time, inside by a lurking zombie device inside the network (see Figure 3).

The deployment of the DL-HIDS was made on five different steps using a trial-and-error cycle, by repeated, multiple attempts beginning with the best accurate model then optimized it once more until getting the best wanted results for the tested IoT board (see Figure 4).

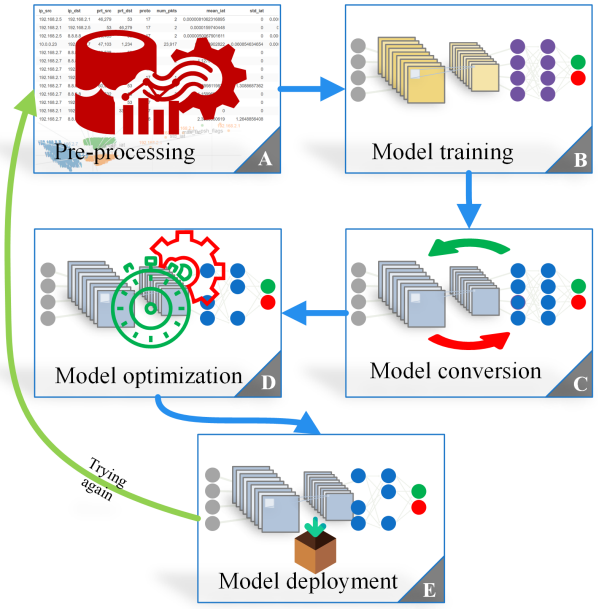

Figure 4. DL-IDS preparation and deployment steps

A. First step (Pre-processing)

First of all, we need to alter the raw data on the dataset by encoding the data, and normalize its values; meaning scaling the vectors separately to unit form, and converting the normalized output data into image data shape. Then we split the resulting data randomly into a training subset (60%), validation subset (20%), and a testing subset (20%).

Afterward, in a pre-optimization context (when needed for some devices) we make a future selection in order to decrease the number of features even though deep learning doesn't need this process but to produce a lightweight model it is a needed process, where we decreased the number of features from 16 to only 7.

In order to select the best features to build the model, we utilize the chi-square test, a statistical test that allows us to detect the relationship between the features. Chi-Square estimate whether the class label is independent of a feature. Chi-square score with "c" class and "r" values [28], as exposed in the formulas (1) and (2):

$$X^2 = -\sum_{(i=1)}^{r}\sum_{(j=1)}^{c}\frac{(n_{ij}-\mu_{ij})}{\mu_{ij}} \quad (1)$$

$$Where \qquad \mu_{ij} = \frac{(n_{*j}\, n_{i*})}{n} \quad (2)$$

n_{ij} : is the number of samples value with the i^{th} value of the feature;
n_{i*} : is the number of samples with the i^{th} the feature value;
n_{*j} : is the number of samples in class j;
n : is the number for samples.

