applied sciences

MDPI

Article

A Deep Learning Method for Lightweight and Cross-Device IoT Botnet Detection †

Marta Catillo ‡, Antonio Pecchia ‡ and Umberto Villano *,‡

Dipartimento di Ingegneria, Università degli Studi del Sannio, Palazzo Bosco Lucarelli C.so Garibaldi 107, 82100 Benevento, Italy; marta.catillo@unisannio.it (M.C.); antonio.pecchia@unisannio.it (A.P.)

* Correspondence: villano@unisannio.it; Tel.: +39-0824-305844

† This paper is an extended version of our paper published in Proceedings of the ACM International Conference on Availability, Reliability and Security, Vienna, Austria, 23–26 August 2022. Art. no. 90.

‡ These authors contributed equally to this work.

Abstract: Ensuring security of Internet of Things (IoT) devices in the face of threats and attacks is a primary concern. IoT plays an increasingly key role in cyber–physical systems. Many existing intrusion detection systems (IDS) proposals for the IoT leverage complex machine learning architectures, which often provide one separate model per device or per attack. These solutions are not suited to the scale and dynamism of modern IoT networks. This paper proposes a novel IoT-driven cross-device method, which allows learning a single IDS model instead of many separate models atop the traffic of different IoT devices. A semi-supervised approach is adopted due to its wider applicability for unanticipated attacks. The solution is based on an all-in-one deep autoencoder, which consists of training a single deep neural network with the normal traffic from different IoT devices. Extensive experimentation performed with a widely used benchmarking dataset indicates that the all-in-one approach achieves within 0.9994–0.9997 recall, 0.9999–1.0 precision, 0.0–0.0071 false positive rate and 0.9996–0.9998 F1 score, depending on the device. The results obtained demonstrate the validity of the proposal, which represents a lightweight and device-independent solution with considerable advantages in terms of transferability and adaptability.

Keywords: Internet of Things; deep learning; autoencoder; botnet detection; cybersecurity; cyber–physical systems

Citation: Catillo, M.; Pecchia, A.; Villano, U. A Deep Learning Method for Lightweight and Cross-Device IoT Botnet Detection. *Appl. Sci.* **2023**, *13*, 837. https://doi.org/10.3390/app13020837

Academic Editor: Agostino Forestiero

Received: 15 November 2022
Revised: 29 December 2022
Accepted: 3 January 2023
Published: 7 January 2023

1. Introduction

The Internet of Things (IoT) is intertwined with many critical assets of our daily lives, and it plays an increasingly key role in cyber–physical systems (CPSs). In fact, CPSs are becoming more and more advanced—integrating industrial IoT, Edge and Cloud computing—although their protection mainly relies on physical protection and isolation, which makes security an open topic [1]. Assuring security of IoT devices in the face of threats and attacks is a primary concern. To this aim, intrusion detection systems (IDS) are a key component in IoT security as they support online detection and response to incidents. The body of scientific literature on IDS for IoT is huge and ever-increasing. IDS for IoT is often addressed through machine learning and (deep) neural networks, e.g., [2,3]. This trend is pushed by: (i) the availability of commercial and open-source products to transform raw network packets into ready-to-use *records* suited for machine learning, (ii) the large number of public IoT datasets, such as MedBIoT (https://cs.taltech.ee/research/data/medbiot/, accessed on 3 January 2023), N-BaIoT (http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT, accessed on 3 January 2023) and IoTID20 (https://sites.google.com/view/iot-network-intrusion-dataset/home, accessed on 3 January 2023)—just to mention a few—and (iii) specialized hardware and deep learning frameworks (e.g., `Keras`, `TensorFlow` and `PyTorch`). It is a fact that IoT network traffic—transformed into fixed-length records of features—can be successfully

leveraged to recognize potential attacks, which is the primary aim of an IDS. As a result, a wide community of academics and practitioners has the chance to conduct measurement studies at the intersection of machine learning and intrusion detection for IoT.

In spite of the large availability of scientific proposals, there is a gap between "lab-made" machine learning and real-life operations. For example, highly complex deep networks proposed so far for intrusion detection, such as convolutional neural network (CNN), long short-term memory (LSTM) and cascades/ensembles of autoencoders (AE), might find no or limited adoption in production IoT environments. Another point is that recent contributions in the area use the training data to learn a separate IDS model per IoT device [2] or per attack [3]. Different from several related proposals, the aim of this work *is not* the mere application of increasingly complex deep learning models for intrusion detection. Rather, the novelty of our proposal is the application of well-founded principles to a **cross-device method**,which allows us to learn a single IDS model—as opposed to many separate models—atop the traffic of different IoT devices. The method stems from the following principles. A "usable" IDS should opt for unsupervised and semi-supervised approaches over supervised ones. It is unlikely that attacks are known beforehand; as such, unsupervised and semi-supervised approaches are more widely applicable. As for **IoT-specific constraints**, intrusion detection should pursue simplicity over complexity (e.g., small-footprint neural networks) in order to assure low detection latency, portability and energy efficiency. More importantly, given the ever-growing number and the dynamicity and complexity of devices in an IoT network, IDS models should be scalable and maintainable: learning separate models per device clashes with the ever-increasing scale of current IoT networks.

This study instantiates the cross-device method in the context of the detection of IoT botnets by **deep autoencoders (AE)**. The use of deep learning and autoencoders is motivated by several reasons. First, while understanding the "power of depth" in deep neural networks is an ongoing challenge in learning theory [4], deep networks perform better than traditional shallow neural networks in many practical applications (e.g., [5,6]). Furthermore, multiple AEs— possibly complemented by sophisticated feature selection methods— are often used in complex cascades/ensembles for IoT intrusion detection; as such, they are suitable to explore whether the complexity of related deep learning IDS proposals is actually justified. More importantly, AEs can be conveniently trained only by means of normal network traffic in order to learn a semi-supervised IDS model, i.e., one of the principles driving our proposal. The study is based on the widely used N-BaIoT dataset, which provides normal and attack traffic data collected with 9 IoT devices ranging from a thermostat to webcams and security cameras and arranged into *separate* datasets—1 per device—in the form of records of 115 features. Our study is twofold. First, we conduct a fine-grain experiment aiming to pursue a "conventional" approach by training a distinct AE per IoT device, i.e., **separate autoencoding**. Second, we train a single AE with the normal traffic of all the IoT devices, i.e., **all-in-one autoencoding**, which provides a single cross-device IDS model for botnet detection. Both separate and all-in-one models are tested by means of the typical metrics of recall (R), precision (P), false positive rate (FPR) and F1 score computed through a test set of normal and attack traffic held-out from training. These metrics—extensively described in Section 5—provide insights into the effectiveness of the classification: values of R, P and F1 score close to one and FPR close to zero indicate that the IDS is effective.

The results indicate that it is relatively easy to achieve impressive detection figures by separate training–testing an autoencoder on top of each individual device. In fact, a "minimal" AE with three hidden layers could be successfully applied—although after retraining from device to device—to seven out of nine devices with *no changes* of the implementation; moreover, the same AE could be extended to all the devices by means of a minor addition of neurons. Overall, **separate autoencoding** achieves 0.9995–1.0 recall, 0.9997–0.9999 precision, 0.0002–0.0417 FPR and 0.9997–0.9999 F1 score, depending on the device. Although remarkable, the separate approach underlies the need for maintain-

ing one model per device, which poses major scalability and maintainability issues in large-scale IoT networks. As for the **all-in-one autoencoding**, we observe that an AE of 5 hidden layers is enough to obtain a single cross-device IDS model, which achieves within 0.9994–0.9997 recall, 0.9999–1.0 precision, 0.0–0.0071 FPR and 0.9996–0.9998 F1 score, depending on the device. The results indicate that it is possible to train a single model with normal traffic collected from different devices, which is strongly beneficial to the FPR. The cross-device learning method paves the way for more scalable intrusion detection solutions in the context of IoT. As for the Cloud–Edge–IoT paradigm, Edge nodes are suited to host our all-in-one IDS, which means IoT devices are not subjected to extra computational and energy burden. Moreover, the proposed approach does not interfere with IoT operations. As only passive tracing of network traffic—also known as *sniffing*—is required, the approach is inherently nonintrusive.

