Network Anomaly Detection Methods in IoT Environments via Deep Learning: A Fair Comparison of Performance and Robustness

Giampaolo Bovenzi[a], Giuseppe Aceto[a], Domenico Ciuonzo[a], Antonio Montieri[a], Valerio Persico[a], Antonio Pescapè[a]

[a] *University of Napoli Federico II, Department of Electrical Engineering and Information Technologies (DIETI), Via Claudio 21, Naples, 80125, Italy*

Abstract

The Internet of Things (IoT) is a key enabler in closing the loop in Cyber-Physical Systems, providing "smartness" and thus additional value to each monitored/controlled physical asset. Unfortunately, these devices are more and more targeted by cyberattacks because of their diffusion and of the usually limited hardware and software resources. This calls for designing and evaluating new effective approaches for protecting IoT systems at the network level (Network Intrusion Detection Systems, NIDSs). These in turn are challenged by the heterogeneity of IoT devices and the growing volume of transmitted data.

To tackle this challenge, we select a *Deep Learning* architecture to perform *unsupervised early anomaly detection*. With a data-driven approach, we explore in-depth multiple design choices and exploit the appealing structural properties of the selected architecture to enhance its performance. The experimental evaluation is performed on two recent and publicly available IoT datasets (IoT-23 and Kitsune). Finally, we adopt an *adversarial approach* to investigate the robustness of our solution in the presence of *Label Flipping* poisoning attacks. The experimental results highlight the improved performance of the proposed architecture, in comparison to both well-known baselines and previous proposals.

Keywords: Anomaly Detection, Deep Learning, Internet of Things, Intrusion Detection System, Network Security, Robustness.

1. Introduction

The Internet of Things (IoT) is the (most exposed) forefront of the latest industrial revolution, permeating both production systems and post-market management and monitoring of goods. It constitutes the enabler in closing the loop of Cyber-Physical Systems (CPSs), providing geographically-distributed intelligence and thus additional value to each monitored/controlled physical asset, as well as the system of which they are part. Because of their increasingly pervasiveness, IoT devices are estimated to count 55.7B units by 2025,[1] pushing the need for IoT-tailored security solutions. Indeed, these devices are characterized by (economical and physical) constraints, leading to limited hardware and software resources. This reason, together with their distributed nature and their key role in current CPSs evolution, makes IoT devices a *primary target for cyberattacks* [1, 2]: 1.5B attacks targeted IoT devices in the 1H 2021, doubling their estimate with respect to the 1H 2020.[2]

In general, systems performing attack detection (Intrusion Detection Systems, IDSs) can be implemented both at the device level (host-based IDS) and at the network level (Network IDS or NIDS): for the aforementioned characteristics of IoT devices, NIDSs are usually preferred. Moreover, the high heterogeneity of IoT devices and the growing volume of transmitted data to be analyzed in quasi-real-time make the design and evaluation of effective IDSs in IoT environments *particularly challenging*. Equally important, to grant pro-active response against novel (unknown) attacks, *unsupervised Anomaly Detection (AD)* represents a key milestone.

To respond to this urge, in recent years the research has focused on approaches based on neural networks and more specifically on *Deep Learning* (DL) ones, for their capability of effectively performing feature extraction (or limiting it), thus *avoiding (or reducing) costly and slow human expert involvement*. Moreover, *biased* inputs—erroneously inflating classification performance but not meaningful in realistic scenarios—have been employed for the design and evaluation of many of these approaches [3]. Linked to this issue and given the *black-box nature* of such neural-network-based approaches, the understanding of their behavior (and thus the reliability of results) has been barely investigated. Finally, almost all relevant works focus on *post-mortem* analysis and assume *ideal conditions*, as they use flow-based inputs (summary data over whole communications) and hypothesize a "clean" (viz. attack-free) benign dataset for training, severely limiting the usefulness of proposed solutions for protecting CPSs in a real-world adversarial setting.

Accordingly, in this work, we tackle and solve these limitations by providing the following **main contributions**, which capitalize on several design and evaluation choices.

- We select state-of-art *DL architectures* to perform *unsupervised AD*, namely "classic" AutoEncoders (AEs) and the recent KitNET proposal [4], to face the need for rapid adaptation to both new devices and new (unknown) attacks. In designing the architectures, we adopt advanced approaches (ensemble learning) to reach improved performance and obtain operational advantages in terms of mod-

[1] `https://bit.ly/idc-future-of-industry-ecosystems`
[2] `https://bit.ly/kaspersky-iot-attacks-doubling`

ularity and thus adaptability to fast-evolving scenarios.

- We adopt packet-level processing to be able to perform *early AD*, obtaining a response for a bidirectional flow after just 4 packets. To this aim, we investigate the impact of different design factors, such as the *number of packets* considered per biflow and the *depth of the DL architectures* (both with operational significance).

- We perform an in-depth exploration of the *distance metrics* used to learn and assess the inner representation of the input data: leveraging Mahalanobis-based distances, ablation studies, and comparative evaluations on different datasets, we find the best-performing (and the most stable) parameter set.

- We propose and evaluate *multiple enhancements of the original KitNET*, attaining better detection performance and improved robustness: (*i*) ensemble equalization, (*ii*) changing the output stage reconstruction target, and (*iii*) ensemble normalization.

- We compare the performance of the designed DL architectures with *standard Machine Learning (ML) techniques* performing AD: Isolation Forest, Local Outlier Factor, and One-Class Support Vector Machine.

- We assess the robustness of the proposed detectors in adversarial (more realistic) scenarios. In detail, we conduct a *Data Poisoning Attack* (specifically, a *Label Flipping Attack*) to gradually degrade the training set, where the attacker can affect the traffic on the observed network during the (periodically needed) re-training of the NIDS.

- We perform the experimental evaluation on *two real, recent, and publicly available IoT-traffic datasets* [4, 5] using only *unbiased* inputs. The experimental results validate our goal of improving both well-known baselines and previous proposals.[3]

The rest of the paper is organized as follows. Section 2 surveys the recent application of DL-based techniques to network AD in IoT environments, positioning our contribution against related literature. Section 3 describes the considered AD-based methodology; the experimental setup and the related experimental results are reported in Secs. 4 and 5, respectively. Section 6 ends the paper with conclusions and future directions of research.

2. Related Work

Related works designing Network IDSs mainly leverage two main approaches: AD or Misuse Detection (MD). The former aims at modeling anomalies as deviations (i.e. outliers) from the profile of benign traffic, while the latter at directly identifying patterns of known attacks. Given the increasing adoption of ML and DL approaches for designing effective NIDSs, the main advantage of AD methods is that they are trained only on benign traffic, whereas MD ones require both benign and malicious samples [19, 20, 21, 22].

Table 1 summarizes the most recent works performing AD, positioning our paper against the latter according to different axes. Specifically, we consider recent works published in the **last five years** and categorize them by highlighting their key aspects. The last row summarizes the present paper. Firstly, we focus on **unsupervised AD** approaches that do not require malicious samples during training. In this regard, we underline that some works combine both unsupervised and supervised techniques [11], also at different levels of the detection/classification architecture in a hierarchical fashion [12, 13].

First of all, we can notice that the works aiming at detecting cyberattacks against **IoT** devices employ different recent datasets (e.g., Kitsune [4], N-BaIoT [8], Bot-IoT [20], IoT-23 [5]) as opposed to other works who commonly leverage the "traditional" KDD99 [23] or NSL-KDD [24]. Unfortunately, *the latter datasets hardly exhibit a current real-world network traffic profile*, particularly considering that they were collected more than two decades ago.

We also underline that almost all related works leverage different variants of the deep AE (see the **DL** column) which outperform the classic outlier detector models (e.g., Isolation Forest, One-Class Support Vector Machine, Local Outlier Factor) reaching higher detection rate and incurring very low error in confirming a normal behavior. Interestingly, the usage of a shallow (i.e. non-deep) AE to detect Distributed Denial of Service (DDoS) attacks [25] is only investigated in [14]—achieving better performance than considered baselines (i.e. up to 82% detection with 0% false-positive rate)—in Zhu et al. [18]—testing both DL and shallow methods for AD—and in [4]—proposing an ensemble of shallow AEs.

Similarly, regarding the traffic segmentation adopted, we observe that bidirectional flows (briefly biflows) and single packets are the most common choices as **traffic objects**. In addition to these latter, HTTP/HTTPS requests [7] and coarse-grained sequences of flows [10] are also employed. Indeed, such choices depend on the specific scenario (e.g., attacks against web applications [7]) or the specific technique and input data considered (e.g., generation of "sentences" representing a conversation between computers [10]).

Unfortunately, also when considering the finest-grained traffic objects (i.e. single packets), the choice of **input data** can prevent **early AD**. As relevant examples, Mirsky et al. [4] and Meidan et al. [9] compute (context and statistical) features based on time windows going up to one minute into the past for each packet to analyze. On the other hand, when leveraging input data suited for early AD, as in [18], these encompass biased fields (e.g., local IP/ETH addresses or source/destination ports) that are likely to inflate AD performance [26]. Interestingly, in the case of biflows and coarse-grained traffic objects, most works employ "post-mortem" features, namely they manually engineer statistics on the sets of packet/payload lengths,

[3] Preliminary results have been published as a conference publication [6]. Detailed contributions and positioning with respect to our previous work are discussed in Sec. 2.

Table 1: Related papers using unsupervised Machine Learning and Deep Learning approaches for Anomaly Detection. The papers are ordered by year. The last row summarizes the present work. Acronyms' meaning is reported at the bottom of the table.