B. Second step (Model training)

While we build our model (on a conventional machine [1]), we try to minimize the number of layers as possible without losing the depth of the deep learning model to just a shadow learning model. We build our model on CNN, initially with multiple layers and then we narrowed it down to five layers (a Convolution1D layer, MaxPooling1D layer, Flatten layer, and an output layer), and train it in 10 epochs with a batch size of 32. With these pre-optimizations; decreasing the layers number the features number, we reduce the number of model's parameters from 26,365 to just 645 parameters (see Figure 5).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 7, 32)             160
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 3, 32)             0
_________________________________________________________________
flatten (Flatten)            (None, 96)                0
_________________________________________________________________
dense (Dense)                (None, 5)                 485
=================================================================
Total params: 645
Trainable params: 645
Non-trainable params: 0
```

Figure 5. Optimized DL-HIDS Model summary

C. Third step (Model conversion)

After getting a trained model, we will convert it to a TensorFlow Lite version. TensorFlow lite model is on a special format model called Lite flat buffer file (.tflite); an efficient one in the accuracy, and a light-weight one with less space. The "tflite" model contains the architecture of the original model with its weights and biases, and the training configuration (see Figure 6) [29];

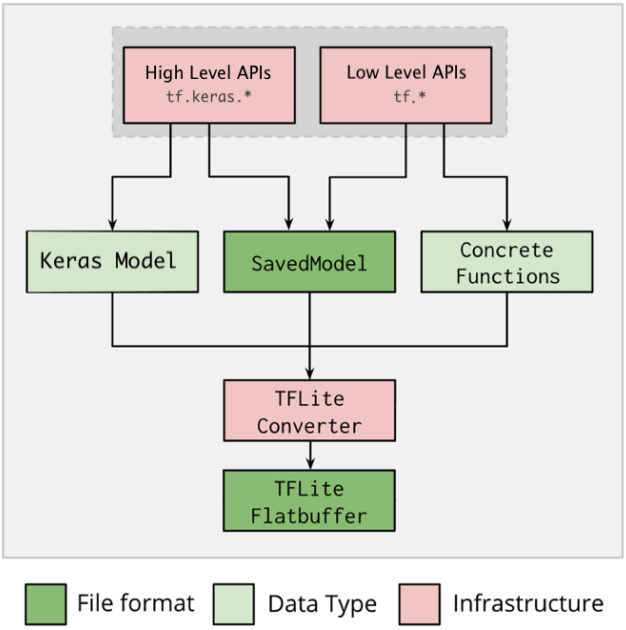

Figure 6. TensorFlow Lite converter [29]

D. Fourth step (Model Optimization)

In order to deploy a truly tiny HIDS on an Edge device with limited resources, optimization is necessary for some devices (not all of them). In this step, we worked with several optimizations [30]:

1) Weights Pruning

Here, we set separate parameters to zero to make the network thin by dropping the parameters number in the model while keeping its original architecture, and by estimating the saliency of each weight, which is defined by the alteration in the loss function upon the perturbation of the weight. We obtain the pruned model by setting the values of the dropped weights to zero and then keep it there for the rest of the process. The algorithm retrains the pruned model multiple times until it got the best results.

2) Post-training Quantization

Due to the hardware constraints of the edge device (mostly microcontrollers), post-training quantization helps improving the optimization process using a conversion technique that reduces the model size and improves the hardware accelerator latency, but with a slight decrease in the accuracy. In our experiment, we performed a "full-integer quantization"; it converts all the weights and the

activation outputs into only 8-bit integer data, because it has the compatibility with the integer only hardware devices or accelerators such as microcontrollers.

3) Weight clustering

Also called "weight sharing", it is an optimization technique that decreases a model's quantity of unique weight values, which benefits the model deployment. It works by clustering the weights of each layer into N clusters, afterwards it shares the cluster's centroid value for all the cluster's weights. The weight clustering advantage resides in the compression of the model, which is vital for the deployment on IoT objects with limited resources. But it can have some decrease in the performance for the convolution and dense layers.

E. Fifth step. (Model Deployment)