In a previously paper [7], we documented a preliminary experimentation of the "all-in-one" notion. The novelty of this study with respect to the previous paper is a better exploration of the design space of the autoencoder, a more comprehensive set of experiments—leading to refinements and improvements of the original results—and additional findings along different directions on the subject. For example, here we address a larger number of IoT devices and an improved data partitioning scheme that aims to preserve sequences of related records. The findings of this paper should be contextualized with respect to the attacks and data available in the N-BaIoT dataset. Moreover, it is worth noting that privacy issues due to the adoption of the cross-device method are not in the scope of this paper. Our long-term objective is to capitalize on *federated learning* [8] and to leverage the decentralized data concept to cope with privacy facets. The rest of the paper is organized as follows. Section 2 discusses related work in the area. Section 3 provides the background on deep autoencoders and our semi-supervised intrusion detection approach. Section 4 addresses the IoT devices and datasets and presents data partitioning, training and implementation of the autoencoders. Section 5 presents the results of our study. Section 6 discusses the limitations and threats to validity of our study and how they have been mitigated, while Section 7 concludes the paper.

2. Related Work

This section presents related studies, surveying the state of the art of intrusion detection in IoT environments, with emphasis on the methods based on machine learning techniques.

Nowadays, the Internet of Things (IoT) has spawned a new ecosystem of connected devices, and an increasing number of organizations are using it to improve their performance, e.g., to operate more efficiently or to improve decision making. However, while the IoT has gained popularity, security challenges pose a significant barrier to widespread adoption and deployment of these devices. The security vulnerabilities introduced by complexity and interconnectivity of IoT devices and applications pave the way for the development of increasingly sophisticated anomaly detectors. Over the last few years, the use of machine learning to aid security and anomaly detection in IoT environments has become extremely important to face their security issues [9] and to develop subsequently appropriate lines of defense [10]. Al-Fuqaha et al. [11] surveyed some challenges and issues for the design and the deployment of IoT applications.

Since intrusion detection in the IoT domain is increasingly addressed through machine learning and (deep) neural networks, many ready-to-use public intrusion detection **IoT datasets** have been produced to support the testing of new designs. Most of these datasets are collected in synthetic environments—across various IoT domains—under normative conditions and multiple intrusion scenarios. They attempt to emulate real network traffic, and they do not contain any confidential data. Popular public IoT intrusion detection datasets are MedBIoT [12], N-BaIoT [2] and IoTID20 [13].

2.1. IDS with IoT, Machine and Deep Learning

Lopez-Martin et al. [14] propose a novel network intrusion detection method specifically developed for an IoT network. The approach is based on a conditional variational autoencoder (CVAE) that integrates the intrusion labels inside the decoder layers. The proposed method is less complex than other methods based on a variational autoencoder, and it provides better classification results than other familiar classifiers. The authors of [15], instead, propose a network intrusion detection system design for the IoT, which is based on a deep learning model comprising a customized feed-forward neural network. They tested the efficacy of the models for binary and multi-class classification. The results obtained show the efficacy of the proposed technique. In particular, the performance of the binary classifier was found to be close to 99.99%, while a detection accuracy of approximately 99.79% was achieved for multi-class classification.

Albulayhi et al. [16] proposed and implemented a novel extraction approach and feature selection (FS) for anomaly-based IDS in the IoT domain. The method starts by utilizing two entropy-based concepts (gain ratio (GR) and information gain (IG)) to extract and select appropriate characteristics in different ratios. A comparison of various deep learning algorithms such as convolutional neural networks (CNN) and recurrent neural networks (RNNs) such as long short-term memory (LSTM) and gated recurrent unit (GRU) networks is instead proposed by Ahmad et al. [17] and used to find zero-day anomalies within an IoT network with a false alarm rate (FAR) ranging from 0.23% to 7.98%. The authors in [18] propose a semi-supervised learning method for detecting intrusions, which is very similar to our approach. However, their experimentation is not conducted in an IoT context. In particular, an autoencoder and a variational autoencoder are used to extract flow-based features from network traffic data of the CICIDS2017 dataset.

The Fog computing domain along with elasticity of cloud and auto-scaling techniques are also active research topics [19]. Almieani et al. [20] show a model which uses multi-layered recurrent neural networks designed to be implemented for Fog computing security that is very close to the end-users and IoT devices. However, the authors show the validity of the proposal by using a balanced version of the NSL-KDD dataset. It is an obsolete dataset, not specifically conceived for IoT applications. As highlighted in [21], this issue might also lead to the lack of transferability of the impressive results obtained on reference datasets (possibly outdated and not free from statistical biasing) in even slightly different data collection settings.

In the last few years, federated learning (FL) has gained importance in the field of cybersecurity, with several works already using this paradigm for IoT security. The works presented in [22,23], for example, are specifically conceived for industrial IoT devices, and they analyze application samples and sensor readings, respectively, rather than network data. In [8], FL was studied through the use case of intrusion detection systems. This work also includes blockchain technology to mitigate the problems faced in adversarial FL. However, it concentrates on the early steps of intrusion detection rather than detecting already running malware, and it does not focus specifically on IoT devices.

2.2. Work on N-BaIoT Dataset

In [3], an IoT micro-security add-on is presented. The model comprises two key security mechanisms working cooperatively, namely: (i) a distributed convolutional neural network (DCNN) model for detecting phishing and application layer DDoS attacks, and (ii) a cloud-based temporal long short-term memory (LSTM) network model for detecting botnet attacks and ingesting CNN embeddings to detect distributed phishing attacks across multiple IoT devices. The N-BaIoT dataset is used for training the backend LSTM model.

The work closest to our proposal is Kitsune [24], an unsupervised learning approach to detect attacks online. Kitsune's core algorithm is KitNet, which uses a collection of autoencoder neural networks to distinguish between normative and abnormal traffic. The approach involves the integration of multiple autoencoders into a classifier. The experimental results show that Kitsune is effective with different attacks, and its performance is

as outstanding as offline detectors. Similarly, the authors in [2] propose a network-based anomaly detection method which extracts behavior snapshots of the network and uses deep autoencoders to detect anomalous network traffic emanating from compromised IoT devices. It is worth pointing out that the aforementioned methods, differently from our approach, create individual models per IoT device [2] or per attack [3], which are not suited to the ever-evolving IoT environments and security threats. Moreover, authors of [25] show that a single AE can obtain classification accuracy comparable to the ones published in the research literature for supervised networks and for more complex designs built around one or several AEs.

In [26], the authors propose an ML-based method for efficient botnet detection in IoT networks. The approach uses a hybrid model which pipelines a trained variational autoencoder (VAE) for meaningful latent feature extraction, and a one-class classifier OCC to classify network traffic from the IoT devices. The experimental results on the N-BaIoT dataset show that the latent representations generated from VAE help the OCC to perform better for botnet detection in terms of AUC metric with an acceptable detection time. Reference [27] proposes a federated-based approach which uses a deep autoencoder to detect botnet attacks using on-device decentralized traffic data. The proposed model, evaluated by means of the N-BaIoT dataset, differentiates benign patterns of behavior from malicious activities by means of decentralized on-device data at the edge layer.

In [28], the authors present a novel deep ensemble learning model framework called DeL-IoT for IoT anomaly detection. They use the deep and stacked autoencoders to extract features for stacking into an ensemble of probabilistic neural networks (PNNs) learning model for performance improvement while addressing the data imbalance problem. The N-BaIoT dataset is used as a benchmark dataset. The approach can detect anomalies with 0.99 detection rate for this dataset. In [29], the authors propose nested Log-Poly, a communicationally efficient model for distributed density estimation in naive Bayes classification. The method is evaluated on the N-BaIoT dataset.

Al Shorman et al. present in [30] a botnet detection mechanism based on a one-class support vector machine (OCSVM). The approach incorporates an unsupervised evolutionary IoT botnet detection method using a grey wolf optimization (GWO) algorithm to optimize the hyperparameters of the OCSVM and simultaneously identify features that best describe the IoT botnet problem. The system is tested using the N-BaIoT dataset and shows the best recall for the `Samsung SNH 1011 N` webcam device. Finally, Kan et al. [31] propose an adaptive particle swarm optimization convolutional neural network (APSO-CNN) to detect intrusions in the IoT networks. The model achieves the best accuracy on the N-BaIoT dataset if compared with three popular algorithms such as SVM, FNN and R-CNN.

Although deep learning approaches are extensively used for intrusion detection in IoT domains—as for many of the papers referenced above—we take a different perspective by considering well-founded principles and IoT-specific constraints in order to pave the way to better IDS design. Primarily, our approach capitalizes on a cross-device method, which allows us to learn a single IDS model atop the traffic of different IoT devices. Existing approaches to intrusion detection in the IoT domain, such as [2,3], differ from our solution because they provide one separate model per device [2] or per attack [3]. Furthermore, some of these approaches capitalize on training sets made up of normal and attack data. For example, in [3], the authors train one model for each attack by merging the normal traffic along with the malicious traffic representing the attack of interest. Therefore, they train, validate and test the model ten times, since in the reference dataset there are ten attack categories. We remark that our method leverages a semi-supervised learning approach, and it does not require anomalies during training. This makes it potentially effective for the detection of zero-day attacks, since the model is not constrained by specific attack data. One more key point is that we developed around the intuition that complexity is not justified because a single autoencoder is enough to obtain similar (if not better) performance figures compared to existing proposals. Among the existing proposals, multiple autoencoders

are often used in complex and mixed configurations (e.g., [24,26,28]). Our approach with a single AE ends up with a small-footprint neural network without any assistance from other external components, such as feature selection: this is a clear advantage in terms of simplicity of training and tuning.

