Botnet Detection in the Internet of Things through All-in-one Deep Autoencoding

Marta Catillo
Università degli Studi del Sannio
Benevento, Italy
marta.catillo@unisannio.it

Antonio Pecchia
Università degli Studi del Sannio
Benevento, Italy
antonio.pecchia@unisannio.it

Umberto Villano
Università degli Studi del Sannio
Benevento, Italy
villano@unisannio.it

ABSTRACT

In the past years Internet of Things (IoT) has received increasing attention by academia and industry due to the potential use in several human activities; however, IoT devices are vulnerable to various types of attacks. Many existing intrusion detection proposals in the IoT leverage complex machine learning architectures, which may provide one separate model per device or per attack. These solutions are not suited to the dynamicity and scale of modern IoT environments. This paper proposes an initial analysis of the problem in the context of deep autoencoders and the detection of botnet attacks. Our findings, obtained by means of the N-BaIoT dataset, indicate that it is relatively easy to achieve impressive detection results by training-testing separate and minimal deep autoenconders on the top of the data individual IoT devices. More important, our all-in-one deep autoencoding proposal, which consists in training a single model with the benign traffic collected from different IoT devices, allows to preserve the overall detection performance obtained through separate autoencoders. The all-in-one model can pave the way for more scalable intrusion detection solutions in the context of IoT.

CCS CONCEPTS

• **Security and privacy** → **Intrusion/anomaly detection and malware mitigation**; **Intrusion detection systems**;

KEYWORDS

IoT, deep learning, botnet, anomaly detection, autoencoder

ACM Reference Format:
Marta Catillo, Antonio Pecchia, and Umberto Villano. 2022. Botnet Detection in the Internet of Things through All-in-one Deep Autoencoding. In *The 17th International Conference on Availability, Reliability and Security (ARES 2022), August 23–26, 2022, Vienna, Austria.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3538969.3544460

1 INTRODUCTION

Internet of Things (IoT) devices underlie many critical assets of our daily lives. Assuring security in face of threats and attacks is a primary concern, as IoT devices can be infected in order to form a botnet that can be leveraged to conduct malicious activities. The body of scientific literature on machine learning and (deep) neural networks applied to intrusion detection systems (IDS) for IoT is huge and ever-increasing. This trend is pushed by the large number of public IoT datasets, such as MedBIoT[1], N-BaIoT[2] and IoTID20[3] –just to mention a few– and the availability of ready-to-use deep learning frameworks (e.g., Keras, TensorFlow, PyTorch). As a result, a wide community of academics and practitioners can conduct measurement studies at the intersection of machine learning and intrusion detection for the IoT. The "pattern" followed by many paper on intrusion detection, deep learning and IoT is typically the same: (i) proposal of an algorithm or architecture based on neural networks (possibly deep ones), (ii) training-testing with one (more) public dataset(s), (iii) demonstration of impressive detection results (typically close to 100%).

We observe that many existing proposals in the area of deep learning and IoT *do not* really address the gap between development and operation. Beside the machine learning exercise, "brilliant" and highly-complex deep networks proposed for intrusion detection so far, such as Convolutional Neural Network (CNN) and Long short-term memory (LSTM), might find no or limited adoption in real-life production IoT environments. In fact, operational IoT security and machine learning are evolving as independent research areas, being addressed by different communities. In order to be usefully deployed in practice, intrusion detection in IoT 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. Given the ever-growing number and heterogeneity of devices and traffic volume, intrusion detection in IoT should pursue simplicity over complexity (e.g., small-footprint neural networks) in order to assure low detection latency, portability and energy efficiency. Another issue pertains to the learning assumptions. For example, recent contributions in the area tend to create individual models per IoT device [13] or per attack [9], which is not suited to the dynamicity and scale of IoT environments and security threats.