We wrote and deployed our DL-IDS model using python for the Raspberry Pi and Arduino Integrated Development Environment (IDE) for the microcontroller boards. For the model inference, we used the "TensorFlow Lite for Microcontrollers"; a library written in C++. It can be implemented into various microcontrollers with many 32-bit processors based on the ARM Cortex-M Series architecture, such as Arduino nano, Espressif ESP32, and many other microcontrollers. This library does not depend on dynamic memory allocation, therefore it necessitates to supply a memory arena "TENSOR_ARENA_SIZE" when the interpreter is created [31]. In our experimentation, we reserved an arena of "10*1024" for the ESP-32, "5*1024" for the Nodemcu v3 (the dynamic memory allocation size required depends on the used model, and it is determined by experimentation).

4. Results and Discussion

A. Hardware characteristics

IoT based boards in the market today vary from microcontrollers with kilobytes of RAM, to nanocomputers that reaches gigabytes of RAM. Deploying a standard accurate DL-IDS into one of these IoT devices is impossible. We deliberately choose the following devices to try our DL-IDS on; by the wide variety range of configurations between them (see Table I):

Raspberry Pi 3B+ / 4 [32] is a credit card-sized ARM-based single-board nano-computer, operated generally by the Raspberry Pi OS (called formerly Raspbian OS, based on the Debian OS, and optimized for the Raspberry Pi hardware).

ESP-WROOM-32 a low-cost open source IoT platform [33], from Espressif Systems. It targets a variety of applications, such as low-power sensor networks or to more difficult tasks, like voice encoding.

NodeMCU v3 is a low-cost open source IoT platform [34], that runs on the ESP8266 Wi-Fi SoC (cost-effective and highly integrated Wi-Fi MCU, with TCP/IP stack and microcontroller capability) from Espressif Systems [35].

Arduino UNO [36] is the most known, easiest, and most economical microcontroller board based on the ATmega328P, it is equipped with both analog and digital inputs/outputs that can be interfaced with various expansion boards and other circuitry.

Arduino Portenta H7 [37] is a high-performance dual-core development board, it instantaneously runs high-level code (MicroPython / JavaScript) and Artificial Intelligence (TensorFlow™ Lite) while performing low-latency operations on its customizable hardware.

In our experiments, we worked with TensorFlow Lite (2.3.0) [24] Google's open source deep learning framework for IoT devices and mobiles (on-device inference).

B. Evaluating the results

We started by deploying the original converted TensorFlow Lite model in all devices without any optimization. It did not work on all devices due to their limited capabilities. So, in order to ensure the best accuracy and memory sufficiency, we repeated the loop steps (Figure 4) until getting the suited model for each device.

Raspberry Pi 3B+/4, one of the most powerful IoT devices, runs the original and the converted original "tflite" model (26365 parameters) smoothly with its best accuracy 99.52%, and an inference of not more of 1μs for attacks prediction (see Table II), we tested the optimized models versions but we got a decrease in the accuracy concluding that the accurate suited model for such resources is the original model,

Arduino Portenta H7, also runs the original and the converted original "tflite" model (26365 parameters) smoothly with its best accuracy 99.52%, but with an inference close to 2μs for attacks prediction (see Table II).

ESP-WROOM-32, runs the original "tflite" model, but it seizes too much memory (99.74%), a more optimization was necessary (we used Weights Pruning, Post-training Quantization and Weight clustering) in order to minimize the memory consumption but with the cost of losing a fragment of the accuracy (down to 97.21%) (see Table II), and for the inference we got 2μs for attacks prediction.

NodeMCU v3, a mini version-like of the ESP32, was not able to handle the original "tflite" model nor the optimized version. Subsequently we were forced to do a pre-optimization phase by selecting just the minimal best features possible, therefore the accuracy was dropped to 96.69% (see Table II), and the inference took around 2μs for attacks prediction.

Arduino UNO; despite all attempts to deploy a working optimized model, neither the original nor the optimized models were able to run on the device due to low resources especially the RAM memory.