Paper	**Dataset**	**IoT**	**DL**	**TO**	**Input Data**	**EAD**	**AD Technique**	**R**
Mac et al. [7], 2018	CSIC2010	○	●	H	HTTP/HTTPS request tokens		AE, OC-SVM†, IF†, LSTM†	
Madani and Vlajic [8], 2018	NSL-KDD	○	●	B	Flow-based statistics		AE, PCA†	✓
Mirsky et al. [4], 2018	Kitsune*	●	◐	P	Time window-based statistics		KitNET, AE†	
Meidan et al. [9], 2018	N-BaIoT*	●	●	P	Time window-based statistics		AE, OC-SVM†, IF†, LOF†	
Radford et al. [10], 2018	ISCXIDS2012	○	●	S	Per-protocol #bytes & L4 ports		BiLSTM	✓
Andresini et al. [11], 2019	NSL-KDD	○	●	B	Flow-based statistics		AE, CNN†, LSTM†, RNN†	
Khan et al. [12], 2019	KDD99, UNSW-NB15	○	●	B	Flow-based statistics		AE	
Bovenzi et al. [13], 2020	Bot-IoT	●	●	B	Flow-based statistics of the first N_p packets	✓	M2-DAE, AE†	
Yang et al. [14], 2020	SYNT*, MAWI, UNB2017	○	○	B	Flow-based statistics		AE, PCA†, DT†, IF†, OC-SVM†	
Vu et al. [15], 2020	N-BaIoT	●	●	P	Time window-based statistics		MAE, MDAE, MVAE, AE†, DAE†, VAE†, DBN†, RF†	
Kye et al. [16], 2022	NSL-KDD, CIC-IDS2018	○	●	B	Flow-based statistics		AE	
Yang and Hwang [17], 2022	UNSW-NB15	○	●	B	Flow-based statistics		AE	
Zhu et al. [18], 2022	Kitsune, CICIDS2017, MAWILAB, UNSW-NB15	◐	◐	P	Per-packet L2-L4 fields	✓	KitNET, IF, LR, MLP	✓
Bovenzi et al. [6], 2022	Kitsune	●	●	B	Flow-based statistics		AE, KitNET	✓
This paper	Kitsune, IoT-23	●	●	B	Flow-based statistics of the first N_p packets	✓	AE-1/2/3, KitNET-1/2/3, OC-SVM†, IF†, LOF†	✓

IoT: Internet of Things. **DL**: Deep Learning. **TO**: Traffic Object: Biflow (B), Flow (F), HTTP/HTTPS request (H), Packet (P), Sequence of Biflows (S).
EAD: Early Anomaly Detection. **R**: Robustness against data poisoning attacks.
Anomaly Detection Technique: AutoEncoder (AE), Bidirectional LSTM (BiLSTM), Convolutional Neural Network (CNN), Denoising AE (DAE), Deep Belief Network (DBN), Decision Tree (DT), Isolation Forest (IF), Local Outlier Factor (LOF), Linear Regressor (LR), Long Short-Term Memory (LSTM), Multimodal Deep AE (M2-DAE), Multidistribution AE (MAE), Multidistribution DAE (MDAE), MultiLayer Perceptron (MLP), Multidistribution VAE (MVAE), One-Class Support Vector Machine (OC-SVM), Principal Component Analysis (PCA), Random Forest (RF), Recurrent Neural Network (RNN), Variational AE (VAE).
"$*$" symbol indicates self-generated datasets; "†" symbol indicates baselines; ● present, ◐ partial, ○ lacking.

inter-arrival times, etc. regarding the whole traffic object (i.e. needing to wait for its end) [7, 10, 14, 15, 17], or they leverage already preprocessed datasets (e.g., KDD-Cup-99, NSL-KDD) [8, 11, 12, 16]. Conversely, in our previous works, we calculate statistical features related to the *first N_p packets* (up to 25), aiming to attain early AD [13], or perform a robustness analysis when leveraging "post-mortem" features [6].

Referring to the specific **AD technique** applied, all the works—with the exception of [10]—employ variants of the AE. Indeed, as mentioned before, AEs are inherently simple neural network models, composed of low-complexity layers, and thus demanding limited computing resources; also, we recall that AEs naturally allow to exploit the unsupervised learning paradigm, thus not requiring labeled traffic for training. Among the various proposals, it is worth mentioning KitNET [4], a double-stage ensemble of shallow AEs, where each AE belonging to the first stage is fed with a subset of the features; then the reconstruction errors from the first stage are fed to a single AE at the second stage which implements a voting procedure. Moreover, in [13], we have first proposed M2-DAE (Multi-Modal Deep AutoEncoder), a multimodal variant of an AE-based approach for AD having similar performance but reduced model size with respect to a common deep AE. In addition to other AE variants (e.g., multi-distribution, variational, and denoising AEs) [15], several ML/DL approaches are commonly used as baselines for comparison (flagged with a † in Tab. 1), usually showing poorer performance than (deep) AE-based ones. They include both ML methods (e.g., Isolation Forest, Local Outlier Factor, MultiLayer Perceptron, One-Class Support Vector Machine, and Random Forest) and baseline DL architectures (e.g., Convolutional Neural Network, Long Short-Term Memory, and Recurrent Neural Network).

In the last column of Tab. 1, we highlight the works evaluating the **robustness** of AD approaches against **data poisoning** attacks: Label Flipping Attacks (viz. adversarial contamination during training) [6, 8, 10] and generation (via e.g., Generative Adversarial Networks) of forged data packets to evade NIDS detection [18]. Indeed, due to the unsupervised nature of the

training phase of AEs, the injection of malicious data during training is more effective than in the supervised scenario, since no labeling is required and malicious traffic could be injected without knowledge of the underlying model and features (i.e. black-box attacks). We underline that similar investigations are conducted in CPSs by evaluating the vulnerability of anomaly-detecting AEs to different types of adversarial attacks in e.g., water treatment [27] and industrial control systems [28]. Despite being potential targets of attack by malicious actors, such scenarios are beyond the scope of this paper. Finally, to the best of our knowledge, existing works barely optimize model hyperparameters without providing hints about the model generalization capabilities [4, 14].

Contribution Positioning. In this work, in accordance with the recent literature [4, 9, 13, 15, 18], we rely on datasets *truly* containing cyberattacks targeting IoT devices (i.e. Kitsune [4] and IoT-23 [5]). Secondly, we design and evaluate a *comprehensive* set of variants of single/ensemble shallow/deep AEs (as opposed to [16, 17]) and compare them with ML-based AD approaches. Thirdly, all the approaches considered herein are fed with *unbiased* input data allowing us to attain early AD, a *practical assumption* which contrasts most of the reviewed literature (i.e. save from [13, 18]). Finally, our study is complemented with a robustness assessment of the effects associated with data poisoning (which well models both intentional and unintentional attacks), which has been tackled only by a relatively small part of the recent literature [8, 10, 18, 6]. Therefore, to the best of our knowledge, we provide an all-around analysis in terms of both the effectiveness and robustness of various AD approaches in IoT environments not present in the related literature to date.

Furthermore, with respect to our previous proposal [6], in this paper: (*i*) we compare AE-based detectors with ML-based baselines (i.e. Isolation Forest, Local Outlier Factor, and One-Class Support Vector Machine); (*ii*) we evaluate the usage of deep AEs with different sizes at both stages of KitNET (resulting in `KitNET-2` and `KitNET-3`); (*iii*) we implement the following enhancements to the KitNET architecture: (*a*) ensemble equalization, (*b*) changing the output stage reconstruction target, and (*c*) ensemble normalization; finally (*iv*) we evaluate the robustness of AD via different distance metrics (i.e. $\texttt{RMSE}_{\texttt{INT}}$/$\texttt{RMSE}_{\texttt{EXT}}$, $\texttt{MHLN}_{\texttt{INT}}$/$\texttt{MHLN}_{\texttt{EXT}}$, `SAP`, and `NAP`).

3. Methodology

This section aims at introducing the proposed methodology. Section 3.1 presents the DL detectors based on reconstruction error, while Sec. 3.2 those based on ML algorithms. The selection of the detection threshold is deepened in Sec. 3.3. Finally, Sec. 3.4 discusses the threat model.

3.1. Reconstruction-based DL Detectors

Autoencoder-based Detectors. An *AutoEncoder* (AE) is a peculiar neural network able to self-reproduce data. It is made of two principal components: an encoder $g(\cdot)$–which computes a compression of the input vector ($\boldsymbol{x}$) into a latent space; a decoder $f(\cdot)$–which tries to reconstruct the input vector starting

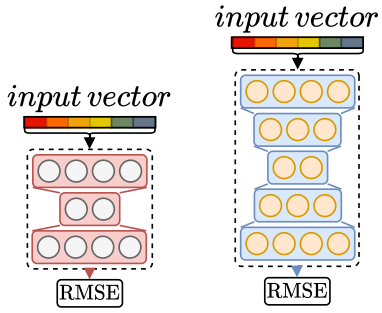


Figure 1: Structure of shallow AE (left) vs deep AE (right).

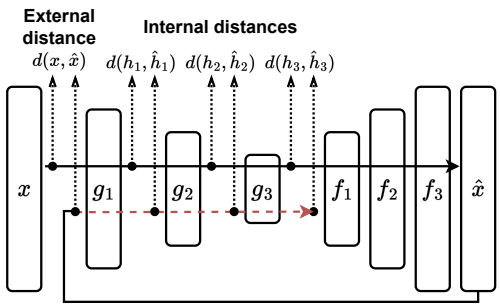


Figure 2: Datapath of a generic AE architecture. The second half round of reconstruction (dashed red line) is used to compute `SAP`, `NAP`, and partial distance metrics.

from the encoder output (i.e. the latent space). An AE can be broadly considered either as a *shallow* or a *deep* AE, with the main difference between these represented by the number of encoding/decoding layers (see left and right side of Fig. 1, respectively): the shallow AE has a single encoder/decoder layer whereas a deep AE presents multiple encoding/decoding layers (i.e. $g(\cdot) = g_\ell \circ \ldots \circ g_1(\cdot)$ and $f(\cdot) = f_\ell \circ \ldots \circ f_1(\cdot)$). The whole reconstruction (encoding plus decoding) path is generically denoted with $\hat{\boldsymbol{x}} = (f \circ g)(\boldsymbol{x})$. Once the reconstruction is obtained, the anomaly score is calculated as $a_{\text{ae}}(\boldsymbol{x}) = \mathcal{L}(\hat{\boldsymbol{x}}, \boldsymbol{x})$, where $\mathcal{L}(\cdot)$ is commonly the squared loss. Despite these models can be used in a standalone fashion, recent proposals [4, 13] showed the advantage of considering shallow AEs in an ensemble architecture.

Reconstruction Errors. According to [29], herein we adopt a reconstruction error taking into account also the terms originating from the hidden layers of the AEs and possibly based on the adoption of Mahalanobis distance. In brief, the idea is to combine the classic output reconstruction error with the error committed at the hidden layers in a double round of reconstructions. For convenience, before proceeding and referring to a generic (deep) AE, we give the following *auxiliary definitions*: (*i*) $\boldsymbol{h}_i(\boldsymbol{x}) = (\boldsymbol{g}_i \circ \cdots \circ \boldsymbol{g}_1)(\boldsymbol{x})$ and (*ii*) $\hat{\boldsymbol{h}}_i(\boldsymbol{x}) = (\boldsymbol{g}_i \circ \cdots \circ \boldsymbol{g}_1)(\hat{\boldsymbol{x}}) = (\boldsymbol{g}_i \circ \cdots \circ \boldsymbol{g}_1)((\boldsymbol{g} \circ \boldsymbol{f})(\boldsymbol{x}))$. The former hold for $i \geq 1$. In the special case $i = 0$, it holds $\boldsymbol{h}_0(\boldsymbol{x}) = \boldsymbol{x}$ and $\hat{\boldsymbol{h}}_0(\boldsymbol{x}) = \hat{\boldsymbol{x}}$.