This paper proposes an initial exploration of the problem in the context of deep autoencoders (AEs) and the detection of botnet attacks available in the widely-used N-BaIoT dataset, which provides benign and attack traffic data collected with different IoT devices and arranged into individual datasets –one per device– in the form of records of 115 features. Among the wide corpus of existing proposals in intrusion detection for IoT, multiple AEs (possibly complemented by sophisticated feature selection methods)

ARES 2022, August 23–26, 2022, Vienna, Austria

ACM ISBN 978-1-4503-9670-7/22/08…$15.00
https://doi.org/10.1145/3538969.3544460

[1] https://cs.taltech.ee/research/data/medbiot/
[2] http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT
[3] https://sites.google.com/view/iot-network-intrusion-dataset/home

are often used in complex cascade and ensemble configurations. Our proposal develops around the intuition that this complexity is not justified because a single autoencoder is enough to obtain remarkable detection figures. We assess three different AEs consisting of 3, 5 and 7 hidden layers, respectively. AEs are trained in a semi-supervised manner, in that only benign traffic is used for training. Our experiment is twofold. First, we conduct a fine-grain experiment by training the AEs with the benign traffic of the individual IoT devices, separately (i.e., one **separate** model per device); second, we train the AEs with the benign traffic of all the IoT devices (i.e, **all-in-one** model). We assess the detection capability of the AEs –separate and all-in-one model– by means of the typical metrics of recall, precision, false positive rate and F1 score, computed through benign and attack traffic *held-out* from training.

The results indicate that a simple and "minimal" AE of 3 hidden layers is enough to achieve satisfactory results, e.g., recall in the range 0.999-1.0 and precision within 0.990-0.999, in case the AE is trained-tested with the data of the IoT devices, separately. As for the all-in-one model, the AE of 7 hidden layers assures the best results, i.e., 0.999 recall for all the IoT devices, while the recall of the AE of 3 hidden layers ranges between 0.355-0.785 depending on the device. We observe that it is relatively easy to achieve impressive detection figures by training-testing a model on the top of individual devices (i.e., the experimental setting of the original N-BaIoT paper); however, this approach underlies the need for maintaining one model per device, which poses major scalability issues in large-scale IoT networks. Different from this perspective, our all-in-one deep autoencoding approach indicates that it is possible to train a single model with benign traffic collected from different devices, which paves the way for more scalable IoT intrusion detection solutions. It is worth noting that privacy issues due to the adoption of the all-in-one model are not in the scope of this exploratory paper. Our long-term objective is to capitalize on *federated learning* [16] and to leverage the decentralized data concept to cope with privacy facets.

The rest of the paper is organized as follows. Sect. 2 discusses related work in the area. Sect. 3 provides the background on deep autoencoders and datasets. Sect. 4 addresses training and validation of the autoencoders. Sect. 5 presents the results of our study, while Sect. 6 concludes the paper.

2 RELATED WORK

Security challenges pose a significant barrier to the widespread adoption and deployment of IoT devices. The security vulnerabilities introduced by heterogeneity 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 for anomaly detection in the IoT has become extremely important [3]. Al-Fuqaha et al. [1] surveyed some challenges and issues for the design and the deployment of IoT applications. In order to tune and test machine learning techniques, many ready-to-use public intrusion detection **IoT datasets** have been produced. Most of these datasets are collected in synthetic environments under normative conditions and different intrusion scenarios. Some popular public IoT intrusion detection datasets are: **MedBIoT**, **N-BaIoT** [13] and **IoTID20** [17].

Lopez-Martin et al. [12] propose a novel network intrusion detection method specifically developed for an IoT network. The approach is based on a Conditional Variational Autoencoder (CVAE) which 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 [10], 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. Almieani et al. [2] propose a model that 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 shown in [4], this issue might 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, such as the enablement of defense modules [7]. The detection of different classes of anomalies has been recently addressed by means of system log analysis and a deep autoencoder [6]: the proposed approach, called AutoLog, is based solely on a deep autoencoder network without any kind of artifice in the infrastructure.

The work closest to our proposal is Kitsune [14], an unsupervised learning approach to detect attacks online. Kitsune's core algorithm is KitNet, which uses a collection of auto-encoder 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 [13] 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. Finally, in [9] an IoT micro-security add-on is presented that is hosted in the device and uses a CNN model for detecting URL based attacks directed to a client's IoT devices. It is worth pointing out that the aforementioned methods –different from our approach– create individual models per IoT device [14] or per attack [9], which might not be suited to the ever-evolving IoT environments and threats.