3. Anomaly Detection Method

3.1. Background on Deep Autoencoders

Our cross-device method is based on the use of deep autoencoders, i.e., a specific type of neural network where the *input layer* has the same length as the *output layer*. The middle, hidden, layer of an autoencoder is also known as the *bottleneck layer*, and its dimension is lower than the input/output layer. An autoencoder consists of two parts: **encoder** and **decoder**. Let **x** be a vector of n real numbers $[x_1,x_2,...,x_n]$, such as the records representing IoT traffic for the dataset used in the experiments. The encoder maps **x** to a code vector—or hidden representation—**y** at the bottleneck layer. On the other hand, the decoder transforms **y** into a vector of n, i.e., the same size of **x**, real numbers $\mathbf{z} = [z_1, z_2, ..., z_n]$. Figure 1 represents an autoencoder with three hidden layers. Encoding–decoding formulas are given in Equations (1) and (2). They represent the case of an autoencoder with only one hidden layer:

$$\mathbf{y} = \sigma(W\mathbf{x} + b) \tag{1}$$

$$\mathbf{z} = \sigma'(W'\mathbf{y} + b') \tag{2}$$

where W, W', b and b' are weight matrices and bias vectors, while σ and σ' are activation functions. An autoencoder compresses the input into a lower-dimensional representation at the *bottleneck* layer, and then it reconstructs the output from the representation. Deep learning can be applied to autoencoders. In particular, multiple hidden layers can be used to provide depth: the resulting network is known as a *deep* or *stacked autoencoder* [32].

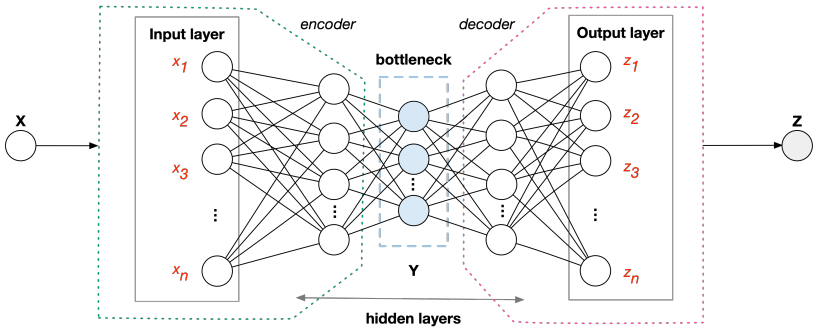

Figure 1. Representation of an autoencoder (three hidden layers).

In the autoencoder terminology, **z** is called the **reconstruction** of the input vector **x**. The "quality" of the reconstruction is summarized by the **reconstruction error** (RE), which measures the difference between the output **z** and the originating input **x**:

$$RE = \frac{1}{n}\sum_{i=1}^{n}(z_i - x_i)^2 \tag{3}$$

where z_i and x_i (with $1 \leq i \leq n$) denote the components of the output and input vector, and n is the dimensionality.

3.2. Autoencoder-Based IDS

An autoencoder is trained by means of a given set of points, i.e., the typical *training set* of a machine learning experiment. Each point **x** of the training set is fed to the autoencoder, and weight matrices and bias vectors are progressively adjusted in order to minimize the difference between **x** and its reconstruction **z**. After training, the autoencoder will

reconstruct accurately, i.e., low RE, future points "similar" to those used for training. Based on this principle, in order to pursue an IDS, we train the autoencoder solely by means of **normal** data points, i.e., records related to network traffic generated by the IoT devices under regular operations, which means the autoencoder learns a latent subspace of normal data points [33]. After training, the autoencoder—embedding a model of the "normal profile"—can identify any instance not conforming to the model as a potential intrusion.

The reconstruction error (RE) is a viable indicator to detect intrusions. Since the autoencoder is trained using only normal data points, it will generate (i) *low* RE (good reconstructions) for future normal inputs, and (ii) *high* RE (bad reconstructions) for intrusions. In fact, when the autoencoder attempts to process a data point, i.e., an IoT traffic record, that deviates from the norm, it will generate a high RE because it was never trained to reconstruct intrusions. The approach adopted in this paper falls within the larger scope of **semi-supervised** anomaly detection [34]. Moreover, as for any anomaly detection technique assigning a score to data points (RE in this study), we need a cut-off **detection threshold** to discriminate normal points from intrusions. In particular, intrusion detection is based on the use of the threshold value: the data points producing RE values under the threshold are considered normal, and those with REs above the threshold are marked as intrusions. The value of the detection threshold is an outcome of the training phase. As such, it is inferred from normal data points as described in the following.

3.3. Selection of the Detection Threshold

There are several practical **challenges** that undermine the selection of a suitable threshold in a semi-supervised training setting, such as for our study. An incorrectly selected threshold (either too low or too high) might cause misclassifications. It must be noted that assembling a "reliable" database of normal points for training purposes is a complex matter. For example, the normal points might be fraught with uncommon behaviors or outliers being accidentally included within the normal points. The labeling might be imperfect, i.e., intrusions being occasionally labeled as normal behaviors. The rationale behind our method is to clear out as many "strange"—although normal—training points as possible *before* computing the threshold: though belonging to normal data, spurious "out-of-the-crowd" behaviors will be more similar to intrusions than to normal points.

The threshold is computed by relying on a small, i.e., 10%, disjoint subset of the training set, which we call the **threshold set**. A representation of the method is in Figure 2 and consists of four steps:

1. **AE training**: the autoencoder undergoes the typical semi-supervised training procedure described above, which allows it to learn the normal profile of the data points;
2. **Outlier detection**: an *outlier detection* algorithm is applied to the threshold set in order to discriminate **inliers** from **outliers**;
3. **RE computation**: inliers and outliers are fed to the autoencoder: this step produces two separate vectors of reconstruction errors, i.e., RE_{IN} and RE_{OUT} of inliers and outliers, respectively;
4. **Threshold selection**: the detection threshold is obtained through a sensitivity analysis performed with RE_{IN} and RE_{OUT}.

At first, the threshold is initialized with the maximum RE in RE_{OUT}; then, the threshold is progressively lowered until it finds an "equilibrium" between inliers and outliers, i.e., inliers whose RE is below the threshold against outliers characterized by an RE above the threshold. In this study, we use the **isolation forest** [35], although the threshold selection method *does not* mandate a specific outlier detection algorithm.

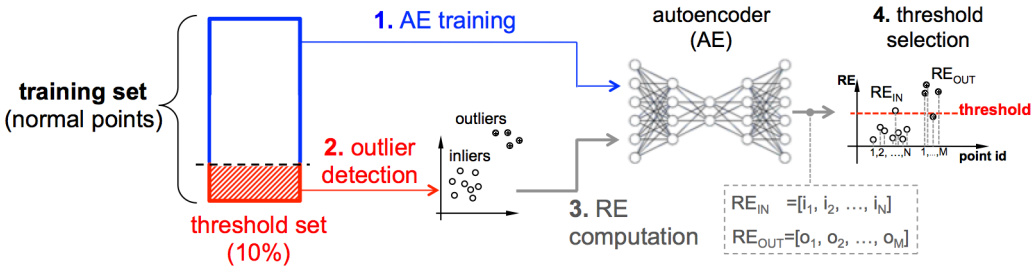

Figure 2. Steps underlying the threshold selection method.

4. Dataset, AE Design and Implementation

The validity of the cross-device method is assessed by a **twofold** experimentation. First, we pursue a "conventional" approach, which consists of training–testing the autoencoder-based IDS with the data of one IoT device each time, which leads to one separate model per device. It is worth pointing out that this approach is used by several related papers presented above. Later, the autoencoder-based IDS is trained with a dataset consisting of the normal traffic of all the IoT devices in hand—and therefore **all-in-one**—that leads to a single model for all the devices. It is worth noting that the notion of all-in-one autoencoder underlies the availability of training data pertaining to different devices. Consequently, training is not expected to happen "in place" on single IoT devices. Rather, the proposed approach is *suited* to the nodes traversed by the traffic generated from different network devices. A typical example is the Cloud–Edge–IoT paradigm: Edge nodes—at the boundary between two networks —implement common functions, such as routing, monitoring, and storage of data passing between networks. In the context of the IoT, Edge nodes encompass a broad range of devices and are eligible to host our all-in-one IDS. As a consequence of deploying the all-in-one IDS onto Edge nodes, IoT devices **are not** subjected to extra computational and energy burden; as said above, energy efficiency is among the principles that underlie the design of our proposal. Moreover, as only passive tracing of network traffic is required, the approach is inherently nonintrusive. In the following, we describe the datasets, tuning and training of both separate and all-in-one autoencoding.