The resulting reconstruction errors are referred to as *Simple Aggregation along Pathway* (SAP) and *Normalized Aggregation along Pathway* (NAP). Specifically, SAP is computed based on the Root Mean Squared Error (RMSE) on the concatenation of output and hidden layers reconstructions, whereas

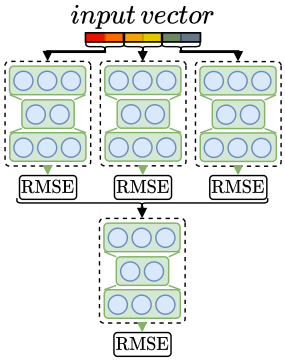


Figure 3: KitNET structure.

NAP is computed on the same reconstructions but with the Mahalanobis distance. In other terms, the anomaly score based on SAP is:

$$a_{\text{sap}}(\boldsymbol{x}) = \sum_{i=0}^{\ell} \left\|\boldsymbol{h}_i(\boldsymbol{x}) - \hat{\boldsymbol{h}}_i(\boldsymbol{x})\right\|^2 = \left\|\boldsymbol{h}(\boldsymbol{x}) - \hat{\boldsymbol{h}}(\boldsymbol{x})\right\|^2 \tag{1}$$

where the stacked vectors $\boldsymbol{h}(\boldsymbol{x}) \triangleq \left[\boldsymbol{h}_0(\boldsymbol{x})^T \cdots \boldsymbol{h}_\ell(\boldsymbol{x})^T\right]^T$ and $\hat{\boldsymbol{h}}(\boldsymbol{x}) \triangleq \left[\hat{\boldsymbol{h}}_0(\boldsymbol{x})^T \cdots \hat{\boldsymbol{h}}_\ell(\boldsymbol{x})^T\right]^T$ have been used in the last equality. Hence, SAP considers an anomaly score that relies on the reconstruction errors from multiple AE layers—i.e. the output layer $i = 0$ and the internal (viz. encoder) layers $i = 1, \ldots, \ell$—and weights them *equally*. Conversely, the anomaly score based on NAP is based on a Mahalanobis-type distance:

$$a_{\text{nap}}(\boldsymbol{x}) = \left\|(\boldsymbol{d}(\boldsymbol{x}) - \boldsymbol{\mu}_x)^T \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\right\|^2 \tag{2}$$

where $\boldsymbol{d}(\boldsymbol{x}) \triangleq (\boldsymbol{h}(\boldsymbol{x}) - \hat{\boldsymbol{h}}(\boldsymbol{x}))$, $\boldsymbol{\mu}_x$ is the average of $\boldsymbol{d}(\boldsymbol{x})$ on the training set, and $\bar{\boldsymbol{D}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, with $\bar{\boldsymbol{D}}$ being the matrix obtained by stacking all the vectors $\boldsymbol{d}(\boldsymbol{x}_i)$ within the training set and subtracting the mean $\boldsymbol{\mu}_x$ from each.[4]

KitNET [4]. We evaluate an *Ensemble of Autoencoders* (Fig. 3) named KitNET [4] that is composed of two stages. Specifically, the *first* stage (the "ensemble" stage) is constituted of several autoencoders, each one reconstructing a portion of the input data. More specifically, partitioning the input in P non-overlapping portions as $\boldsymbol{x} = \left[\boldsymbol{x}_1^T \cdots \boldsymbol{x}_P^T\right]^T$, there is one AE for each portion, namely $\hat{\boldsymbol{x}}_p = (f^p \circ g^p)(\boldsymbol{x}_p)$ for the pth portion. Then, the *second* stage (the "output" stage) is made of a single AE that enforces a non-linear voting mechanism by reconstructing the reconstruction errors from the ensemble stage. Specifically, for each portion, the corresponding reconstruction error is calculated as $\mathcal{L}_p = \mathcal{L}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p)$. Then, the AE is trained to reconstruct $\boldsymbol{\ell} = [\mathcal{L}_1 \cdots \mathcal{L}_P]^T$ via the encoding-decoding structure, resulting in $\hat{\boldsymbol{\ell}} = f_{\text{rec}} \circ g_{\text{rec}}(\boldsymbol{\ell})$. Once the reconstruction is obtained, the anomaly score is calculated as $a_{\text{kitnet}}(\boldsymbol{x}) = \mathcal{L}(\hat{\boldsymbol{\ell}}, \boldsymbol{\ell})$, where $\mathcal{L}(\cdot)$ is the error loss (specifically, RMSE).

[4]To deal with the application of the Singular Value Decomposition (SVD) when computing the Mahalanobis distance, we resort to a randomized SVD with an (empirically-)fixed cutting threshold (equal to 10^{-10}) to avoid the inclusion of nearly-zero singular values.

There are different *design choices* for the aforementioned structure, such as the number of input portions and the grouping within a given portion. We also investigate the performance enhancements achievable by KitNET by optimizing these aspects.

KitNET Enhanced. Hereinafter, we describe three enhancements we propose for KitNET, each one tailored to solve a specific problem. In particular, we implement the following solutions.

- *Ensemble Equalization*—to properly schedule the features for the AE of the ensemble stage. This enhancement results in an unsupervised procedure that aims at balancing the training of the ensemble stage. In detail, the ensemble equalization procedure assigns each feature to a specific ensemble AE in a round-robin fashion, taking each feature based on its feature-importance rank. To determine the latter, because the AD scenario is somewhat unsupervised, we resort to the Arithmetic Mean-Geometric Mean (AMGM) feature-ranking procedure. This dispersion measure is used to rank the features: the higher the value, the more relevant the feature.

- *Changing the Output Stage Reconstruction Target*—to enhance the reconstruction error. This enhancement aims at substituting the reconstruction error used to train the output stage (i.e. the RMSE) with more advanced solutions, such as the `NAP`.

- *Ensemble Normalization*—to adjust the output stage modeling. This enhancement aims at transforming the ensemble stage output (i.e. the output stage input) into a probability distribution. Particularly, this procedure scales up the reconstruction errors introduced with the previous enhancement.

We underline that these enhancements have been experimentally evaluated in terms of both detection performance and model robustness, considering the impact of each of them taken individually and the effect of their combination.

3.2. ML-based Detectors

Isolation Forest (`IF`). The `IF` is designed with the idea that anomalies are "few and distinct" data points. It is an ensemble method (similar to the well-known supervised Random Forest) where each tree is built by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node. As a result, random partitioning produces noticeably *shorter paths* i_t's for anomalies. Hence, when a forest collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies. Hence the path length, averaged over a forest of such random trees, is a measure of normality and it is employed as $a_{\text{if}}(\boldsymbol{x}) = \rho(\sum_{t=1}^{T} i_t(\boldsymbol{x}))$, where $\rho(\cdot)$ is a decreasing function of this average.

Local Outlier Factor (LOF). The LOF belongs to the class of local outlier methods and uses the notion of "reachability distance", defined as:

$$d_{\text{reach}}(\boldsymbol{x}, \boldsymbol{x}_j) = \max\left\{d_{\text{knn}}(\boldsymbol{x}_j), d(\boldsymbol{x}, \boldsymbol{x}_j)\right\} \tag{3}$$

This distance is used to calculate the local reachability density of a point:

$$\text{lrd}(\boldsymbol{x}) = \frac{|\mathcal{N}_i|}{\sum_{j \in \mathcal{N}_i} d_{\text{reach}}(\boldsymbol{x}, \boldsymbol{x}_j)} \tag{4}$$

Finally, this density measure is compared with that of neighboring points to determine the local outlier factor:

$$a_{\text{lof}}(\boldsymbol{x}) = \frac{\sum_{j \in \mathcal{N}_i} \text{lrd}_k(\boldsymbol{x}_j)}{|\mathcal{N}_i| \, \text{lrd}_k(\boldsymbol{x})} \tag{5}$$

which is then used as the anomaly score.

One-Class Support Vector Machine (OC-SVM). The OC-SVM basically separates all the benign input points from the origin in the input space $\boldsymbol{x}$ and maximizes the distance from this hyperplane to the origin. This results in a binary function that captures regions in the input space where the probability density of the normal data lives. Thus, the mapping returns $+1$ in a "small" region (capturing the benign data points) and -1 elsewhere. Hence, such a method creates a hyperplane that has (*i*) maximal distance from the origin and (*ii*) separates all the data points from the origin. This leads to the following form:

$$a_{\text{ocsvm}}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, K(\boldsymbol{x}, \boldsymbol{x}_i) \tag{6}$$

where n denotes the number of training samples used as support.

3.3. Choice of the Detection Threshold

When training an anomaly detector to flag malicious traffic, the likelihood of the traffic object being considered is an increasing function of the anomaly score (e.g., the reconstruction error for AE-based AD techniques). Once the anomaly score is computed using the generic AD technique, it is interesting to understand how a threshold can be designed to ensure a given false-alarm rate cap. In this work, we consider a *data-driven* threshold expressed in the form:

$$\lambda = \mu_{a_{ben}} + k \cdot \sigma_{a_{ben}} \tag{7}$$

where $\mu_{a_{ben}}$ and $\sigma_{a_{ben}}$ represent the average and standard deviation of the generic anomaly score, respectively, when evaluated on (training) benign traffic [9], and k is a positive integer.

3.4. Threat Model – Label Flipping Attack

The main assumption of recent works proposing unsupervised AD-based solutions for IoT devices is the cleanliness of training traffic data [4, 9, 13], meaning that at the moment of the model construction, no malicious biflows are present.

In this section, we introduce the threat model defined by relaxing this assumption (Fig. 4), which requires not only that benign traffic generated by IoT devices must be collected immediately after the device installation, but also that no attackers should inject malicious traffic into the monitored network, adversarially introducing adversarial faults into the learned model. The consequent condition after the training of the detector is identifiable as a variant of a Data Poisoning Attack (DPA), named *Label Flipping Attack (LFA)* [30].

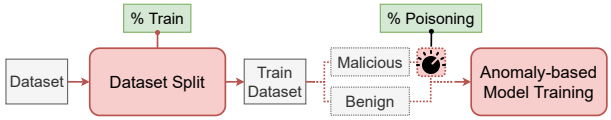


Figure 4: Threat model workflow of the Label Flipping Attack. The arrows follow the flowing of data (gray boxes) and the dots connect functions (red rounded boxes) to related attributes (green boxes). The dotted lines denote info which are available only in controlled scenarios.

From the adversarial ML perspective, the DPA involves the forging of input features that deviates the learning phase from the expected behavior when fed to the ML-based model. DPA is a *causative* attack, which means that the attacker can alter the training data used by the detector [31]. LFA belongs to this family of attacks but impacts the sole label space, namely the attacker *flips* the label of a specific class to the target class label, e.g., network-attack samples are labeled as benign. Thus, LFA does not resort to any modification of the feature space.