3 BACKGROUND, DATASET AND ANOMALY DETECTION METHOD

3.1 Autoencoders (AE)

The core of the proposed method is the use of a **deep autoencoder**. In general, an autoencoder is a specific type of neural network where the *input layer* has the same dimension as the *output layer*. An autoencoder compresses the input into a lower-dimensional representation at the *middle* hidden layer and then it reconstructs the output from the representation. The middle hidden layer of an

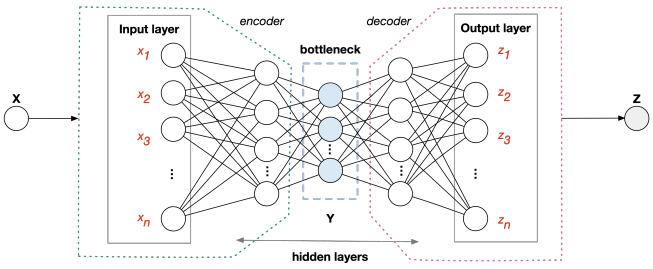

Figure 1: Representation of an autoencoder.

autoencoder is also known as the *bottleneck layer* and its dimension is lower than the input/output layer. Deep learning can be applied to autoencoders. In particular, multiple hidden layers can be used to provide depth: the resulting network is known as *deep* or *stacked autoencoder* [18]. Figure 1 shows the representation of an autoencoder with three hidden layers.

An autoencoder consists of two parts: **encoder** and **decoder**. Let **x** be an input vector of n real numbers $[x_1,x_2,...,x_n]$, 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 real numbers $\mathbf{z} = [z_1, z_2, ..., z_n]$. In the autoencoder terminology, **z** is called the reconstruction of the input vector **x**. Encoding-decoding formulas are given in Eq. 1 and Eq. 2. They represent the case of a simple autoencoder with one hidden layer:

$$y = \sigma(Wx + b) \tag{1}$$

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

where W, W', b and b' are weight matrices and bias vectors, while σ and σ' are activation functions.

When suitably trained, an autoencoder will reconstruct the input as accurately as possible. The quality of the reconstruction is given by the **reconstruction error** (RE), which measures the difference between the reconstructed, i.e., **z**, and the original version of the input, i.e., **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 AE for anomaly detection

An autoencoder can reconstruct accurately, i.e., low reconstruction error (RE), future points "similar" to those used for training. Based on this principle, in order to pursue anomaly detection we train the autoencoder solely by means of normal data points, so that the autoencoder learns a latent subspace of normal data points [15]. After training, the autoencoder will embed a model of the "normal profile", and it can –in principle– identify any instance not conforming to the normal profile as anomalies.

The reconstruction error (RE) is a viable *anomaly indicator*. Since the AE is trained using only normative data points, it will experience (i) *low* RE (good reconstructed representation) for future normative inputs, and (ii) *high* RE (bad reconstructed representation) for future anomalous inputs. More specifically, we use the AE to capture a *normative baseline*: when it tries to reconstruct a data point that is different from the norm –and thus a potential anomaly– it will report an increment of the RE value because it was never trained to reconstruct anomalies. The reconstruction error is a measure of the anomaly degree.

According to Chandola et al. [8] the approach adopted in this paper is termed **semi-supervised** anomaly detection, which means that it does not need anomalies at training time. Moreover, as for any anomaly detection technique assigning a score to data points (RE in our approach), we rely on a cut-off **anomaly threshold** to discriminate normal from anomalous data points. In particular, detection is based on the use of the threshold value: the data points producing RE values under the threshold are considered normative, and those with REs above the threshold are deemed anomalous. Implementation details, including configuration and parametrization of the autoencoder for the datasets in hand, anomaly threshold selection and any other pertinent aspects are addressed in Sect. 4.2.

3.3 Reference dataset: N-BaIoT

The dataset considered in this paper is **N-BaIoT** [13]. It provides a public botnet IoT dataset collected from the **University of California at Irvine (UCI)** machine learning repository. The dataset contains a total number of 7,062,606 benign and attack records gathered in a lab environment that replicates a typical organizational data flow. Traffic data is collected from nine IoT devices, namely a *thermostat* (`Ecobee`), a *baby monitor* (`Philips B120N/10`), a *webcam* (`Samsung SNH 1011 N`), two *doorbells* (`Danmini - Ennio`), and four *security cameras* (`Provision PT-737E - Provision PT-838 - SimpleHome XCS7-1002-WHT - SimpleHome XCS7-1003-WHT`) connected via Wi-Fi to several access points, wire-connected to a central switch which also connects to a router. In order to sniff the network traffic, the port mirroring is performed on the switch, and data is recorded by means of the `Wireshark`[4] tool.