4.1. Reference Dataset and Partitioning

The dataset considered in this paper is **N-BaIoT** [2]. It provides a public botnet IoT dataset available at the **University of California at Irvine (UCI)** machine learning repository. The authors deployed all of the components of two botnets in an isolated lab and used them to infect nine commercial IoT devices listed in Table 1. All the devices execute the attacks of two botnets—namely `Mirai` and `BASHLITE`—that have previously infected the IoT devices.

Table 1. IoT devices in the N-BaIoT dataset.

IoT Device	Category
`Danmini`	`Doorbell`
`Ecobee`	`Thermostat`
`Ennio`	`Doorbell`
`Philips B120N/10`	`Baby Monitor`
`Provision PT-737E`	`Security Camera`
`Provision PT-838`	`Security Camera`
`Samsung SNH 1011 N`	`Webcam`
`SimpleHome XCS7-1002-WHT`	`Security Camera`
`SimpleHome XCS7-1003-WHT`	`Security Camera`

For each device, the data were obtained under both normal operations and attack conditions. The dataset consists of fixed-length records of features computed from the

lower-level network activity. In particular, each record is identified by 115 features with a systematic structure plus a class label (e.g., "normal" or "TCP attack"). The features model traffic statistics over several temporal windows. Each statistic is further temporally aggregated using a weighted sum that progressively leads to the decay of the oldest contributions to the sum. It is worth pointing out that botnet infections consist of multiple steps, such as propagation, bot infection, communication with C and C server, and performing other types of malicious activities [36]. According to [2], it is not sufficient to determine only the early stages of infection. The data contained in the N-BaIoT dataset pertain only to the last stage of botnet construction, when IoT bots begin launching attacks. Interested readers are referred to [2] for any additional information on the N-BaIoT dataset.

Our experiments are based on all the devices described in the N-BaIoT dataset. Experiments are performed in a binary classification scenario. All the records produced within different types of attacks are considered as belonging to a unique generic class named `ATTACK`—encoded with the numeric label 1—whereas `NORMAL` records are assigned 0 as label. It is worth remarking that the N-BaIoT dataset is organized into *separate datasets*, each containing both normal and attack traffic corresponding to a single IoT device. For our experiments, we split the original datasets into three **disjoint splits**: *training set*, *validation set* and *test set*. While splitting the dataset corresponding to an IoT device, we preserve the original sequence of the records because—as said above—the features of N-BaIoT are based on the temporal windows and are aggregated using weighted sums. Moreover, each record of the original dataset is assigned to a unique split. For each IoT device, we obtain:

- *Training set.* It contains 70% of the total `NORMAL` records; moreover, 10% of the training set, according to the threshold selection criteria described in Section 3, is the "threshold set", meant for the threshold selection process;
- *Validation set.* It contains 15% of the total `NORMAL` records as for any machine learning experiments, it is used to provide an unbiased evaluation of the model in order to find the optimal values for the hyper-parameters;
- *Test set.* It contains 15% of the total `NORMAL` records and all `ATTACK` records. The records in this set are accompanied by the corresponding labels that are used to assess the correctness of the predictions.

Table 2 shows the cardinality of the sets for the first group of experiments, i.e., separate autoencoding. Table 3 shows the cardinality of the training, validation and test sets for the second group of experiments, i.e., all-in-one autoencoding, where a single training/validation set—meant to come up with one model—sum up to the cardinalities of the training/validation sets in Table 2. It must be noted that test sets are "held-out" from training/validation, and they will be used in Section 5 for measuring the detection capabilities of the autoencoder-based IDSes.

Table 2. Training, validation and test set size (separate autoencoding).

IoT Device	*Training Set*	*Validation Set*	*Test Set*	
	NORMAL	**NORMAL**	**NORMAL**	**ATTACK**
Danmini	34,684	7432	7432	968,740
Ecobee	9181	1966	1966	822,753
Ennio	27,370	5865	5865	316,395
Philips B120N/10	122,668	26,286	26,286	924,327
Provision PT-737E	43,508	9323	9323	766,096
Provision PT-838	68,960	14,777	14,777	738,367
Samsung SNH 1011 N	36,506	7822	7822	323,067
SimpleHome XCS7-1002-WHT	32,611	6987	6987	816,461
SimpleHome XCS7-1003-WHT	13,670	2929	26,286	831,285

Table 3. Training, validation and test set size (all-in-one autoencoding).

IoT Device	*Training Set* NORMAL	*Validation Set* NORMAL	*Test Set* NORMAL	*Test Set* ATTACK
Danmini			7432	96,874
Ecobee			1966	822,753
Ennio			5865	316,395
Philips B120N/10			26,286	924,327
Provision PT-737E	389,158	83,387	9323	766,096
Provision PT-838			14,777	738,367
Samsung SNH 1011 N			7822	32,3067
SimpleHome XCS7-1002-WHT			6987	816,461
SimpleHome XCS7-1003-WHT			26,286	831,285

4.2. AE Design

The design of a deep neural network, such as the autoencoder, is based on choosing the values of many hyperparameters—number of layers, neurons per layers, activation functions and so forth—that are subject to fine-grain tuning. Other relevant and critical parameters, such as the number of epochs, bath size and optimization algorithms, pertain to the learning process. For the time being, there is no *scientific* rule for optimizing the hyperparameters of an AE. In this work, the selection of the hyperparameters is driven by our previous experience and good practices on tuning deep autoencoders in a close IDS domain [25]. After having set a "reasonable" initial configuration of the autoencoder (e.g., number of neurons at the bottleneck much less than the number of features, use of the rectified linear unit activation function for the hidden layers, hyperbolic tangent activation function at the output layer and RMSProp optimizer), additional fine tuning was performed through manual search. To this aim, the manual search was validated by experimental tests carried out by analyzing the outcome of the autoencoder—RE in our study—with respect to the abovementioned **validation set**, i.e., the independent set of data points purposely intended to support the selection of the hyperperameters. The use of a validation set makes it certain that the final values of the hyperperameters *are not* biased by the test sets; rather, the test sets contain "held-out" data points, i.e., not used at all for training and hyperperameters selection.

4.2.1. Separate Autoencoding

We found that the configuration reported in Table 4 guarantees an effective design, i.e, low RE on the validation set, for the separate AEs setting. The selected AE is made up of **three hidden layers** and a different number of neurons depending on the configuration (*A* Table 4a or *B* Table 4b). We ended up with two different configurations because, despite the numerous fine tuning operations, it was not possible to find a *single working* configuration for all IoT devices. In particular, the three densely connected layers include N-48-6-48-N neurons for *Configuration A* and N-64-6-64-N neurons for *Configuration B*. In the rest of the paper, N (i.e, the number of neurons of the input/output layer of the autoencoder) is equal to 115, which is the dimensionality of the traffic records of the N-BaIoT dataset. The classical rectified linear unit (ReLu) has been selected for the encode layer, the decode layer and the bottleneck layer, while for the output layer the hyperbolic tangent (Tanh) activation function has been used. We train an AE on NORMAL data points of each IoT device for 100 epochs with batch size 512 using the RMSProp optimizer with learning rate value $lr = 0.0001$.

Table 4. Separate autoencoding: layering structure of the AE (all the layers are dense).

AE Design		
Layer	**Activation**	**Neurons**
Input	-	*N*
Hidden 1	ReLU	*48*
Hidden 2	ReLU	*6*
Hidden 3	ReLU	*48*
Output	tanh	*N*

(a) Configuration A

AE Design		
Layer	**Activation**	**Neurons**
Input	-	*N*
Hidden 1	ReLU	*64*
Hidden 2	ReLU	*6*
Hidden 3	ReLU	*64*
Output	tanh	*N*

(b) Configuration B

4.2.2. All-in-One Autoencoding

For the second set of experiments, we tested different network designs before choosing a good configuration of the parameters. We found that adding **width** and **depth** to the AE is beneficial. In particular, Table 5 shows the design used for the all-in-one autoencoding experiment. The chosen AE is made up of **five hidden layers**. These layers are densely connected and include *N*-64-24-6-24-64-*N* neurons, where *N* is the number of features of the data points. Again, the rectified linear unit (ReLu) has been selected for the encode layer, the decode layer and the bottleneck layer, while for the output layer the hyperbolic tangent (Tanh) activation function has been used. We train the autoencoder on `NORMAL` data points for 100 epochs with batch size 2048 using the RMSProp optimizer with learning rate value $lr = 0.0001$.