DPA can cause different kinds of damage depending on the target of the attacker. In fact, when benign traffic is misclassified, DPA could evolve into a DoS for legitimate users [32] (i.e. *availability disruption*). On the other hand, when malicious traffic is confused with benign traffic (i.e. *integrity disruption*), DPA could ensure evasion. The capabilities required from an attacker range from direct control of a portion of the generated traffic (e.g., one of the devices is already infected) to the influence in the traffic generation path (e.g., injecting crafted traffic from an external network).

From the defensive point of view, the attacker's actions can be mitigated by limiting the knowledge of (*i*) the learning algorithm, (*ii*) the feature space, and (*iii*) the training and evaluation data [31]. However, limiting the first two goes in the security-by-obscurity direction, which should be avoided. Moreover, acting on the collected data is infeasible because we assume a workflow that automatically extracts traffic features from the raw network traffic, with no means to distinguish the poisoned biflows from the legitimate ones. Once explored the other alternatives, what remains is acting on the preprocessed traffic samples or on the design of the detector model [31], thus resulting in a simple cleaning mechanism of the training dataset for the former, or in the design of robust detection solutions for the latter. However, from our preliminary work [6], in contrast to what was experienced in [33], the usage of advanced detector models (i.e. an ensemble of classifiers) has not resulted in higher robustness to DPA.

Based on the aforementioned issues, to enhance and extend the analysis we conducted in our previous work [6], in this paper we evaluate the robustness of different ML- and AE-based AD solutions, showing for the latter the impact of several optimization strategies on attack robustness. More specifically,

starting from the commonly assumed traffic cleanliness, we consider the scenario where an AD model is trained with only benign traffic data. Then, we emulate the presence of an attacker performing LFA by purposely injecting into the training dataset an increasing percentage of malicious traffic.

4. Experimental Setup

In the following, we describe the experimental setup, with the aim of fostering the reproducibility of the conducted analysis. In detail, in Sec. 4.1, we present the two datasets we leverage herein and the related preprocessing operations. Then, in Sec. 4.2, we introduce the features used to feed the anomaly detectors. Finally, in Sec. 4.3, we discuss the evaluation metrics and the tools adopted.

4.1. Datasets and Pre-processing Operations

In this work, we employ *two publicly-available datasets* collecting benign and malicious network traffic captured in IoT environments. Hereinafter, we provide the description of common *pre-processing operations* performed on them before giving individual details on each.

Specifically, pre-processing steps parse raw network traffic to obtain the relevant *traffic object* and associated information (viz. input) to feed the ML and DL algorithms that perform the AD task. To conduct our analysis we segment (PCAP) network traces into *bidirectional flows* (viz. *biflows*), being the most common choice of the vast majority of state-of-art works tacking AD (see Tab. 1). Each biflow is a set of packets sharing the same 5-tuple {`Src IP`, `Src Port`, `Dst IP`, `Dst Port`, `Proto`} where the source and destination IP addresses and ports of the 5-tuple can be swapped [13], in order to capture the bidirectional communication pattern between sender and receiver.

`IoT-23`. The first dataset employed is `IoT-23` [5], collected at the Stratosphere Laboratory of the Czech Technical University during 2018-19. `IoT-23` is made of 23 PCAP traffic traces captured in a controlled IoT environment with an unrestrained network connection. Each trace corresponds to a specific malware sample or to benign traffic, with a total of 20 malicious and 3 benign traces. A Raspberry Pi infected with a certain malware is exploited to generate malicious traffic, while three real IoT devices (i.e. a Philips HUE Smart Led Lamp, an Amazon Echo Home, and a Somfy Smart Doorlock) generate benign one. The dataset is manually labeled (at biflow level) by describing the relation between malicious flows and malicious activities performed, while non-malicious traffic is simply labeled as "benign".[5] `IoT-23` exhibits a severe class imbalance problem: the four most highly-populated classes (i.e. *PartOfAHorizontalPortScan*, *Okiru*, *DDoS*, and *Benign*) have more than 15M biflows and the three least-populated classes (i.e. *C&C-Mirai*, *Okiru-Attack*, and *PartOfHorizontalPortScan-Attack*) present less than 10 biflows, whereas all the other classes have no more than 40k biflows. To address this issue, we down-sampled (randomly, without replacement) the former majority classes to the 0.25% of the original dataset[6], and we have removed the latter minority classes. We underline that—although preprocessing operations guarantee a sufficient number of biflows for each class and reduce the computational burden—given the unbalanced per-class share of samples, such a dataset represents a realistic and challenging evaluation testbed. After such preprocessing operations, the `IoT-23` dataset comprises $\approx$ 870k biflows distributed among 13 (one benign + 12 attacks) classes, as depicted in Fig. 5a.

`Kitsune`. The second dataset used in our experimental evaluation is the `Kitsune` network-attack dataset [4]. The dataset is collected in an IP-based commercial surveillance system by setting up an IoT network testbed consisting of two deployments of four HD surveillance cameras each. The authors release raw data (in PCAP format) and *per-packet labeling* information that we exploited to assign a (benign/attack) label to each biflow. The attacks are conducted by means of different tools (e.g., Nmap, Hping3, Ettercap) and are targeted to affect the availability and integrity of the video uplinks. In more detail, the authors focused on 9 attack classes grouped into 4 categories. Given the unbalanced yet manageable per-class share of biflows, in this case, we have not adopted any pre-processing operations. Overall, `Kitsune` counts $\approx$ 150k biflows distributed among 10 (one benign + 9 attacks) classes. Figure 5b gives the details on the distribution of biflows across the classes.

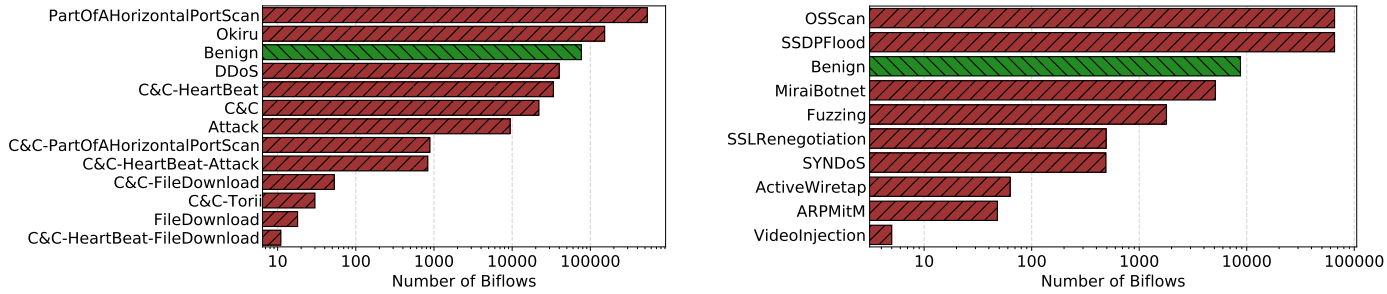
4.2. Feature Definition & Extraction

Starting from raw traffic data, *feature extraction* is performed by computing (experts-defined) characteristics related to traffic objects. The identified features can encompass different aspects of interest, such as the set of statistical features that can be read (*i*) by observing the sequence of packets composing a traffic object and (*ii*) by inspecting their content (i.e. the value of some packet fields or even their payload). Since traffic encryption dramatically hinders the information that can be extracted from a ciphered payload, we focus on the former kind of features together with some (IP and TCP/UDP) header fields which are not subject to encryption. This choice is also motivated by the increasing need for IoT devices to protect communications with encryption [34] (for both security and privacy reasons). In addition, it is worth mentioning that the features can be either extracted considering the entire traffic object (AD), or from its first chunks (early AD). For the sake of prompt and close-to-practical AD, *we focus on the latter case*.

In detail, we extracted 79 unique features from both datasets focusing on the characteristics of the first N packets, with $N \in [1, \ldots, 20]$. These features are: (*i*) *Number of Packets*, (*ii*) *Packet Size* (PS), (*iii*) *Inter-Arrival-Time* (IAT), (*iv*) *Time To Live*, (*v*) *TCP Window*, (*vi*) *TCP Flags* (FLG), and (*vii*)

[5] We refer to `IoT-23` website [5] for further details on the labels used for malicious traffic. Unfortunately, the meaning of the various *C&C* attacks (except for the *C&C* generic attack) is not specified. We assume that each part of the label indicates a different phase of the same attack session.

[6] We have chosen to down-sample the *whole* dataset, as opposed to the sole training set, since the latter choice would have biased the overall accuracy (evaluated on the test set) toward the performance of the majority classes.



(a) `IoT-23`.

(b) `Kitsune`.

Figure 5: Number of per-class biflows (in log scale) of preprocessed `IoT-23` and `Kitsune` datasets. For details on label meaning, please refer to [5] and [4], respectively.

Byte Rate. Regarding time-series characteristic (*ii-v*), we compute 17 statistics for each, namely *minimum*, *maximum*, *average*, *standard deviation*, *mean absolute deviation*, *kurtosis*, *skew*, *variance*, and *percentiles* from 10th to 90th with step 10. Uniquely for the PS, we report its *summation* (viz. the total number of transmitted bytes). Moreover, to compute statistics for the IAT we discard zero values (viz. the IAT of the first packet of each biflow). Finally, we encode FLG by using 8 counters, each one reporting the number of occurrences of the corresponding flag.

4.3. Evaluation Metrics and Adopted Tools

In this section, we provide details about used (*i*) evaluation metrics and (*ii*) tools/procedures. First, the experimental evaluation is performed by applying a 10-fold stratified cross-validation procedure, in order to better assess detection capabilities. Hence, the reported metrics are averaged over the considered folds and, when needed to assess statistical significance, also confidence intervals are included.

AD performance is evaluated through the classic *True Positive Rate* (TPR) and *False Positive Rate* (FPR) which represent the rate of malicious biflows correctly labeled as malicious and the rate of benign biflows wrongly labeled as malicious, respectively. We underline that usually the FPR can be set by adjusting the anomaly threshold λ to which the anomaly score $a(\boldsymbol{x})$ is compared. The behavior of AD when varying the threshold and reporting the TPR vs. FPR is summarized by means of the (Receiver Operating Characteristics) ROC curve. Furthermore, to report concise results about a given AD technique, we also report the well-known F1 Score (i.e. the harmonic mean of precision and recall for the AD task) evaluated at 1% of FPR.