Malicious data is divided into ten attacks carried out by two botnets namely `Mirai` and `BASHLITE` and collected from the IoT devices. In particular, the `BASHLITE` botnet infects Linux-based IoT devices by brute forcing default credentials of devices with open Telnet ports. Once a new bot is connected to the Command and Control (C&C) server and is under its control, this server is able to command the infected device to launch attacks. The `Mirai` botnet, instead, consists of a Command and Control (C&C) server and a server with a scanner and loader. The scanner and the loader scan and identify vulnerable IoT devices, and load the malware to the vulnerable ones. After infection, the device automatically starts scanning the network for new victims while waiting for instructions from the C&C server. The types of `BASHLITE` attacks in the N-BaIoT dataset are: `Scan`, `Junk`, `Flooding` (TCP/UDP), and `COMBO`. Also, `Mirai` botnet comprises: `Scan`, `Ack`, `Syn`, `UDP Flooding`, and `UDP Plain` attacks.

For each device, the data was obtained under both normal operations and attack conditions. In fact, the dataset is released in the form of comma separated values files (`csv`) for each device

[4] https://www.wireshark.org

and each of them has its own benign and attack data. The number of instances varies for every device and attack. Each data point is identified by 115 independent features, plus a class label to be derived from the respective csv filename (e.g., "benign" or "TCP attack"). Each feature models traffic statistics over several temporal windows. Given 23 recorded statistical features (e.g., size of outbound packets –*mean* and *variance*– packet jitter –*mean* and *variance*–, and so forth) and 5 values of decay factor λ= (0.01, 0.1, 1, 3, 5) –100ms, 500ms, 1.5sec, 10sec and 1min– which is used to forget information over time, there are a total of $23 \times 5 = 115$ features.

It is worth pointing out that botnet infections consist of multiple steps such as propagation, bot infection, communication with C&C server, and performing other types of malicious activities [11]. According to [13], it is not enough to determine only the early stages of infection. The N-BaIoT dataset focuses on the last stage of botnet, when IoT bots begin launching attacks.

4 EXPERIMENTAL SETUP

Our experimentation is **twofold**. First, we pursue a "conventional" learning approach, which consists in training an autoencoder with the benign traffic of individual IoT devices, i.e., one separate model per device. Later, different from similar work in the area, we train a unique AE –and therefore **all-in-one**– whose training set is made up by merging the benign traffic of different IoT devices.

4.1 Dataset partitioning

For our experiments we consider the `Provision PT-737E` security camera, the `Samsung SNH 1011 N` webcam, the `Ecobee` thermostat and the `Philips B120N/10` baby monitor. Moreover, the experiments refer to a binary classification scenario. As such, records referring to different types of attacks are considered as belonging to a unique class named `ATTACK`.

It is worth remarking that the N-BaIoT dataset is organized into *separate datasets*, each containing both benign 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 a given IoT device, we adopt a stratified sampling strategy with no replacement. This means that (i) the ratio between benign and attack classes of the original datasets is preserved in the output splits and (ii) each record of the original dataset is assigned to a unique split. For each set of dataset, we obtain:

- *Training set.* It contains only `BENIGN` records and it is meant for training the AE. Labels are removed.
- *Validation set.* It contains only `BENIGN` records and it is used to monitor overfitting and to set the anomaly threshold (more on this later). Again, labels are removed.
- *Test set.* It contains both `BENIGN` and `ATTACK` records. Records in this set are accompanied by the corresponding labels that are checked to evaluate the correctness of the predictions.

Table 1 shows the cardinality of the three sets –training, validation and test– for the first group of experiments, i.e., separate AEs; Table 2, instead, highlights the cardinality of the training, validation and test sets for the second group of experiments, i.e., all-in-one AE, where training/validation sets sum up to the cardinalities of training/validation sets in Table 1. It is worth noting that for each group of experiments the cardinality of the validation set is 10% of the training set. Moreover, test sets are "held-out" from training and they will be used in Sect. 5 for measuring the detection capabilities of the autoencoders.

Table 1: Training, validation and test set size (separate AEs).

IoT device	*training set*	*validation set*	*test set*	
	BENIGN	BENIGN	BENIGN	ATTACK
Provision PT-737E	43 507	4350	9323	114 910
Samsung SNH 1011 N	36 505	3650	7822	48 458
Ecobee	9179	917	1966	123 408
Philips B120N/10	122 667	12 266	26 286	138 511

Table 2: Training, validation and test set size (all-in-one AE).