Table 5. All-in-one autoencoding: layering structure of the AE (all the layers are dense).

AE Design		
Layer	**Activation**	**Neurons**
Input	-	*N*
Hidden 1	ReLU	64
Hidden 2	ReLU	24
Hidden 3	ReLU	6
Hidden 4	ReLU	24
Hidden 5	ReLU	64
Output	tanh	*N*

4.3. Implementation and Training

We implement the autoencoders in `python` with `Keras` (https://keras.io/, accessed on 3 January 2023) (Version 2.6.0) and TensorFlow `TensorFlow` (https://www.tensorflow.org/, accessed on 3 January 2023) (Version 2.6.0) libraries on a *Lambda* workstation provided with an AMD Threadripper 3975WX processor with 32 cores. During the training phase, the weights and biases of the encoder and decoder are calculated and optimized with respect to `NORMAL` training data. When the training is started, the AE neurons are randomly initialized, and input data are presented in batches (512 for the separate approach and 2048 for the all-in-one approach) and through a given number of epochs (100 for both approaches). The

system tries to minimize the loss, setting aside a small ratio of reserved data to validate the optimization actions performed—modifications of the weights in the network—so as to signal overfitting. A solution is to compute the loss as the **mean squared error** at the output units; this matches the definition of reconstruction error (RE) presented above. As outlined in Section 3, during training the autoencoder learns the relationships among the features in the training set; it is worth pointing out that the training phase of an AE takes around 1 minute in the worst case. Our semi-supervised training process (*no anomalies* at training time) is potentially valuable to complement current technologies that rely on pre-established specifications of anomalies.

5. Results

We run the test sets of the devices in hand against the AE described in Section 4. In order to evaluate our proposal, we focus on the following two points: (i) the performance of separate AEs, individually trained with the normal traffic of each IoT device, and (ii) the performance of the all-in-one AE (i.e., adopting **all-in-one autoencoding**) trained once with the normal traffic of all devices and then applied to the test of the devices. Since the test sets are labeled, each RE produced by the AEs in both configurations can be linked to the label of the corresponding data point, which can be used for evaluation. It is worth pointing out that the AE saw no attack at training time.

The detection performance is measured by computing the typical metrics of *recall* (R), *precision* (P), *false positive rate* (FPR) and *F1 score*. These metrics are computed as follows:

$$R = \frac{TP}{TP + FN} \qquad P = \frac{TP}{TP + FP} \tag{4}$$

$$FPR = \frac{FP}{FP + TN} \qquad F1\ score = 2 \cdot \frac{P \cdot R}{P + R} \tag{5}$$

where true positive (TP) and true negative (TN) represent the points that are correctly classified, while false positives (FP) and false negatives (FN) indicate misclassifications. For example, TP is the set of `ATTACK` points whose RE is higher than the threshold; similarly, TN is the set of `NORMAL` points whose RE is lower than the threshold. In particular, *recall* is the ratio of the number of true positives to the sum of the number of true positives and false negatives, *precision* is the ratio of the number of true positives to the sum of the number of true positives and false positives, while the *false positive rate* is the ratio of the number of false positives to the total number of normal data points in the test set. Finally, *F1 score* is the harmonic mean of precision and recall.

5.1. Separate Autoencoding

We process the test set of a given device after the AE is retrained for that specific device beforehand. Table 6 provides the evaluation metrics for all the IoT devices. The results show that the use of individual AEs leads to high detection figures. The recall values range from 0.9995 (`Samsung SNH 1011 N` webcam) to 1.0 (`Danmini` doorbell). The minimum false positive rate (0.0002) is obtained for the `Provision PT-737E` security camera. Overall, the results are outstanding for all the IoT devices, with recall and precision close to 1.0 with a reasonable false positive rate. Moreover, the F1 score—the harmonic mean of precision and recall—is computed as a key metric, since precision and recall are both relevant indicators of the model performance. From the analysis of the values in Table 6, it is possible to note that the F1 score is within 0.9997–0.9999, i.e., close to 1.0. This result indicates a good trade-off between precision and recall.

Table 6. Separate autoencoding: evaluation metrics (*Configuration A*: *N*-48-6-48-*N*; *Configuration B*: *N*-64-6-64-*N*).

IoT Device	Recall	Precision	FPR	F1 Score	Configuration
`Danmini`	1.0	0.9999	0.0108	0.9999	*A*
`Ecobee`	0.9999	0.9999	0.0417	0.9999	*A*
`Ennio`	0.9998	0.9997	0.0114	0.9998	*A*
`Philips B120N/10`	0.9998	0.9999	0.0007	0.9999	*A*
`Provision PT-737E`	0.9998	0.9999	0.0002	0.9999	*B*
`Provision PT-838`	0.9999	0.9998	0.0088	0.9998	*B*
`Samsung SNH 1011 N`	0.9995	0.9999	0.0006	0.9997	*A*
`SimpleHome XCS7-1002-WHT`	0.9999	0.9999	0.0007	0.9999	*A*
`SimpleHome XCS7-1003-WHT`	0.9998	0.9999	0.0187	0.9998	*A*

It is worth pointing out that the rightmost column of Table 6 reports the network configuration used for each IoT device. In particular, as highlighted in Section 4, *Configuration A* refers to a *N*-48-6-48-*N* network design, while *Configuration B* refers to a *N*-64-6-64-*N* network design. We emphasize that the use of multiple AE designs is almost mandatory, since it was not trivial to find a *"one fits all"* configuration for all the IoT devices. As previously explained, the selection of two "best" configurations—*A* and *B*—is guided by experimental tests carried out by analyzing the RE figures obtained by multiple AE models.

Finding: Different from similar proposals in the area, which rely on complex cascades and ensembles of autoencoders—possibly complemented by the use of feature selection methods—if not other schemes, such as CNNs and LSTMs, a "minimal" and simple autoencoder with three hidden layers is more than enough to obtain remarkable results when train–test is performed separately for each device.

Overall, the results obtained are notable. However, the hypothesis of training an AE for each device remains unrealistic, especially if the intrusion detection system is intended to be deployed in time-dependent and rapidly evolving IoT environments.

5.2. All-in-One Autoencoding

The use of an all-in-one detector—applied to the devices assessed with *no change* to its architecture (layers, number of units and weights)—is more suited to large-scale and dynamic IoT environments. We evaluated the performance of the all-in-one AE, trained with the merged normal traffic of all IoT devices. Table 7 provides the evaluation metrics. It turns out that the detection performance of the all-in-one model with five hidden layers is in line with the figures obtained by the separate autoencoding solution. The best results in terms of recall (R = 0.9997) are obtained with the test sets of the `Danmini` doorbell, `Ecobee` thermostat, `Philips B120N/10` baby monitor, `Provision PT-838`, `SimpleHome XCS7-1002-WHT` and `SimpleHome XCS7-1003-WHT` security cameras; the FPR is reasonably low—exactly 0.0 for the `Provision PT-838` and `Provision PT-737E` security cameras—for most devices. The F1 score, instead, is 0.9996 for the `Ennio` doorbell, 0.9997 for the `Samsung SNH 1011 N` webcam, and 0.9998 for all the other devices. This confirms we reached an acceptable trade-off between precision and recall.

Table 7. All-in-one autoencoding: evaluation metrics (*N*-64-24-6-24-64-*N*).

IoT Device	Recall	Precision	FPR	F1 Score
Danmini	0.9997	0.9999	0.0004	0.9998
Ecobee	0.9997	0.9999	0.0071	0.9998
Ennio	0.9994	0.9999	0.0017	0.9996
Philips B120N/10	0.9997	0.9999	0.0001	0.9998
Provision PT-737E	0.9996	1.0	0.0	0.9998
Provision PT-838	0.9997	1.0	0.0	0.9998
Samsung SNH 1011 N	0.9994	0.9999	0.0006	0.9997
SimpleHome XCS7-1002-WHT	0.9997	0.9999	0.0002	0.9998
SimpleHome XCS7-1003-WHT	0.9997	0.9999	0.0058	0.9998

For the sake of completeness, we also show in Table 8 the evaluation metrics for one "smaller" configuration, the aforementioned *Configuration A*, which was mostly outstanding in the separate autoencoding experiment; however, it performs worse in the all-in-one test. For example, the recall drops to 0.3549 for the Samsung SNH 1011 N webcam device and to 0.3506 for the Ennio doorbell device. This indicates that with respect to the problem addressed and the data in hand, *deepening* and *widening* the autoencoder can improve intrusion detection in the all-in-one setting. Regarding the effects of the deeper and wider configuration on processing times, we found that for the *N*-64-24-6-24-64-*N*, the detection latency per record is reasonable (about 1 microsecond per record).