Additionally, we consider the *area under the ROC curve* (AUC), which is often used to summarize in a single number the detection ability of the generic AD approach. The AUC is simply defined as the area of the ROC space that lies below the ROC curve, i.e.

$$\text{AUC} = \int_0^1 \text{TPR}(\text{fpr})\, d\text{fpr} \tag{8}$$

Unfortunately, the (plain) AUC integrates also taking into account high FPR values which may be of little practical value for AD. Hence, the idea of the *partial AUC* (pAUC) is to restrict the evaluation of given ROC curves in the range of FPR values that are considered interesting for AD purposes:

$$\text{pAUC} = \int_{p_\ell}^{p_u} \text{TPR}(\text{fpr})\, d\text{fpr} \tag{9}$$

where p_u (resp. p_ℓ) represents the highest (resp. the lowest) FPR value evaluated in the integral. Accordingly, in the following, we leverage the pAUC when $p_\ell = 0$ and $p_u = 0.01$, namely considering the AUC limited at the 1% FPR.

The prototypes we used are written in `Python`, leveraging the `tensorflow` library. Details about the configuration of each model are reported in Tab. 2. The AE-based detectors we consider in this work are six: the first three are autoencoders at increasing depths, namely `AE-1` is a shallow autoencoder, while `AE-2` and `AE-3` are two deep autoencoders with two and three encoding/decoding layers, respectively; the last three are ensembles of such variants of autoencoders, resulting in `KitNET-1`, `KitNET-2`, and `KitNET-3`, which are ensembles of `AE-1`, `AE-2`, and `AE-3`, respectively. It is worth noting that we maintained unaltered the size of the coding at 0.50× the input size (viz. number of features).

All the models are trained using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.01, a Momentum of 0.9, the Mean Squared Error (MSE) as a loss function, and a batch size set to 32. The training lasts up to 200 epochs, with an early stopping mechanism that monitors the validation set (10% of the training set) with patience of 10 epochs and minimum delta of 10^{-4}. Min-Max scaling is applied to project each feature in the range $[0, 1]$ before feeding the AD models. It is worth underlining that the scaling is applied by fitting the sole training set. Finally, when the ensemble normalization optimization is enforced, the intermediate distance metrics—which are computed from the KitNET ensemble stage to feed its output stage—are converted into a probability vector.

5. Experimental Evaluation

In the experimental analysis that follows, we describe the data-driven design aimed at identifying the most suitable configurations for the IoT anomaly detectors under investigation.

Table 2: Details about the AE-based detectors employed.

Name	Configuration
AE-1	$I(n)$; $D(0.50{\times}n, R)$; $D(n, S)$.
AE-2	$I(n)$; $D(0.75{\times}n, R)$; $D(0.50{\times}n, R)$; $D(0.75{\times}n, R)$; $D(n, S)$.
AE-3	$I(n)$; $D(0.83{\times}n, R)$; $D(0.66{\times}n, R)$; $D(0.50{\times}n, R)$; $D(0.66{\times}n, R)$; $D(0.83{\times}n, R)$; $D(n, S)$.
KitNET-1	5 *ensemble* autoencoders: $I(0.20{\times}n)$; $D(0.50{\times}0.20{\times}n, R)$; $D(0.20{\times}n, S)$. *output* autoencoder: $I(5)$; $D(3, R)$; $D(5, S)$.
KitNET-2	5 *ensemble* autoencoders: $I(0.20{\times}n)$; $D(0.75{\times}0.20{\times}n, R)$; $D(0.50{\times}0.20{\times}n, R)$; $D(0.75{\times}0.20{\times}n, R)$; $D(0.20{\times}n, S)$. *output* autoencoder: $I(5)$; $D(4, R)$; $D(3, R)$; $D(4, R)$; $D(5, S)$.
KitNET-3	5 *ensemble* autoencoders: $I(0.20{\times}n)$; $D(0.83{\times}0.20{\times}n, R)$; $D(0.66{\times}0.20{\times}n, R)$; $D(0.50{\times}0.20{\times}n, R)$; $D(0.66{\times}0.20{\times}n, R)$; $D(0.83{\times}0.20{\times}n, R)$; $D(0.20{\times}n, S)$. *output* autoencoder: $I(5)$; $D(5, R)$; $D(4, R)$; $D(3, R)$; $D(4, R)$; $D(5, S)$; $D(5, S)$.

Legend: Input (I), #feats (n), Dense (D), ReLU (R), and Sigmoid (S).

Specifically, we inspect the **impact of several factors**, such as the *number of packets* per biflow taken into account in the training and in the inference phase, the *distance metric* used to compute the distance between representations (Sec. 5.1), and the *depth of the DL architectures* (Sec. 5.2). Also, we deepen the **nature of the different distance metrics considered**, both investigating the *distribution of the scores* provided and their impact on the detection thresholds that can be defined in a practical scenario (Sec. 5.3) and putting them in relation with the *contribution imputable to either the external representations or the latent space*, which is leveraged by more advanced solutions (Sec. 5.4). Then, we investigate the **enhancements** that can be enforced when exploiting KitNET-based detectors (Sec. 5.5). Finally, we consider an adversarial scenario, where the training phase of the model is partially compromised (i.e. where the supposedly benign amount of traffic used to train the architectures contains a fraction of malicious samples), thus providing an evaluation of the **robustness** of the designed solution against *data poisoning attacks* (Sec. 5.6).

5.1. Sensitivity to the Number of Packets

First, we aim at identifying the optimal number of packets to be taken into account for each biflow. These packets are then used during both the training and inference phases. Indeed, on the one hand, considering a higher number of packets can improve the detection performance of the models (i.e. larger knowledge to capitalize on). On the other hand, it negatively impacts the complexity of the model (i.e. larger input to manage) and introduces delays during the inference phase reducing the "earliness" of detection: the detector has to wait for more packets to observe before providing its verdict—with inter-arrival times possibly depending on the specific application generating the traffic.

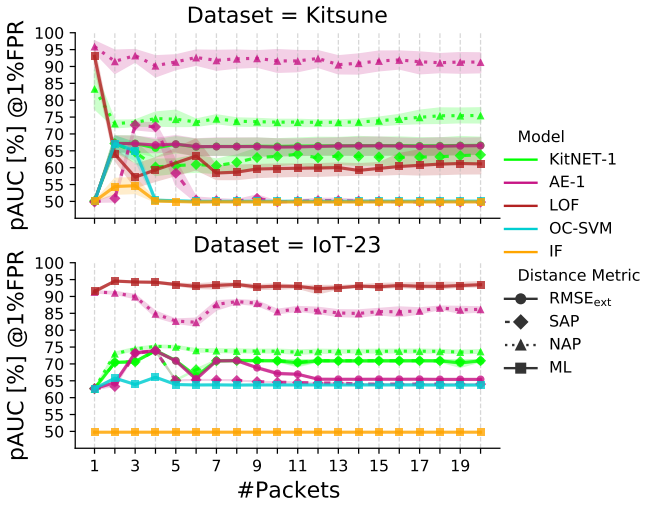


Figure 6: Sensitivity to the number of packets. The results are shown as *avg.* ± 95% *CI* over a 10-fold cross-validation procedure.

Hence, in this first analysis we *empirically assess the impact of the number of packets on the detection performance*. Specifically, in Fig. 6 we experimentally evaluate how varying the number of observed packets from 1 to 20 impacts the effectiveness of detection (measured through pAUC @1%FPR, which we recall corresponds to $p_\ell = 0$ and $p_u = 0.01$ in Eq. (9)) of each shallow model (i.e. AE-1 and KitNET-1, see Tab. 2) and considering the three distance criteria (i.e. RMSE, SAP, and NAP). In addition, three state-of-the-art ML detectors, namely LOF, OC-SVM, and IF, are considered as baselines. The analysis is performed on both Kitsune and IoT-23 datasets, reporting the results averaged over a 10-fold cross-validation procedure.

Looking at the results, a clear trend emerges: for both models and regardless of the distance metric the pAUC @1%FPR (which is a stable indicator) reaches a plateau starting from the 12th packet. This applies to both Kitsune and IoT-23. Deepening, the maximum pAUC @1%FPR is attained by looking at the first 4 packets: at this value, some of the curves even show performance peaks. In general, the models based on NAP outperform those based on RMSE and SAP. Noteworthy, despite ML-based detectors fail in the majority of cases, LOF reaches outstanding performance on IoT-23, with pAUC @1%FPR stable at ≈ 95% regardless of the number of packets considered.

According to these results, *we select* NAP *as the most promising option to be investigated in the following analyses*. Furthermore, *we select* 4 *as the number of packets to take into account for both datasets*. Also, since the datasets contain both TCP and UDP traffic, this choice allows us to avoid limiting the observations to the sole three-way handshake for TCP biflows.

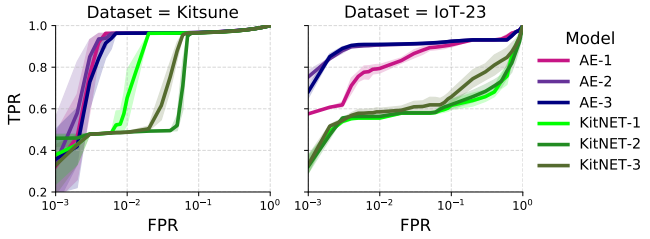


Figure 7: Sensitivity to the model depth via ROC curve. Models are fed with 4 packets and anomalies are detected via NAP score. The results are shown as *avg.* ± 95% *CI* over a 10-fold cross-validation procedure.

5.2. Sensitivity to the Depth of the Model

AE-based architectures can be naturally designed with varying complexity, i.e. introducing additional layers to both the encoding and decoding (sub)networks. The following analysis aims at providing an empirical assessment of the impact of this design choice on the detection performance.

In more detail, we evaluate the benefit of increasing the *depth of the considered architectures*: to investigate the effects of supplementary encoding/decoding layers, we consider three variants (with different depths) for each architecture, which are named based on the number of layers constituting both the encoding and the decoding network [17, 16], i.e. AE-1, KitNET-1 (one internal encoding/decoding layer), AE-2, KitNET-2 (two internal encoding/decoding layers), AE-3, and KitNET-3 (three internal encoding/decoding layers).

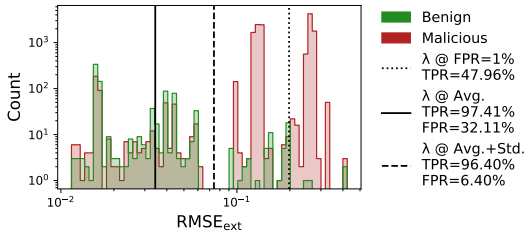
Figure 7 shows the sensitivity of the model to the depth via the ROC curve for the resulting architectural configurations, leveraging the NAP score and considering 4 input packets.[7] When focusing on the AE-based models, no major performance discrepancy can be spotted from the results obtained on the Kitsune dataset. On the other hand, the analysis involving IoT-23 suggests that AE-2 and AE-3 perform better than AE-1. Concerning KitNET-based models, the ROC curves suggest that KitNET-1, KitNET-2, and KitNET-3 report comparable performance on IoT-23, whereas on Kitsune they provide different performance pictures. In fact, while KitNET-1 results in a larger AUC, overall, the major benefits come for FPR values larger than 10^{-2}, which are of little practical interest for network AD. Moreover, focusing on smaller values (left side of the figure), KitNET-2 even outperforms the other KitNET-based models, boosting higher TPR even for very low FPR values, i.e. 45.83% TPR @0.1%FPR scored by KitNET-2 against 37.85% obtained by KitNET-3.