IoT device	*training set*	*validation set*	*test set*	
	BENIGN	BENIGN	BENIGN	ATTACK
Provision PT-737E			9323	114 910
Samsung SNH 1011 N	211 858	21 185	7822	48 458
Ecobee			1966	123 408
Philips B120N/10			26 286	138 511

4.2 AE design and training

In general, the design of an AE is optimized by selecting and tuning various hyperparameters of the network, which range from the number of layers, neurons per layers, activation functions and so forth. At the time being, there is no specific rule for optimizing the hyperparameters of an AE, and they are mostly determined by the experience and intuition of the designer. Another set of critical hyperparameters pertain to the learning process: epochs, batch size, optimizer algorithm and learning rate. As for any machine learning experiment, our selection was controlled by carrying out experimental tests where we analyzed the outcome of the model –RE in our study– with respect to the **validation set**.

4.2.1 Separate autoencoding. We found that the configuration shown in Table 3 guarantees an effective design, i.e, low RE on the validation set, for the separate AEs setting. The selected AE is made up of **3 hidden layers**. These layers include N-36-6-36-N neurons, where N is the number of features identifying each data point. 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 `BENIGN` data points of each IoT device for 100 epochs with batch size 1024 using the RMSProp optimizer with learning rate value *lr=0.0001*.

4.2.2 All-in-one autoencoding. For the second set of experiments, after testing different designs, i.e., N-36-6-36-N and N-64-36-12-36-64-N, we found that adding **width** and **depth** to the AE is beneficial. In particular, Table 4 shows a successful design for this case study. The chosen AE is made up of **7 hidden layers**. These layers include N-64-36-12-6-12-36-64-N neurons. Again, ReLu has been selected for the encode layer, the decode layer and the bottleneck layer, while for the output layer the Tanh activation function has been

Table 3: Separate autoencoding: layering structure of the AE (all the layers are dense).

layer	activation
Input	-
Hidden 1	ReLU
Hidden 2	ReLU
Hidden 3	ReLU
Output	tanh

Table 4: All-in-one autoencoding: layering structure of the AE (all the layers are dense).

layer	activation
Input	-
Hidden 1	ReLU
Hidden 2	ReLU
Hidden 3	ReLU
Hidden 4	ReLU
Hidden 5	ReLU
Hidden 6	ReLU
Hidden 7	ReLU
Output	tanh

used. We train the all-in-one AE on BENIGN the data points across all IoT devices, setting the number of epochs to 100, with batch size 1024 and using again the RMSProp optimizer with learning rate value *lr=0.0001*.

4.2.3 Notes on Training. Our semi-supervised training process –*no anomalies* at training time– is crucial for both sets of experiments. During the training phase, the weights and biases of the encoder and decoder are calculated and optimized with respect to BENIGN training data. As outlined in Sect. 3, during training the autoencoder learns the relationships among the features in the training set. As for any typical deep learning experiment, the AE neurons are randomly initialized at the beginning of the training process, and input data are presented in batches for a given number of epochs, 1024 and 100 in our study. The goal of training is 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 monitor the occurrence of overfitting. The loss describes the objective that the AE tries to reach. Since our goal is to reconstruct the input as accurately as possible, we compute the loss as the **mean squared error** at the output units; this matches the definition of reconstruction error (RE) above.

4.2.4 Threshold selection. If an unseen data point is given to a trained AE, it can reconstruct it with low RE as long as the data point is similar to those encountered during training. If this condition does not hold, the AE cannot reconstruct the data point and the related RE value is high. Therefore, any divergence from a "benign behavior" could lead to a high reconstruction error, making it possible to recognize ATTACK records. As for many anomaly detection techniques, we apply a threshold to the RE in order to discriminate *good* from *bad* reconstructions and, in turn, anomalies. We choose the threshold value by using BENIGN instances, and hence the training and validation procedure is semi-supervised.

The *anomaly threshold* is set with the following approach. First, we apply an outlier detection algorithm to the BENIGN data points of the validation set, which makes it possible to discriminate inliers from outliers. Second, inliers and outliers are fed to the autoencoder in order to obtain the corresponding REs. The threshold is the highest RE value assuring that the number of inliers falling below that RE is higher then the number of outliers. As the model, the threshold is an output of the learning phase.