Finding: The cross-device training method assures remarkable detection figures when compared to the separate autoencoders. Although the best all-in-one AE is a wider and deeper network than separate autoencoding, its training time and detection latency per record are acceptable.

Our results show that it is possible to train a single model with normal traffic collected from different devices. While the findings of this paper should be contextualized with respect to the attacks and devices of N-BaIoT, we believe there is room to conceive more scalable and centralized intrusion detection solutions in the context of IoT, based on the notion of all-in-one models. An AE model trained *"once and for all"* represents a lightweight and device-independent solution. This is a considerable advantage in terms of transferability and adaptability.

Table 8. All-in-one autoencoding: evaluation metrics (AE configuration *A*: *N*-48-6-48-*N*).

IoT Device	Recall	Precision	FPR	F1 Score
Danmini	0.7955	1.0	0.0	0.8861
Ecobee	0.7568	0.9999	0.0040	0.8616
Ennio	0.3506	1.0	0.0	0.5192
Philips B120N/10	0.7851	0.9999	0.0006	0.8796
Provision PT-737E	0.7276	1.0	0.0	0.8423
Provision PT-838	0.7371	0.9999	0.0001	0.8486
Samsung SNH 1011 N	0.3549	1.0	0.0	0.5239
SimpleHome XCS7-1002-WHT	0.7639	1.0	0.0	0.8661
SimpleHome XCS7-1003-WHT	0.7577	0.9999	0.0068	0.8621

5.3. Discussion of the Results

In the following, we complement the results presented above by a visual analysis and a discussion on the operation of the two considered approaches.

Figure 3 shows the reconstruction error (RE) obtained on the test set of the `Danmini` doorbell and the `Provision PT-838` security camera devices by considering the separate autoencoding approach (Figure 3a,c) and the all-in-one autoencoding approach (Figure 3b,d). We limit the visual analysis to these two devices because all the remaining cases lead to similar findings. The REs of normal and attack points of the test sets refer to the separate and all-in-one experiment, respectively. Each data point is marked by either •—normal data point— or ▼—attack data point—for better visualization; the x-axis is the id of the point in the test set; the y-axis is the corresponding RE. A semi-logarithmic scale (x-axis in linear scale and y-axis in log scale) is used to better visualize the REs. The horizontal dashed line shows the anomaly threshold (a by-product of the training phase).

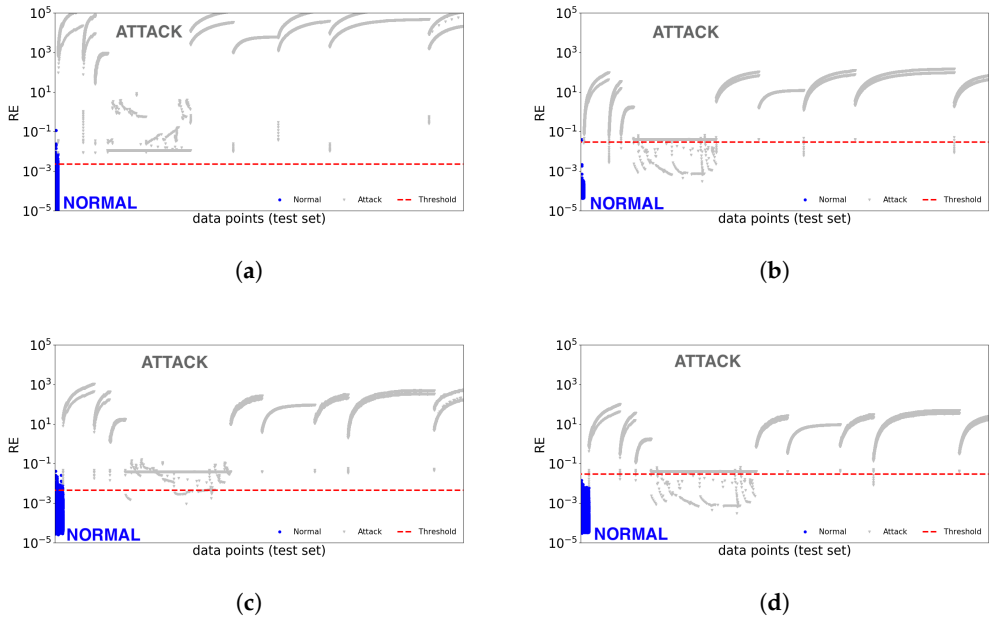

Figure 3. RE of the test sets for the `Danmini` doorbell and `Provision PT-838` security camera devices: (**a**) `Danmini`–Separate autoencoding; (**b**) `Danmini`–All-in-one autoencoding; (**c**) `Provision PT-838`–Separate autoencoding; (**d**) `Provision PT-838`–All-in-one autoencoding.

In general, according to Figure 3, it can be noted that both autoencoding approaches—separate and all-in-one—return a low RE for almost all the normal points, which are thus below the detection threshold; on the other hand, most of the attack points lead to high RE and are well above the threshold. Overall, Figure 3 is useful to gain a visual understanding of true negatives (`NORMAL` points below the threshold), false positives (`NORMAL` points above the threshold), false negatives (`ATTACK` points below the threshold) and true positives (`ATTACK` points above the threshold) of both autoencoding approaches.

It is easy to link the visual results and the AE performance figures by analyzing the recall and FPR values—`Danmini` and `Provision PT-838` devices—reported in Table 6 (separate autoencoding) and Table 7 (all-in-one autoencoding). In particular, for the `Danmini` doorbell device, Figure 3a (separate autoencoding) shows a higher number of `NORMAL` points above the threshold—false positives—if compared with Figure 3b (all-in-one autoencoding); in fact, the FPR values for this device are 0.0108 (separate autoencoding) and 0.0004 (all-in-one autoencoding). On the other hand, Figure 3a (separate autoencoding) shows no attack points under the threshold—false negatives—if compared with Figure 3b (all-in-one autoencoding). Again, this visual finding is confirmed by the performance figures. As a matter of fact, the recall values for this device are 1.0 (separate autoencoding) and 0.9997 (all-in-one autoenconding). The same considerations can be extended to the `Provision`

PT-838 security camera device (Figure 3c,d). It is worth pointing out that this trend—lower FPR at the expense of lower recall for the all-in-one approach—is preserved on almost all devices (as shown in Tables 6 and 7).

Overall, we believe that the all-in-one approach has remarkable strengths. Although the recall values obtained with this approach are slightly lower than the separate approach ones, they are surely acceptable (in the range 0.9994 to 0.9997). Moreover, the FPR is always close—or even equal—to 0; on the other hand, the FPR values obtained with the separate autoencoding are worse, and they range from 0.0002 to 0.0417. In real-life environments, false positives can be cumbersome to deal with for network administrators. A high number of false positives leads to lost confidence in the alerts and to lower defense levels. Therefore, in addition to the advantages related to the scalability and the portability (a single network topology for all the devices), the all-in-one approach might also be beneficial to mitigate the false positive problem.

5.4. Comparison with the Related Proposals Using N-BaIoT

It is worth noting that N-BaIoT is a widely used dataset in the IoT-based anomaly detection field. Consequently, we can compare the metrics obtained in our study with other similar proposals in the literature. Authors in [2,3,24], as mentioned in Section 2, assess different state-of-the-art anomaly detection methods on the same N-BaIoT dataset used in our paper. However, different from our study, which shows the performance metrics per IoT device, the authors report the performance figures per attack. For example, in [3], the authors obtain a recall of 0.99 for the `gafgyt combo` attack and of 1 for the `mirai_ack` attack. Again, the authors of the detector proposed in [26] show only the ROC curves—and AUC values—as performance figures; therefore, they cannot be directly compared with our results. As for the use of the autoencoder, this type of neural network is leveraged to perform automatic feature extraction with the aim of reducing the dimensions of the data being processed. Therefore, differently from our study, the autoencoder is simply a building block of a more complex detection architecture. As for the federated-based approach proposed in [27], the recall values are below 0.75 under the 50 epoch range; the score jumps up to 0.99 from 60 epochs onward and remains at that high level while using the federated learning approach, for all experiments. These results are in line with our recall values. The performance obtained in DeL-IoT [28], instead, is comparable to the ones obtained in our study. However, the authors use the autoencoder to perform feature extraction and not anomaly detection (as intended in our study). As for the approach described in [30] mentioned in Section 2, tested using the N-BaIoT dataset, the best recall—0.999—is found for the `Samsung SNH 1011 N` webcam device. We also obtained a recall equal to 0.9994 for this device. Similar values of recall are preserved for all the IoT devices in our study, different from the results shown in [30]. Moreover, according to the figures reported in [30], the best FPR—0.009—is obtained for the `Ecobee` thermostat device. For the other devices, the FPR ranges from 0.016 to 0.098. The FPR obtained with our approach, instead, ranges from 0.0 to 0.0071.