Figure 7 plays the role of a template to choose the suit-

TABLE I. Hardware configuration of the investigated devices

IoT Device	Manufacture	Processor	Cores	Memory	Storage	Connectivity	GPIO
Raspberry Pi 4	Raspberry Pi	Cortex-A72 (ARMv8) 64-bit SoC 1.5GHz	4	4 GB LPDDR4 SDRAM	16Gb (SD-Card)	Bluetooth:v5.0, BLE, Wifi IEEE 802.11b/g/n/ac 2,4/5,0 GHz	40 pins
Raspberry Pi 3B+	Raspberry Pi	Cortex-A53 (ARMv8) 64-bit SoC 1.4GHz	4	1GB LPDDR2 SDRAM	16Gb (SD-Card)	Bluetooth: v4.2, BLE, Wifi IEEE 802.11b/g/n/ac 2,4/5,0 GHz	40 pins
Arduino Portenta H7	Arduino	STM32H747 dual-core processor : Cortex M7,480 MHz Cortex M4,240 MHz	2	SDRAM: 8MB	Flash Memory: 16MB	Bluetooth: 5.0, BLE, WiFi IEEE 802.11 b/g/n 65Mbps option	Dual 80 pins
ESP-WROOM-32	Espressif Systems	Xtensa dual-core 32-bit LX6, 240 MHz	2	SRAM: 520 KB	Flash Memory: 4 MB	Bluetooth: v4.2, BLE, Wi-Fi IEEE 802.11 b/g/n BR/EDR	48 pins
NodeMCU V3	NodeMCU	32-bit RISC Tensilica Xtensa LX106 80 MHz	1	SRAM: 64 KB	Flash Memory: 4 MB	Wi-Fi IEEE 802.11 b/g/n	17 pins
Arduino UNO	Arduino	ATMega328 single-chip 8-bit RISC processor core	1	SRAM: 2 kB	Flash Memory: 32 kB	None (can have connectivity using external modules)	16 pins

TABLE II. Hardware configuration of the investigated devices

Model	Model's depth	Accuracy	Model's size
Original Model (OM)	7 layers, 16 Features	99.74%	343 Kb
Pre-Optimized Original Model (POM)	5 layers, 16 Features	98.75%	120 Kb
"tflite" Model (tM)	7 layers, 16 Features	99.74%	106 Kb
"tflite" Model Optimized (tMO)	5 layers, 7 Features	97.21%	4237 bytes
"tflite" Model Optimized and Pre-Optimized (tMOP)	5 layers, 7 Features	96.69%	2704 bytes

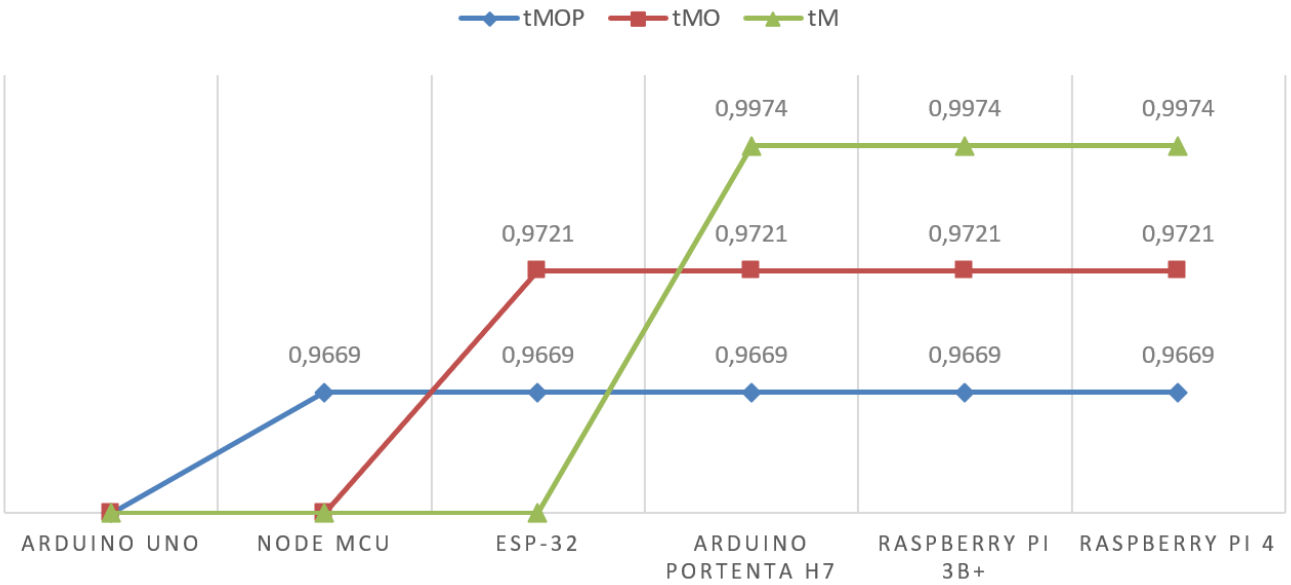

Figure 7. The suited model for each device

able and accurate model for a device based on its hardware resources. For example, if we have a device with a close resource to the NodeMCU or better, we suggest to choose the "tflite Model Optimized and Pre-Optimized"; if we get better resources than the ESP32, the best model is "tflite Model Optimized", but if we have resources close to the Arduino Portenta H7 or even higher like the Raspberry Pi 3/4; the "tflite" or the original model will work sufficiently.

The used optimization techniques were effective to achieve the right lightweight DL-HIDS model for each studied device based on its resources.

5. Conclusions

This paper presents a resourceful methodology to how prepare an appropriate lightweight DL-HIDS model specified by the resources of an IoT device using optimization methods, and deployed it in a real IoT environment. Indeed, each device has its specific architecture and we cannot generalize an IDS to all of them. The solution of this issue is to design a customized IDS for each class of similar devices. So our findings recommend to choose a suitable DL-HIDS for each typical device regarding its hardware capabilities.

From our research investigations, we conclude that deploying a generalized single DL-HIDS on any IoT device is not practical. Due to the differences between available devices on the market, ranging from small sensors to high-definition cameras, we cannot make a generalization, but we can customize this model to fit a specific device. Surely, the reduction of the customized IDS could lightly decrease some of its functionalities, but this will be balanced by the remote available fog/cloud IDS.

Acknowledgment

This work is supported by the Mohammed First University under the PARA1 Program.

References

[1] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, and H. E. Fadili, "Toward a deep learning-based intrusion detection system for IoT against botnet attacks," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 1, pp. 110–120, mar 2021.

[2] "The 10 most popular Internet of Things applications right now." [Online]. Available: https://iot-analytics.com/10-internet-of-things-applications/

[3] M. S. Rahman, N. C. Peeri, N. Shrestha, R. Zaki, U. Haque, and S. H. A. Hamid, "Defending against the Novel Coronavirus (COVID-19) outbreak: How can the Internet of Things (IoT) help to save the world?" *Health Policy and Technology*, vol. 9, no. 2, pp. 136–138, jun 2020.