4.2.5 Implementation. AEs have been implemented in Python using the Keras[5] (Version 2.7.0) library with TensorFlow[6] (Version 2.7.0) as backend. All our experiments are conducted on a MacBook Pro with an Intel Core i5 2.6 GHz processor and 8 GB of RAM. It is worth pointing out that the training phase for a single AE takes around 1 minute in the worst case. This time could be greatly reduced using more powerful or ad hoc hardware (i.e., by GPU acceleration).

5 RESULTS

This section presents our experimental results. We focus on the following two points: (i) the performance of separate AEs, individually trained with the BENIGN traffic of each IoT device, and (ii) the performance of the all-in-one AE (i.e., **all-in-one autoencoding**) trained once with the BENIGN traffic of all devices.

We compute the typical metrics of *recall* (R), *precision* (P), *false positive rate* (FPR), and *F1 score* as follows:

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

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

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 BENIGN points whose RE is lower than the threshold.

5.1 Separate autoencoding

We process the test sets of the datasets in hand with the four AEs trained as described in Section 4. Again, we focus on the following IoT devices: Provision PT-737E security camera, Samsung SNH 1011 N webcam, Ecobee thermostat and Philips B120N/10 baby monitor. As highlighted in Section 4, we evaluate the performance of each AE by means of the test sets of the corresponding device. RE values produced by the four different AEs are accompanied by the labels, which are used for evaluation.

For example, Figure 2 shows the REs for the Philips B120N/10 baby monitor device. It refers to the Philips B120N/10 baby monitor test set (i.e, the split of BENIGN and ATTACK points *held out* from training). Each data point is marked by a ∘. In order to separate BENIGN points from ATTACK points, we superimpose a vertical continuous line. On the left side of the vertical line there are BENIGN

[5]https://keras.io/
[6]https://www.tensorflow.org/

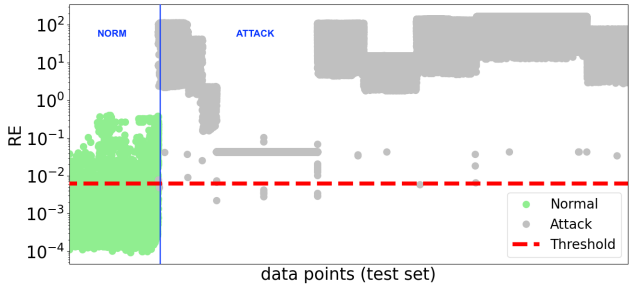

Figure 2: Separate autoencoding: RE of the test set for the `Philips B120N/10` baby monitor device.

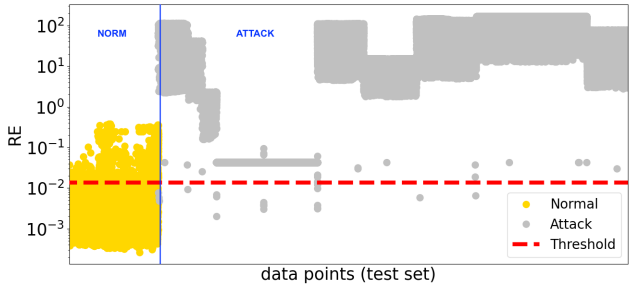

Figure 3: All-in-one autoencoding: RE of the test set for the `Philips B120N/10` baby monitor device.

IoT device	recall	precision	FPR	F1 score
`Provision PT-737E`	0.999	0.999	0.0086	0.999
`Samsung SNH 1011 N`	0.999	0.997	0.0136	0.998
Ecobee	1.0	0.999	0.0274	0.999
`Philips B120N/10`	0.999	0.990	0.0503	0.995

Table 5: Separate autoencoding: evaluation metrics (AE configuration $N-36-6-36-N$).

IoT device	recall	precision	FPR	F1 score
`Provision PT-737E`	0.999	0.983	0.2096	0.991
`Samsung SNH 1011 N`	0.999	0.993	0.0421	0.996
Ecobee	0.999	0.999	0.0503	0.999
`Philips B120N/10`	0.999	0.993	0.0352	0.996

Table 6: All-in-one autoencoding: evaluation metrics ($N-64-36-12-6-12-36-64-N$) .

points, while on the right side of the vertical line there are `ATTACK` points. A semi-logarithmic scale (x-axis in linear scale and y-axis in log scale) is used to better visualize the RE values. The y-axis is the RE; the x-axis is the id of the points in the test set. The horizontal dashed line identifies the anomaly threshold obtained after training.