From the results attained on both datasets, we select the AE-2 *and the* KitNET-2 *as the best detectors when using* NAP.

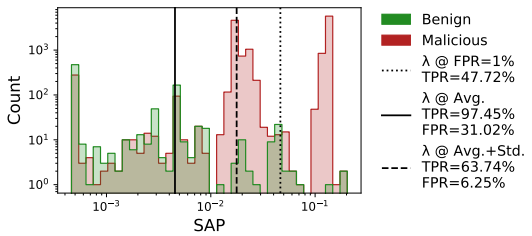
5.3. Deepening the Impact of the Distance Score

In Sec. 5.1 the (black-box) analysis led to the selection of NAP as the most suitable distance score. In this section, we further focus on the motivations that drove us to this result. With this aim in mind, we dissect the impact of distance metrics by analyzing the distributions of the score values attainable with the different distance metrics.

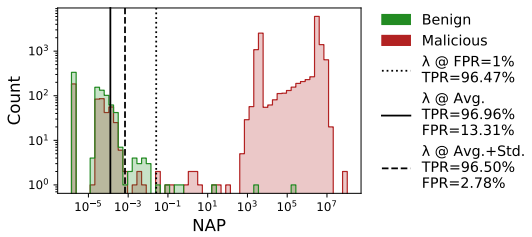
[7] We underline that we have obtained analogous results with a higher number of packets (i.e. 8), not shown for brevity since the related outcomes are in line with those reported for 4 packets.



(a) RMSE$_{EXT}$ score on Kitsune.



(b) SAP score on Kitsune.



(c) NAP score on Kitsune.

Figure 8: Comparison of predicted scores on Kitsune considering the AE-2. Each vertical line represents a different detection threshold. Values are computed on a single fold for visualization purposes.

Taking into account the case of AE-2 as an example of specific interest (also according to the analysis in Sec. 5.2), Fig. 8 reports the histograms of the counts of the anomaliness score $a(\boldsymbol{x})$ obtained with the three distance metrics in an execution on Kitsune. Results on IoT-23 and KitNET-2 show analogous behaviors and are not reported for brevity.

When comparing the distributions achieved with the different metrics, it is evident that the score obtained via NAP (Fig. 8c) is more effective in separating benign and malicious samples, namely, the distribution of red and green bars are less overlapped than those attained with RMSE and SAP (Figs. 8a and 8b, respectively). This results in a remarkably higher TPR (when FPR = 1%), which moves from ≤ 48% (for RMSE and SAP) up to 97% (for NAP).

Also, for each model, we have evaluated the performance attainable when setting the threshold based on a data-driven analysis, namely by relying on the scores observed during the training phase. Specifically, we have considered two strategies for defining the threshold λ, according to Eq. (7). Specifically, λ

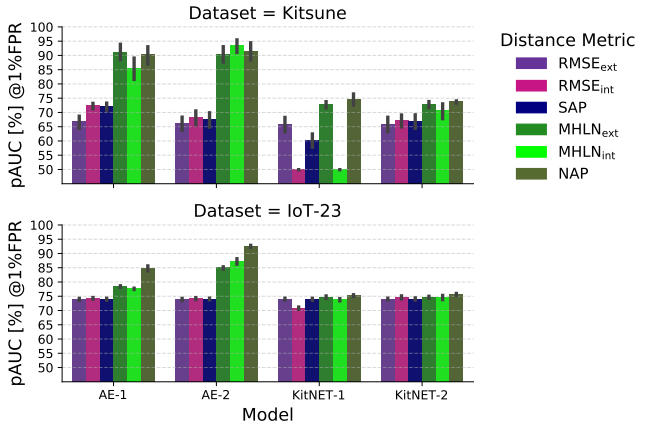


Figure 9: Ablation study for RMSE-based and Mahalanobis-based distances using AE-2 and KitNET-2 models. The results are shown as *avg.* ± 95% *CI* over a 10-fold cross-validation procedure.

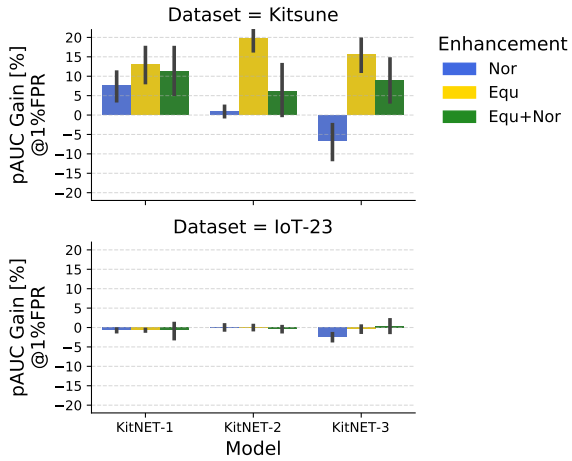


Figure 10: KitNET enhancements evaluation looking at the NAP score. The gain with respect to the KitNET without optimizations is shown. The results are shown as *avg.* ± 95% *CI* over a 10-fold cross-validation procedure.

is set to: (*i*) the average of the scores associated with benign samples @Avg. (corresponding to $k = 0$ in Eq. (7)) and to (*ii*) @Avg.+Std. (corresponding to $k = 1$ in Eq. (7)) as done in recent works [4, 9]. Noteworthy, the NAP score clearly reduces the gap between the theoretical threshold and the two data-driven ones, easing the development of robust anomaly detectors. Indeed, we can more reliably set the detection threshold empirically to the @Avg. or to the @Avg.+Std. scores associated with the benign (training) samples.

In summary, the NAP *distance score performs always better than RMSE and* SAP. Following this direction, we further explore the nature of this improvement with the *ablation study* performed hereinafter.

5.4. Ablation Study for Distance Metrics

Both SAP (cf. Eq. (1)) and NAP (cf. Eq. (2)) aim at improving the detection capability of reconstruction-based models taking into account the errors that can be observed *also* in the latent space. This analysis points to dissect the contribution of internal and external layers in the computation of error metrics based on RMSE and Mahalanobis distances, in order to better explain the results achieved so far.

Accordingly, we decompose the score evaluated by SAP investigating RMSE_{INT} and RMSE_{EXT} (corresponding to $i = 1, \ldots, \ell$ and $i = 0$ in Eq. (1), respectively) separately—with the latter matching the more "classic" RMSE-based score already evaluated in Sec. 5.1. Similarly, NAP scores are analyzed by observing "internal" and "external" Mahalanobis-based scores (MHLN_{INT} and MHLN_{EXT}, respectively) separately. In other terms, in the external (resp. internal) case, the formula in Eq. (2) is applied by using $\boldsymbol{h}_{\text{ext}}(\boldsymbol{x}) \triangleq \left[\boldsymbol{h}_1(\boldsymbol{x})^T \cdots \boldsymbol{h}_\ell(\boldsymbol{x})^T\right]^T$ (resp. $\boldsymbol{h}_0(\boldsymbol{x})$) and $\hat{\boldsymbol{h}}_{\text{ext}}(\boldsymbol{x}) \triangleq \left[\hat{\boldsymbol{h}}_1(\boldsymbol{x})^T \cdots \hat{\boldsymbol{h}}_\ell(\boldsymbol{x})^T\right]^T$ (resp. $\hat{\boldsymbol{h}}_0(\boldsymbol{x})$).

Results are shown in Fig. 9, for both the datasets (top Kitsune and bottom IoT-23) and considering 4 packets. For this setup, we investigate both 1-layer and 2-layer reconstruction-based models, namely AE-1, AE-2, KitNET-1, and KitNET-2. Three main trends emerge: (*i*) the use of Mahalanobis-based score is beneficial on both datasets and for all models considered, *independently on the layer chosen* (i.e. for hidden, output, or both layers, corresponding to $\text{RMSE}_{\text{INT}} \rightarrow \text{MHLN}_{\text{INT}}$, $\text{RMSE}_{\text{EXT}} \rightarrow \text{MHLN}_{\text{EXT}}$, and SAP →NAP, respectively); (*ii*) *the effects* of the combination of internal and external contribution through Mahalanobis-based score *using* NAP (as opposed to MHLN_{INT} or MHLN_{EXT} individually) *strongly depend on the dataset*: such effects are always positive (or at least non-detrimental) on IoT-23 while on Kitsune this joint use rarely outperforms the pAUC achieved by internal or external contribution alone; (*iii*) the combination of internal and external contributions through an equally weighted score, that is, *using* SAP (as opposed to either RMSE_{INT} or RMSE_{EXT} individually) *does not seem to provide an appreciable benefit*.

5.5. KitNET Enhancements

This section experimentally assesses the benefits of KitNET enhancements introduced in Sec. 3.1, namely (*i*) the *ensemble equalization*, (*ii*) *changing the output stage reconstruction target*, and (*iii*) the *ensemble normalization*.