[4] "Internet of Things Market Size — IoT Market Analysis, Trends, ans Forecast." [Online]. Available: https://www.verifiedmarketresearch.com/product/global-internet-of-things-iot-market-size-and-forecast-to-2026/

[5] "IoT Security Market Report 2020-2025 — IoT Platforms/Software." [Online]. Available: https://iot-analytics.com/product/iot-security-market-report-2020-2025/

[6] I. Idrissi, M. Azizi, and O. Moussaoui, "IoT security with Deep Learning-based Intrusion Detection Systems: A systematic literature review," in *4th International Conference on Intelligent Computing in Data Sciences, ICDS 2020*. Institute of Electrical and Electronics Engineers (IEEE), nov 2020, pp. 1–10.

[7] M. Boukabous and M. Azizi, "Review of Learning-Based Techniques of Sentiment Analysis for Security Purposes," in *Innovations in Smart Cities Applications Volume 4*. Springer, Cham, 2021, pp. 96–109.

[8] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Tech. Rep., 2000.

[9] M. Boukabous and M. Azizi, "Crime prediction using a hybrid sentiment analysis approach based on the bidirectional encoder representations from transformers," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 2, feb 2022.

[10] I. Idrissi, M. Azizi, and O. Moussaoui, "An Unsupervised Generative Adversarial Network Based-Host Intrusion Detection System for IoT Devices," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 2, 2022.

[11] A. Kherraki and R. E. Ouazzani, "Deep convolutional neural networks architecture for an efficient emergency vehicle classification in real-time traffic monitoring," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 1, pp. 110–120, mar 2022.

[12] S. Vieira, W. H. Pinaya, and A. Mechelli, "Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications," pp. 58–75, mar 2017.

[13] M. Berrahal and M. Azizi, "Review of DL-Based Generation Techniques of Augmented Images using Portraits Specification," in *4th International Conference on Intelligent Computing in Data Sciences, ICDS 2020*. Institute of Electrical and Electronics Engineers (IEEE), nov 2020, pp. 1–8.

[14] M. Boukabous and M. Azizi, "A comparative study of deep learning based language representation learning models," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 2, pp. 1032–1040, 2021.

[15] M. Berrahal and M. Azizi, "Augmented Binary Multi-Labeled CNN for Practical Facial Attribute Classification," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 2, pp. 973–979, aug 2021.

[16] M. Erza Aminanto and K. Kim, "Deep Learning in Intrusion Detection System: An Overview," Tech. Rep., 2016.

[17] M. Berrahal and M. Azizi, "Optimal text-to-image synthesis model for generating portrait images using generative adversarial network techniques," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 2, feb 2022.

[18] I. Idrissi, M. Azizi, and O. Moussaoui, "Accelerating the update of a DL-based IDS for IoT using deep transfer learning," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 2, pp. 1059–1067, aug 2021.

[19] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog Computing: A taxonomy, survey and future directions," in *Internet of Things*. Springer International Publishing, 2018, vol. 0, pp. 103–130.

[20] "Fog Computing and the Internet of Things: Extend the Cloud to

Where the Things Are What You Will Learn," Tech. Rep., 2015.

[21] W. Shi and S. Dustdar, "The Promise of Edge Computing," *Computer*, vol. 49, no. 5, pp. 78–81, may 2016.

[22] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "Mqtt-iot-ids2020: Mqtt internet of things intrusion detection dataset," 2020.

[23] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based iot intrusion detection system: an mqtt case study (mqtt-iot-ids2020 dataset)," in *International Networking Conference*. Springer, 2020, pp. 73–84.

[24] "TensorFlow Lite — ML for Mobile and Edge Devices." [Online]. Available: https://www.tensorflow.org/lite