Figure 2 shows that almost all `ATTACK` points have a high RE and are well above the threshold. The `BENIGN` data points, instead, have a low RE and, except for a few exceptions, they are below the threshold. It is worth noting that rendering and resolution of Figure 2 overemphasize `BENIGN` points above the threshold; in practice, the number of `BENIGN` points above the threshold is 1322, which is significantly lower than those falling below the threshold (24964).

Table 5 provides the evaluation metrics for all the selected IoT devices. Results prove that individual AEs are notable in detecting attacks based on the recall, precision, false positive rate and F1 score values. For example, recall is in the range 0.999-1.0 and precision within 0.990-0.999. While the best result is obtained with the `Provision PT-737E` device, the outcome of the remaining devices is notable, since each AE achieves a recall, precision and an F1 score very close to 1.0 –exactly 1.0 for the `Ecobee` device– on all three devices, while maintaining a reasonably low false positive rate.

Finding: Different from similar proposals in the area, which rely on complex cascades and ensembles of autoencoders –possibly complemented by feature selection methods– if not other schemes, such as CNNs and LSTMs, a "minimal" and simple autoencoder with 3 hidden layers is more than enough to obtain remarkable results when train-test is done for each device, separately.

While the results are outstanding, the hypothesis of training an AE for each device remains unrealistic, especially if the intrusion detection system is intended to be deployed in a large-scale and dynamic IoT environment.

5.2 All-in-one autoencoding

An all-in-one detector is certainly more viable in a complex IoT scenario. We evaluate the performance of the all-in-one AE, trained with the `BENIGN` traffic of all IoT devices, by running the test sets related to the `Provision PT-737E` security camera, `Samsung SNH 1011 N` webcam, `Ecobee` thermostat and `Philips B120N/10` baby monitor. As for the previous experiment, Figure 3 shows the REs for the `Philips B120N/10` baby monitor device. Also in this case the horizontal dashed line well discriminates `BENIGN` data points from the `ATTACK` ones. We observe that the `ATTACK` points are almost all over the threshold. Almost all `BENIGN` points are below the threshold; the number of `BENIGN` points above the threshold –thus misclassified– is 926 out of 26286. Table 6 provides the evaluation metrics of the AE with 7 hidden layers. For the sake of completeness, we show in Table 7 and Table 8 the evaluation metrics for two "smaller" configurations (i.e., $N-36-6-36-N$ and $N-64-36-12-36-64-N$), which are less performing.

It is worth pointing out that the all-in-one model with 7 hidden layers is comparable to the separate autoencoding solution. The results are outstanding especially by testing the AE with the test set of the `Samsung SNH 1011 N` webcam, `Ecobee` thermostat and `Philips B120N/10` baby monitor; for `Provision PT-737E` performance is worse, especially in terms of FPR. Most notably, by using the same AE architecture of the separate autoencoding experiment (retrained with the all-in-one dataset), performance degrades. For example, the recall dramatically drops to 0.355 for the `Samsung SNH 1011 N` webcam device. The same is true for a slightly deeper and wider network ($N-64-36-12-36-64-N$). This indicates that, with respect to the problem addressed and data in hand, *deepening* and *widening* the autoencoder can improve anomaly detection in the all-in-one setting.

IoT device	recall	precision	FPR	F1 score
Provision PT-737E	0.727	0.999	0.0087	0.842
Samsung SNH 1011 N	0.355	0.989	0.0233	0.522
Ecobee	0.757	0.999	0.0371	0.861
Philips B120N/10	0.785	0.991	0.0342	0.876

Table 7: All-in-one autoencoding: evaluation metrics ($N - 36 - 6 - 36 - N$).

IoT device	recall	precision	FPR	F1 score
Provision PT-737E	0.727	0.998	0.0127	0.841
Samsung SNH 1011 N	0.355	0.983	0.0369	0.521
Ecobee	0.757	0.998	0.0600	0.861
Philips B120N/10	0.785	0.992	0.0302	0.876

Table 8: All-in-one autoencoding: evaluation metrics ($N - 64 - 36 - 12 - 36 - 64 - N$).