The results obtained with our cross-device method are inline—if not even better—with those obtained by other anomaly detection techniques assessed through the N-BaIoT dataset. Besides the inherent benefits of deep learning, the improvement over existing methods is explained by the interplay of **two factors**, which are the novel contributions of this work. The former is the cross-device notion: learning a single model atop the normal traffic related to different IoT devices allows for the recognition of a greater number of nominal behavioral patterns and related variants. This makes our approach valuable in terms of adaptability and transferability across the devices, and it leads to better performance. The latter is the proposed outlier-based threshold selection method. In this respect, there exist several "fixed" criteria, such as *mean*, *median*, *mean plus standard deviation* and n^{th} *percentile*, that can be found in the related literature to determine a threshold from normal data. For example, the authors in [2] consider the mean and the standard deviation to select the threshold. It should be noted that a "fixed" criteria approach may not be the best fit for

all the datasets in hand. In order to overcome this limitation, we proposed a data-driven method that can adapt to potential outliers. Our method improves the results over traditional techniques because it allows mitigating the inherent imperfections of the training data and occasional inconsistencies of the autoencoders at providing good reconstructions also for outliers, which are well-known in the literature [37].

6. Limitations and Threats to Validity

The evaluation of an IDS is a complex subject, and it depends on many factors including—but not limited to—the underlying network devices, topology and speed, the nature of the attacks and the quality of the legitimate traffic being used to infer the "normal profile" of the devices. For example, the normal traffic might be fraught with spurious, out-of-the-crowd behaviors and by imperfect labels. The proposed outlier-based threshold selection method mitigates the effect of the imperfections of the normative traffic. As much related work in the area, the findings of this paper must be contextualized with respect to the attacks and data of the adopted dataset. In this respect, many public security datasets have been proposed over the years; some of them have gained significant popularity and become widely consolidated benchmarks for intrusion detection algorithms and tools, such as N-BaIoT. We are aware that a "lab-made" dataset, such as N-BaIoT and many others, might be a simplification of real-life production networks; however, adopting a reference benchmark makes it possible to compare the results obtained with those of the related proposals (as discussed in Section 2). As for the application of the autoencoder and outlier-based method to a further dataset, we used the 2021 version of the CICIDS2017 dataset (https://downloads.distrinet-research.be/WTMC2021/tools_datasets.html, accessed on 3 January 2023) in a previous paper that investigates the issues of training effective IDS models by a single autoencoder [25]. Besides the use of a synthetic dataset, another simplification is in the format of data addressed by our study. N-BaIoT is based on a network made of different IoT devices, which range from a thermostat, to doorbells and security cameras—possibly from different vendors—and network equipment (e.g., WiFi access point, switch and router); the "nature" of the IoT devices in real-life network environments is even more exacerbated when compared to a synthetic dataset. The model proposed in this paper can detect threats in a network made of different devices under the assumption that all the traffic is transformed into a *standard* data format consisting of fixed-length records irrespective of the device. Nowadays, there exist many products for capturing network traffic and generating fixed-length records suited for machine and deep learning purposes. The availability of a standard data format simplifies the use of typical IDS approaches applied to common networks. As a final remark, the proposed model is not meant to cover the entire life cycle of an infection; rather, our work focuses on the last stage of botnet detection that pertains to IoT bots launching the attacks. As for any data-driven, as those involving deep learning methods, there may be concerns regarding the validity and generalizability of the results. We discuss them based on four aspects of validity listed in [38].

Construct validity. The study builds around the intuition that it is possible to learn a cross-device IDS model only on top of normative training traffic records related to several IoT devices. Our study, based on deep autoencoders, has the potential to drive scalable and maintainable IDS solutions that can cope with the ever-growing complexity of IoT networks. This *construct* has been investigated with normal and attack traffic from a widely accepted benchmark by the related IoT literature. Experiments are based on the ubiquitous `Keras-tensorflow` deep learning framework. Overall, the study is supported by well-founded theory and practice, and the typical evaluation metrics of recall, precision, false positive rate and F1 score for comparative purposes.

Internal validity. Our study implements several countermeasures aiming to mitigate internal validity threats. For example, we made sure to test our autoencoder-based IDS by means of held-out data, i.e., not used at all for training and hyperperameters optimization. The reference dataset used to conduct the experiments contains attacks that follow up

BASHLITE and Mirai botnet infections. Overall, the attacks are well-established in the literature and have been used to validate many existing IDS proposals. The traffic is based on several devices; more importantly, the training of both separate and all-in-one autoencoders—semi-supervised—is not biased by the specific attack types. The use of such a diverse mixture of controlled conditions and techniques aims to mitigate internal validity threats.

Conclusion validity. Conclusions have been inferred through a careful design of the experiments. The conclusions of the study are consistent along the different dimensions of our experiments, which makes our finding perfectly reasonable and technically sound. For example, we assess the sensitivity of the IDS performance with respect to different configurations of the autoencoders, i.e., in terms of the number of hidden layers and neurons. We present an extensive discussion of the results. The key findings of the study are consistent across the devices. The proposed approach detects the attacks at hand irrespective of devices, which means the results are not biased by a specific attack or device. More importantly, the IDS performance achieved is in line with the related literature, which provides a reasonable level of confidence in our analysis.

External validity. The cross-device model can be applied to other similar systems, types of neural networks and attacks. Nowadays, there exist many public datasets and attack tools, which make our approach definitively feasible in practice. Our approach does not interfere with system operations: as only passive tracing is required, the approach is inherently *nonintrusive*. More importantly, there exist many products for capturing network packets and generating fixed-length records, which allow porting our method to other systems. In fact, in this paper we successfully applied the method to the network traffic—transformed into a data format suited for machine learning experiments—related to several IoT devices and attack types to mitigate external validity threats. We are confident that the experimental details provided in the paper would support the replication of our study by future researchers and practitioners.

7. Conclusions

Nowadays, the IoT paradigm is enabling new application scenarios. However, simultaneously to the advances in new technologies, the number and variety of cyberattacks have grown. Ongoing projects for enhancing IoT security include methods for providing data confidentiality and authentication, access control within the IoT network, privacy and trust among users and things and the enforcement of security and privacy policies. Nevertheless, even with these mechanisms, IoT networks are vulnerable to multiple attacks, such as botnets, aimed to disrupt the network. For this reason, another line of defense, designed for detecting attackers is needed. IDS is designed to fulfill this purpose. Traditional machine learning techniques have been widely applied in the literature to detect attacks in IoT scenarios. Frequently, IoT intrusion detectors are implemented by means of deep learning techniques with individual models per IoT devices or per attack. However, these assumptions might be not suited to high-scalable and dynamic IoT environments.

This paper proposes a novel cross-device method, which allows learning a single IDS model atop the traffic of different IoT devices included in the widely used N-BaIoT dataset. In particular, the proposed approach—based on the use of an all-in-one deep autoencoder—differs from the other contributions proposed in this area, which use the training data to learn a separate IDS model per IoT device or per attack. Our results show that it is relatively easy to achieve remarkable detection results by training–testing a model on the top of individual devices. The all-in-one deep autoencoding approach, instead, proves that it is possible to preserve the overall performance within 0.9994–0.9997 recall, 0.9999–1.0 precision, 0.0–0.0071 FPR and 0.9996–0.9998 F1 score, depending on the device, by training a single model with the normal traffic collected from different devices. The method paves the way for more scalable intrusion detection solutions in the context of IoT; moreover, it is suited to the Cloud–Edge–IoT paradigm.

In the future, we will extend our analysis to further datasets and devices in order to discover potential limitations of the approach. In this respect, we believe that *transferability* of the models is a primary concern. Although the IoT devices might belong to the same category (e.g., doorbells or security cameras) with common physical characteristics, these can be released by different manufacturers. The model may not necessarily be transferable to all the devices of the same category but of different manufacturers. Therefore, we will test and tune our approach with a wide set of devices of the same category but from different manufacturers in order to develop both a *device-independent* and *manufacturer-independent*, all-in-one, model. With data privacy and integrity becoming a major concern, in recent years new technologies such as federated learning have emerged. They allow training machine learning models with decentralized data while preserving its privacy by design. Federated learning is a collaborative learning approach where the devices interact with a centralized entity but without the need to share their data. In the future, we will also extend our approach in the context of federated learning, which can be used for intrusion detection. Furthermore, the continuing increase of new unknown attacks requires corresponding improvements to the performance of the IDS solutions to identify zero-day attacks. Future research will also investigate to discover and mitigate the actions attackers might take to evade detection.