The impact of the ensemble-related enhancements (*i* and *iii*) is reported in Fig. 10, focusing on NAP distance score. Looking at the figure, the following observations can be drawn: on the Kitsune dataset (top row) the ensemble equalization gives the highest improvements; on the IoT-23 dataset (bottom row) neither an advantage nor a loss is obtained from the adoption of (part of) these enhancements. *For this reason, we recommend the adoption of equalization due to its benefits in more challenging contexts.*

On the other hand, the advantages introduced by (*ii*) are shown in Fig. 11. The intuition behind this enhancement derives from the higher detection capabilities shown by Mahalanobis-based distances as anomaliness scores: because the AE in the output stage of the KitNET acts as a non-linear voting mechanism, having a more stable reconstruction of the training set on the ensemble stage autoencoders exploiting the

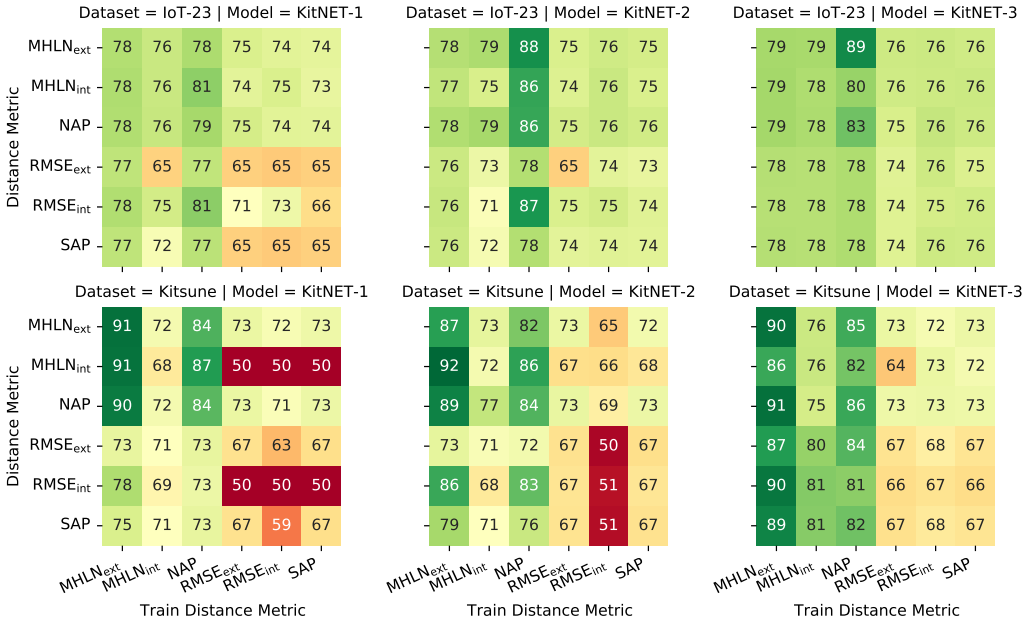


Figure 11: Evaluation of KitNET enhancements looking at the training distance impact on pAUC @1%FPR using 4 packets. The results are shown as *avg*. over a 10-fold cross-validation procedure.

Mahalanobis-based distances, could enhance the general detection capabilities of the KitNET. To this end, the pAUC @1%FPR is shown by considering different combinations of distance metrics (i.e. $\mathtt{MHLN}_{\mathtt{EXT}}$, $\mathtt{MHLN}_{\mathtt{INT}}$, NAP, $\mathtt{RMSE}_{\mathtt{EXT}}$, $\mathtt{RMSE}_{\mathtt{INT}}$, and SAP) used for training the output stage autoencoder of the KitNET (i.e. Train Distance) and for detection purposes (i.e. Distance).

In Fig. 11 *some macro effects* can be observed. First, the impact of a greater depth on the base AEs composing the KitNET, namely moving from KitNET-1 to KitNET-2 and KitNET-3, highlights a weaker sensitivity to the choice of the distance metric used for detection. This trend applies to both datasets, i.e. IoT-23 and Kitsune. Specifically, using RMSE-based train distances (the last three columns of each matrix) negatively impacts the detection performance in all devised scenarios. On the other hand, using Mahalanobis-based train distances, like $\mathtt{MHLN}_{\mathtt{EXT}}$ and NAP, show higher detection capabilities when combined with Mahalanobis-based detection distances (the top-left 3×3 square of each matrix).

Table 3: pAUC @1%FPR by selecting the best configuration per poisoning ratio. Values are shown as *avg*. over 10 folds.

Dataset	Poisoning Percentage	Model	Distance Metric	Enha.	Train Distance	pAUC [%] @1%FPR
Kitsune	0.0	LOF	-	-	-	59.34
		AE-2	$\mathtt{MHLN}_{\mathtt{INT}}$	-	-	**93.55**
		KitNET-2	NAP	Equ	$\mathtt{RMSE}_{\mathtt{EXT}}$	92.91
	1.0	IF	-	-	-	57.84
		AE-2	$\mathtt{RMSE}_{\mathtt{EXT}}$	-	-	66.41
		KitNET-2	$\mathtt{MHLN}_{\mathtt{EXT}}$	Nor	$\mathtt{RMSE}_{\mathtt{EXT}}$	**74.19**
	5.0	LOF	-	-	-	50.39
		AE-2	$\mathtt{RMSE}_{\mathtt{EXT}}$	-	-	52.37
		KitNET-2	SAP	Nor	$\mathtt{RMSE}_{\mathtt{EXT}}$	**74.12**
IoT-23	0.0	LOF	-	-	-	**94.37**
		AE-2	NAP	-	-	92.51
		KitNET-2	$\mathtt{MHLN}_{\mathtt{EXT}}$	None	NAP	88.01
	1.0	OC-SVM	-	-	-	66.12
		AE-2	NAP	-	-	**83.56**
		KitNET-2	$\mathtt{MHLN}_{\mathtt{EXT}}$	None	NAP	76.86
	5.0	IF	-	-	-	67.92
		AE-2	NAP	-	-	**72.62**
		KitNET-2	$\mathtt{MHLN}_{\mathtt{EXT}}$	Equ	$\mathtt{RMSE}_{\mathtt{EXT}}$	70.03

5.6. Robustness to Label Flipping Attacks

This section aims at analyzing the robustness capabilities of the detector models which have been evaluated in the previous sections. Consequently, we relax the assumption of traffic cleanliness for the training phase and perform LFAs that involve portions of the training data with increasing size (see Sec. 3.4). In other words, we inject an increasing percentage of malicious traffic into the training set—which should be only composed of network traffic labeled as benign in the ideal case—analyzing the impact of a poisoning percentage ranging within 0.5%-5% of the benign traffic. To better assess the robustness of the detectors and to adhere to the distribution of samples across the malicious classes, we keep the proportions of the classes of malicious traffic to be injected in the training set balanced (viz. stratified poisoning selection). We underline that the random selection of poisoning samples is re-generated at each fold.

Table 3 summarizes the remarkable results of this analysis, reporting for each dataset and at varying values of poisoning ratio (0%, 1%, and 5%) the best-performing approach for each family (ML-based, AE-based, KitNET-based) with the resulting pAUC @1%FPR. The related configuration (model, depth, distance metric, and optimizations) is also mentioned. Conversely, an in-depth analysis of (*i*) a larger set of detectors, (*ii*) the entire poisoning range, and (*iii*) varying FPR values (via

the ROC) is reported in Appendix A. Looking at the Kitsune dataset, when no poisoning is enforced, AE-2 is the best approach (93.5% pAUC @1%FPR). However, for larger portions of the dataset being compromised (1%, 5%), KitNET-2 proves to be more robust, showing a more limited performance decay (74.2%–74.1% pAUC @1%FPR and boasting up to +22% pAUC @1%FPR w.r.t. AE-2 in the same poisoning condition). Focusing on the IoT-23 dataset, the scenario is quite similar, with the best performing LOF (94.4% without poisoning) outperformed by AE-2 when the poisoning percentage moves to 1.0% and 5.0%. Hence, reconstruction-based methods (AE-2 and KitNET-2) prove to be more robust to poisoning attacks.

6. Conclusions

In this paper, we have investigated advanced strategies for network AD in IoT environments. We have considered two (families of) state-of-the-art DL-based solutions, namely "classic" autoencoders and KitNET, investigating a number of variants and enhancements. Relying on an experimental campaign based on two recent publicly-available IoT datasets (i.e. IoT-23 and Kitsune), we have provided a data-driven design of the considered architectures, evaluating the impact of the available choices, and aiming at identifying the most suitable configurations for the detectors considered.

Specifically, we have assessed the impact of the number of packets taken into account for each biflow when training and running the detectors. The experimental evidence derived from both datasets shows that considering the input provided by the *first* 4 *packets* is the most suitable option. The above result demonstrates the *feasibility* of early AD of network attacks targeting the IoT ecosystem.

Concerning the distance metrics we have considered, the NAP score (leveraging both the errors evaluated at the external and internal layers via the Mahalanobis distance) proved the most suitable alternative, outperforming the other options evaluated in all the scenarios considered. Also, we have investigated the sensitivity of the so-far optimal architecture to model depth. Results suggested that detectors can benefit from deeper architectures, with the solutions with two layers (AE-2 and KitNET-2) providing the best performance when using NAP.

Further investigating the nature of the scores provided by the distance metrics, we found that NAP is able to guarantee a better separation between benign and malicious samples. Also, NAP reduces the gap between the theoretical and the data-driven threshold practically enforceable, thus easing the *fine-grained control of false alarms in realistic scenarios*. Moreover, results witness that the adoption of Mahalanobis-based scores is always beneficial w.r.t. analogous (i.e. internal, external, and joint) RMSE-based metrics and that NAP must be preferred to SAP when merging internal and external scores. Referring to KitNET enhancements, it has been shown that it can be greatly beneficial to apply ensemble equalization in some challenging contexts. Finally, looking at robustness in non-ideal conditions, the *higher appeal of reconstruction-based DL methods* (i.e. AE and Kitnet families) *has been underlined when subject to the reported poisoning attack (LFA)* in comparison to standard ML approaches (i.e. IF, LOF, and OC-SVM).

Future works will include: (*i*) evaluate the effectiveness of countermeasures against DPA, (*ii*) include a larger set of state-of-the-art techniques to be tested for (network) early AD in IoT context, (*iii*) the evaluation of such models in an online deployment (i.e. models are updated incrementally when new traffic data arrive/are collected), (*iv*) use of eXplainable Artificial Intelligence (XAI) tools to interpret (and possibly improve) the working principle of DL-based AD, and (*v*) design/deployment of lightweight techniques.

Acknowledgments

This work is partially supported by the Italian Research Program "PON *Ricerca e Innovazione* 2014-2020 (PON R&I) – Asse IV: *Istruzione e ricerca per il recupero – REACT-EU* – Azione IV.4: *Dottorati e contratti di ricerca su tematiche dell'innovazione*", the "Centro Nazionale HPC, Big Data e Quantum Computing – Italian Center for Super Computing (ICSC)" and the "RESTART" Project funded by MUR.

References

[1] C. Yin, S. Zhang, J. Wang, N. N. Xiong, Anomaly detection based on convolutional recurrent autoencoder for iot time series, IEEE Transactions on Systems, Man, and Cybernetics: Systems 52 (2020) 112–122.

[2] K. Ferencz, J. Domokos, L. Kovacs, Review of Industry 4.0 security challenges, in: IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI), 2021, pp. 245–248.

[3] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges, IEEE Transactions on Network and Service Management 16 (2019) 445–458.

[4] Y. Mirsky, T. Doitshman, Y. Elovici, A. Shabtai, Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection, Network and Distributed Systems Security Symposium (NDSS) (2018).

[5] S. Garcia, A. Parmisano, M. J. Erquiaga, IoT-23: A labeled dataset with malicious and benign IoT network traffic, 2020. URL: https://www.stratosphereips.org/datasets-iot23. doi:http://doi.org/10.5281/zenodo.4743746.