Finding: The *all-in-one* AE solution assures remarkable detection figures when compared to the separate autoencoders. Although the AE is a wider and deeper network with respect to separate autoencoding, the training time is still negligible –in the order of minutes on a computer laptop– and the detection latency per record is acceptable (about 1 microsecond per record).

The results indicate that it is possible to train a single model with benign 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.

6 CONCLUSION

IoT plays a key role in our lives; however, due to some factors, such as the vulnerabilities of devices, IoT applications are attack prone. The ever-growing sophistication of the attacks has attracted a significant interest by the research community, which focused on specialized machine learning attacks detection mechanisms. Frequently, IoT intrusion detectors are implemented by means of deep learning techniques with individual models per IoT devices or per attack. These assumptions might be not suited to high-scalable and dynamic IoT environments.

This paper proposed an initial solution to the problem in the context of deep autoencoders (AEs) and the detection of botnet attacks available in the widely-used N-BaIoT dataset. Our results show that it is relatively easy to achieve remarkable detection results by training-testing a model on the top of individual devices; however, this poses a major issue in large-scale IoT networks. Our all-in-one deep autoencoding approach proves that it is possible, by preserving the overall performance, to train a single model with the benign traffic collected from different devices. This allows the setup of more scalable intrusion detection solutions in the context of IoT.

We are aware that the study is based on one dataset; however, we are not striving for general findings at this stage, as the findings of this paper should be contextualized with respect to the attacks and devices of N-BaIoT. As future work we will investigate the possibility of fitting our solution to the modern context of the *federated learning*, which is a widely-used solution in the anomaly detection context [16]. We will also extend the analysis to further datasets, including traditional network datasets [5], attacks and victim devices.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials* 17, 4 (2015), 2347–2376.

[2] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque. 2020. Deep recurrent neural network for IoT intrusion detection system. *Simulation Modelling Practice and Theory* 101 (2020), 102031.

[3] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. 2014. Network Anomaly Detection: Methods, Systems and Tools. *IEEE Communications Surveys Tutorials* 16, 1 (2014), 303–336.

[4] M. Catillo, A. Del Vecchio, A. Pecchia, and U. Villano. 2022. Transferability of machine learning models learned from public intrusion detection datasets: the CICIDS2017 case study. *Software Quality Journal* (2022).

[5] M. Catillo, A. Pecchia, M. Rak, and U. Villano. 2021. Demystifying the role of public intrusion datasets: A replication study of DoS network traffic data. *Computers & Security* 108 (2021), 102341.

[6] M. Catillo, A. Pecchia, and U. Villano. 2022. AutoLog: Anomaly detection by deep autoencoding of system logs. *Expert Systems with Applications* 191 (2022), 116263.

[7] M. Catillo, A. Pecchia, and U. Villano. 2022. No more DoS? An empirical study on defense techniques for web server Denial of Service mitigation. *Journal of Network and Computer Applications* 202 (2022), 103363.

[8] V. Chandola, A. Banerjee, and V. Kumar. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.* 41, 3, Article 15 (2009).

[9] G. De La Torre Parra, P. Rad, K. R. Choo, and N. Beebe. 2020. Detecting Internet of Things attacks using distributed deep learning. *Journal of Network and Computer Applications* 163 (2020), 102662.

[10] M. Ge, N. F. Syed, X. Fu, Z. Baig, and A. Robles-Kelly. 2021. Towards a deep learning-driven intrusion detection approach for Internet of Things. *Computer Networks* 186 (2021), 107784.

[11] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. 2017. DDoS in the IoT: Mirai and Other Botnets. *Computer* 50, 7 (2017), 80–84.

[12] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. 2017. Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT. *Sensors* 17, 9 (2017), 1967.

[13] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici. 2018. N-BaIoT-Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Computing* 17, 3 (2018), 12–22.

[14] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *Proc. Network and Distributed System Security Symposium*. USENIX.

[15] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel. 2021. Deep Learning for Anomaly Detection: A Review. *ACM Comput. Surv.* 54, 2, Article 38 (2021).

[16] D Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor. 2018. Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study. *Applied Sciences* 8, 12 (2018), 2663.

[17] I. Ullah and Q. H. Mahmoud. 2020. A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks. In *Advances in Artificial Intelligence*, Cyril Goutte and Xiaodan Zhu (Eds.). Springer, 508–520.

[18] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research* 11 (2010), 3371–3408.