Author Contributions: Conceptualization, M.C., A.P. and U.V.; Methodology, M.C., A.P. and U.V.; Software, M.C., A.P. and U.V.; Validation, M.C., A.P. and U.V.; Formal analysis, M.C., A.P. and U.V.; Investigation, M.C., A.P. and U.V.; Resources, M.C., A.P. and U.V.; Data curation, M.C., A.P. and U.V.; Writing—original draft, M.C., A.P. and U.V.; Writing—review & editing, M.C., A.P. and U.V.; Visualization, M.C., A.P. and U.V.; Supervision, M.C., A.P. and U.V.; Project administration, M.C., A.P. and U.V.; Funding acquisition, M.C., A.P. and U.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found at the URL mentioned in the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xu, L.D.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243. [CrossRef]
2. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
3. De La Torre Parra, G.; Rad, P.; Choo, K.R.; Beebe, N. Detecting Internet of Things attacks using distributed deep learning. *J. Netw. Comput. Appl.* **2020**, *163*, 102662. [CrossRef]
4. Malach, E.; Shalev-Shwartz, S. Is Deeper Better Only When Shallow is Good? In Proceedings of the International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Curran Associates Inc.: Red Hook, NY, USA, 2019; Art. no. 577.
5. Vinayakumar, R.; Soman, K.P.; Poornachandran, P. Evaluating effectiveness of shallow and deep networks to intrusion detection system. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics, Manipal, India 13–16 September 2017; pp. 1282–1289.
6. Gamage, S.; Samarabandu, J. Deep learning methods in network intrusion detection: A survey and an objective comparison. *J. Netw. Comput. Appl.* **2020**, *169*, 102767. [CrossRef]
7. Catillo, M.; Pecchia, A.; Villano, U. Botnet Detection in the Internet of Things through All-in-One Deep Autoencoding. In Proceedings of the International Conference on Availability, Reliability and Security, Vienna, Austria, 23–26 August 2022; Art. no. 90.
8. Preuveneers, D.; Rimmer, V.; Tsingenopoulos, I.; Spooren, J.; Joosen, W.; Ilie-Zudor, E. Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study. *Appl. Sci.* **2018**, *8*, 2663. [CrossRef]
9. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Network Anomaly Detection: Methods, Systems and Tools. *IEEE Commun. Surv. Tutorials* **2014**, *16*, 303–336. [CrossRef]

10. Catillo, M.; Pecchia, A.; Villano, U. No more DoS? An empirical study on defense techniques for web server Denial of Service mitigation. *J. Netw. Comput. Appl.* **2022**, *202*, 103363. [CrossRef]
11. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutorials* **2015**, *17*, 2347–2376. [CrossRef]
12. Guerra-Manzanares, A.; Medina-Galindo, J.; Bahsi, H.; Nõmm, S. MedBIoT: Generation of an IoT Botnet Dataset in a Medium-sized IoT Network. In Proceedings of the International Conference on Information Systems Security and Privacy, Valletta, Malta, 25–27 February 2020; SciTePress: Setúbal, Portugal, 2020; pp. 207–218.
13. Ullah, I.; Mahmoud, Q.H. A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks. In Proceedings of the Advances in Artificial Intelligence, Canberra, Australia, 29–30 November 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 508–520.
14. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A.; Lloret, J. Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT. *Sensors* **2017**, *17*, 1967. [CrossRef]
15. Ge, M.; Syed, N.F.; Fu, X.; Baig, Z.; Robles-Kelly, A. Towards a deep learning-driven intrusion detection approach for Internet of Things. *Comput. Netw.* **2021**, *186*, 107784. [CrossRef]
16. Albulayhi, K.; Abu Al-Haija, Q.; Alsuhibany, S.A.; Jillepalli, A.A.; Ashrafuzzaman, M.; Sheldon, F.T. IoT Intrusion Detection Using Machine Learning with a Novel High Performing Feature Selection Method. *Appl. Sci.* **2022**, *12*, 5015. [CrossRef]
17. Ahmad, Z.; Shahid Khan, A.; Nisar, K.; Haider, I.; Hassan, R.; Haque, M.R.; Tarmizi, S.; Rodrigues, J.J.P.C. Anomaly Detection Using Deep Neural Network for IoT Architecture. *Appl. Sci.* **2021**, *11*, 7050. [CrossRef]
18. Zavrak, S.; İskefiyeli, M. Anomaly-Based Intrusion Detection From Network Flow Features Using Variational Autoencoder. *IEEE Access* **2020**, *8*, 108346–108358. [CrossRef]
19. Catillo, M.; Rak, M.; Villano, U. Auto-scaling in the Cloud: Current Status and Perspectives. In Proceedings of the Advances on P2P, Parallel, Grid, Cloud and Internet Computing, Antwerp, Belgium, 7-9 November, 2019; Springer: Berlin/Heidelberg, Germany, 2020; pp. 616–625. .
20. Almiani, M.; AbuGhazleh, A.; Al-Rahayfeh, A.; Atiewi, S.; Razaque, A. Deep recurrent neural network for IoT intrusion detection system. *Simul. Model. Pract. Theory* **2020**, *101*, 102031. [CrossRef]
21. Catillo, M.; Del Vecchio, A.; Pecchia, A.; Villano, U. Transferability of machine learning models learned from public intrusion detection datasets: The CICIDS2017 case study. *Softw. Qual. J.* **2022**, *30*, 955–981. [CrossRef]
22. Taheri, R.; Shojafar, M.; Alazab, M.; Tafazolli, R. Fed-IIoT: A Robust Federated Malware Detection Architecture in Industrial IoT. *IEEE Trans. Ind. Inf.* **2021**, *17*, 8442–8452. [CrossRef]
23. Liu, Y.; Kumar, N.; Xiong, Z.; Lim, W.Y.B.; Kang, J.; Niyato, D. Communication-Efficient Federated Learning for Anomaly Detection in Industrial Internet of Things. In Proceedings of the IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; pp. 1–6.
24. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In Proceedings of the International Conference of Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018.
25. Catillo, M.; Pecchia, A.; Villano, U. Simpler Is Better: On the Use of Autoencoders for Intrusion Detection. In *Quality of Information and Communications Technology*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 223–238.
26. Snoussi, R.; Youssef, H. VAE-Based Latent Representations Learning for Botnet Detection in IoT Networks. *IEEE Access* **2022**, *31*, 4. [CrossRef]
27. Regan, C.; Nasajpour, M.; Parizi, R.M.; Pouriyeh, S.; Dehghantanha, A.; Choo, K.R. Federated IoT attack detection using decentralized edge data. *Mach. Learn. Appl.* **2022**, *8*, 100263. [CrossRef]
28. Tsogbaatar, E.; Bhuyan, M.; Taenaka, Y.; Fall, D.; Gonchigsumlaa, K.; Elmroth, E.; Kadobayashi, Y. DeL-IoT: A deep ensemble learning approach to uncover anomalies in IoT. *Internet Things* **2021**, *14*, 100391. [CrossRef]
29. Khajenezhad, A.; Bashiri, M.A.; Beigy, H. A distributed density estimation algorithm and its application to naive Bayes classification. *Appl. Soft Comput.* **2021**, *98*, 106837. [CrossRef]
30. Al Shorman, A.; Faris, H.; Aljarah, I. Unsupervised intelligent system based on one class support vector machine and Grey Wolf optimization for IoT botnet detection. *J. Ambient Intell. Humaniz. Comput.* **2020**, *11*, 2809–2825. [CrossRef]
31. Kan, X.; Fan, Y.; Fang, Z.; Cao, L.; Xiong, N.N.; Yang, D.; Li, X. A novel IoT network intrusion detection approach based on Adaptive Particle Swarm Optimization Convolutional Neural Network. *Inf. Sci.* **2021**, *568*, 147–162. [CrossRef]
32. Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.A. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.* **2010**, *11*, 3371–3408.
33. Pang, G.; Shen, C.; Cao, L.; Hengel, A.V.D. Deep Learning for Anomaly Detection: A Review. *ACM Comput. Surv.* **2021**, *54*, 38. [CrossRef]
34. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly Detection: A Survey. *ACM Comput. Surv.* **2009**, *41*, 15. [CrossRef]
35. Liu, F.T.; Ting, K.M.; Zhou, Z. Isolation Forest. In Proceedings of the IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422.
36. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and Other Botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]

37. Wan, F.; Guo, G.; Zhang, C.; Guo, Q.; Liu, J. Outlier Detection for Monitoring Data Using Stacked Autoencoder. *IEEE Access* **2019**, *7*, 173827–173837. [CrossRef]
38. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering: An Introduction*; Kluwer Academic: Norwell, MA, USA, 2000.