[25] Intel, "Intel® Neural Compute Stick 2 — Product Sheet," Tech. Rep., 2019. [Online]. Available: www.intel.com/bench-

[26] "Coral." [Online]. Available: https://coral.ai/

[27] "NVIDIA Jetson Nano Developer Kit — NVIDIA Developer." [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[28] N. Rachburee and W. Punlumjeak, "A comparison of feature selection approach between greedy, IG-ratio, Chi-square, and mRMR in educational mining," in *Proceedings - 2015 7th International Conference on Information Technology and Electrical Engineering: Envisioning the Trend of Computer, Information and Engineering, ICITEE 2015*. Institute of Electrical and Electronics Engineers Inc., 2015, pp. 420–424.

[29] "TensorFlow Lite converter." [Online]. Available: https://www.tensorflow.org/lite/convert/index

[30] "TensorFlow model optimization — TensorFlow Model Optimization." [Online]. Available: https://www.tensorflow.org/model_optimization/guide

[31] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," oct 2020.

[32] "Raspberry Pi 4 Model B specifications – Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[33] "Espressif Systems," Tech. Rep., 2021. [Online]. Available: https://www.espressif.com/en/support/download/documents.

[34] "NodeMcu – An open-source firmware based on ESP8266 wifi-soc." [Online]. Available: https://www.nodemcu.com/index_en.html

[35] "ESP8266 Wi-Fi MCU I Espressif Systems." [Online]. Available: https://www.espressif.com/en/products/socs/esp8266

[36] "Arduino Uno Rev3 — Arduino Official Store." [Online]. Available: https://store.arduino.cc/arduino-uno-rev3

[37] "Arduino Pro." [Online]. Available: https://www.arduino.cc/pro/hardware/product/portenta-h7

Idriss Idrissi is a Ph.D. candidate in Computer Engineering at Mohammed First University in Oujda, Morocco, where he is researching internet of things security using Deep Learning. He has an M.Sc. degree in internet of things from Sidi Mohamed Ben Abdellah University in Fez, Morocco (2019), a B.Sc. degree in Computer Engineering from Mohammed First University (2016). Additionally, he holds several certifications in networking, artificial intelligence, cybersecurity, and programming. Also, he was a reviewer for various international conferences and journals. And is currently employed as an administrative at Mohammed First University.

Mostafa Azizi received a State Engineer degree in Automation and Industrial Computing from the Engineering School EMI of Rabat, Morocco in 1993, then a Master degree in Automation and Industrial Computing from the Faculty of Sciences of Oujda, Morocco in 1995, and a Ph.D. degree in Computer Science from the University of Montreal, Canada in 2001. He earned also tens of online certifications in Programming, Networking, AI, Computer Security ... He is currently a Professor at the ESTO, University Mohammed First of Oujda. His research interests include Security and Networking, AI, Software Engineering, IoT, and Embedded Systems. His research findings with his team are published in over 100 peer-reviewed communications and papers. He also served as PC member and reviewer in several international conferences and journals.

Omar Moussaoui is an Associate Professor at the Higher School of Technology (ESTO) of Mohammed First University, Oujda – Morocco. He has been a member of the Computer Science Department of ESTO since 2013. He is currently director of the MATSI research laboratory. Omar completed his Ph. D. in computer science at the University of Cergy-Pontoise France in 2006. His research interests lie in the fields of IoT, wireless networks and security. He has actively collaborated with researchers in several other computer science disciplines. He participated in several scientific & organizing committees of national and international conferences. He served as reviewer for numerous international journals. He has more than 20 publications in international journals and conferences and he has co-authored 2 book chapters. Omar is an instructor for CISCO Networking Academy on CCNA Routing & Switching and CCNA Security.