[6] G. Bovenzi, A. Foggia, S. Santella, A. Testa, V. Persico, A. Pescapé, Data poisoning attacks against autoencoder-based anomaly detection models: a robustness analysis, in: IEEE International Conference on Communications (ICC), 2022, pp. 5427–5432.

[7] H. Mac, D. Truong, L. Nguyen, H. Nguyen, H. A. Tran, D. Tran, Detecting attacks on web applications using autoencoder, in: 9th ACM International Symposium on Information and Communication Technology (SoICT), 2018, pp. 416–421.

[8] P. Madani, N. Vlajic, Robustness of deep autoencoder in intrusion detection under adversarial contamination, in: 5th ACM Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HoTSoS), 2018, pp. 1–8.

[9] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-baiot: network-based detection of IoT botnet attacks using deep autoencoders, IEEE Pervasive Computing 17 (2018) 12–22.

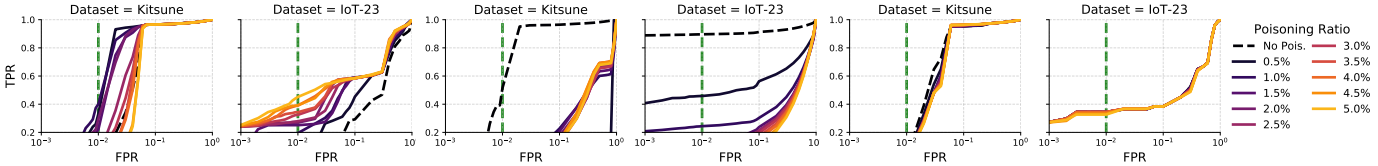
[10] B. J. Radford, L. M. Apolonio, A. J. Trias, J. A. Simpson, Network traffic anomaly detection using recurrent neural networks, arXiv preprint arXiv:1803.10769 (2018).

[11] G. Andresini, A. Appice, N. Di Mauro, C. Loglisci, D. Malerba, Exploiting the auto-encoder residual error for intrusion detection, in: IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2019, pp. 281–290.

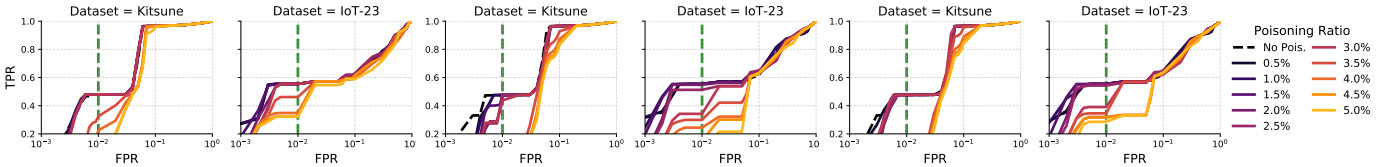
[12] F. A. Khan, A. Gumaei, A. Derhab, A. Hussain, A novel two-stage deep learning model for efficient network intrusion detection, IEEE Access 7 (2019) 30373–30385.
[13] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, A. Pescapé, A hierarchical hybrid intrusion detection approach in iot scenarios, in: IEEE Global Communications Conference (GLOBECOM), 2020, pp. 1–7.
[14] K. Yang, J. Zhang, Y. Xu, J. Chao, DDos attacks detection with autoencoder, in: IEEE/IFIP Network Operations and Management Symposium (NOMS), 2020, pp. 1–9.
[15] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, et al., Learning latent representation for IoT anomaly detection, IEEE Transactions on Cybernetics (2020).
[16] H. Kye, M. Kim, M. Kwon, Hierarchical detection of network anomalies: A self-supervised learning approach, IEEE Signal Processing Letters (2022).
[17] D. Yang, M. Hwang, Unsupervised and ensemble-based anomaly detection method for network security, in: 2022 14th International Conference on Knowledge and Smart Technology (KST), IEEE, 2022, pp. 75–79.
[18] Y. Zhu, L. Cui, Z. Ding, L. Li, Y. Liu, Z. Hao, Black box attack and network intrusion detection using machine learning for malicious traffic, Computers & Security (2022) 102922.
[19] A. Kumar, M. Shridhar, S. Swaminathan, T. J. Lim, Machine learning-based early detection of iot botnets using network-edge traffic, Computers & Security 117 (2022) 102693.
[20] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset, Future Generation Computer Systems 100 (2019) 779–796.
[21] M. Woźniak, J. Siłka, M. Wieczorek, M. Alrashoud, Recurrent Neural Network Model for IoT and Networking Malware Threat Detection, IEEE Transactions on Industrial Informatics 17 (2020) 5583–5594.
[22] I. Guarino, G. Bovenzi, D. Di Monda, G. Aceto, D. Ciuonzo, A. Pescape, On the use of machine learning approaches for the early classification in network intrusion detection, in: IEEE International Symposium on Measurements & Networking (M&N), 2022, pp. 1–6.
[23] UC Irvine, KDD Cup 1999 Data, 2022. URL: `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.
[24] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE symposium on computational intelligence for security and defense applications, Ieee, 2009, pp. 1–6.
[25] A. Dainotti, A. Pescapè, G. Ventre, A cascade architecture for dos attacks detection based on the wavelet transform, J. Comput. Secur. 17 (2009) 945–968. URL: `https://doi.org/10.3233/JCS-2009-0350`. doi:`10.3233/JCS-2009-0350`.
[26] A. Nascita, F. Cerasuolo, D. D. Monda, J. T. A. Garcia, A. Montieri, A. Pescapè, Machine and deep learning approaches for iot attack classification, in: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2022, pp. 1–6.
[27] A. Goodge, B. Hooi, S.-K. Ng, W. S. Ng, Robustness of Autoencoders for Anomaly Detection Under Adversarial Impact., in: 29th International Joint Conference on Artificial Intelligence (IJCAI), 2020, pp. 1244–1250.
[28] M. Kravchik, L. Demetrio, B. Biggio, A. Shabtai, Practical evaluation of poisoning attacks on online anomaly detectors in industrial control systems, Computers & Security (2022) 102901.
[29] K. H. Kim, S. Shim, Y. Lim, J. Jeon, J. Choi, B. Kim, A. S. Yoon, Rapp: Novelty detection with reconstruction along projection pathway, in: International Conference on Learning Representations (ICLR), 2019, pp. 1–14.
[30] F. A. Yerlikaya, Ş. Bahtiyar, Data poisoning attacks against machine learning algorithms, Expert Systems with Applications 208 (2022) 118101.
[31] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, J. D. Tygar, Adversarial machine learning, in: 4th ACM workshop on Security and artificial intelligence (AISec), 2011, pp. 43–58.
[32] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, J. D. Tygar, Antidote: understanding and defending against poisoning of anomaly detectors, in: 9th ACM SIGCOMM Conference on Internet Measurement (IMC), 2009, pp. 1–14.
[33] G. Apruzzese, M. Colajanni, L. Ferretti, M. Marchetti, Addressing adversarial attacks against security systems based on machine learning, in: 11th IEEE International Conference on Cyber Conflict (CyCon), volume 900, 2019, pp. 1–18.
[34] O. Alrawi, C. Lever, M. Antonakakis, F. Monrose, SoK: security evaluation of home-based iot deployments, in: IEEE Symposium on Security and Privacy (SP), 2019, pp. 1362–1380.

Appendix A. ROC Results on Poisoning

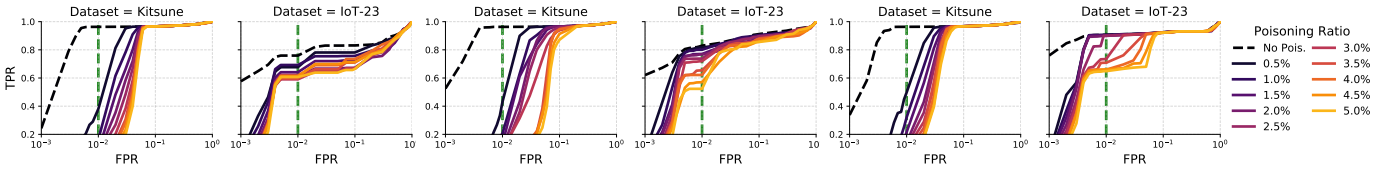
In this Appendix, we provide an analysis of the effect of LFAs for (*i*) a larger set of detectors, (*ii*) the entire poisoning range, and (*iii*) varying FPR values (via the ROC for each AD). Specifically, in Fig. A.12 we show ROC curves for ML models, `AE-2` and `KitNET-2` by varying the distance score. A similar analysis is provided for `KitNET-2` enhancements related to the ensemble stage, namely the equalization and the normalization in Fig. A.13. Finally, in Fig. A.14, we investigate the effect of poisoning on `KitNET-2` by focusing on the output stage, namely the selection of different train distance metrics.



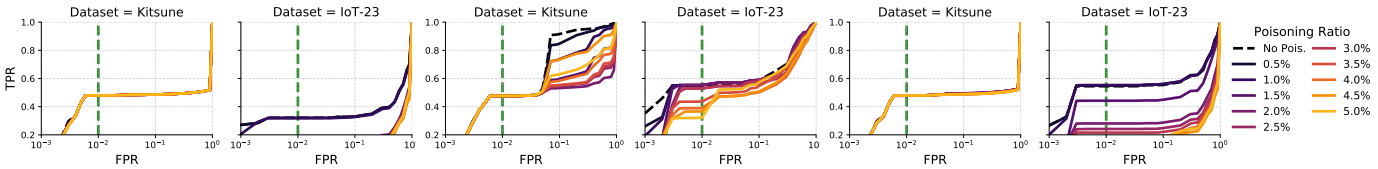
(a) IF. (b) LOF. (c) OC-SVM.



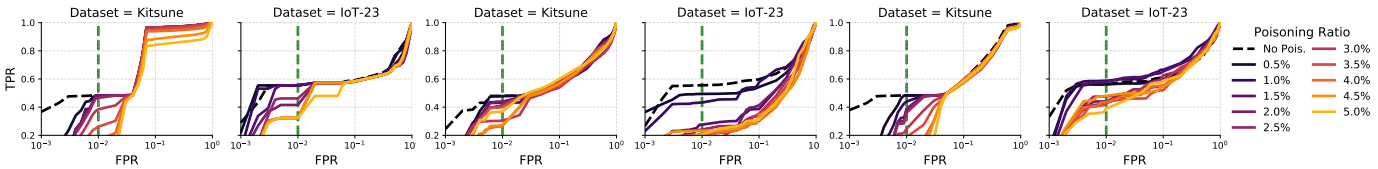
(d) AE-2 +RMSE_{EXT}. (e) AE-2 +RMSE_{INT}. (f) AE-2 +SAP.



(g) AE-2 +MHLN_{EXT}. (h) AE-2 +MHLN_{INT}. (i) AE-2 +NAP.

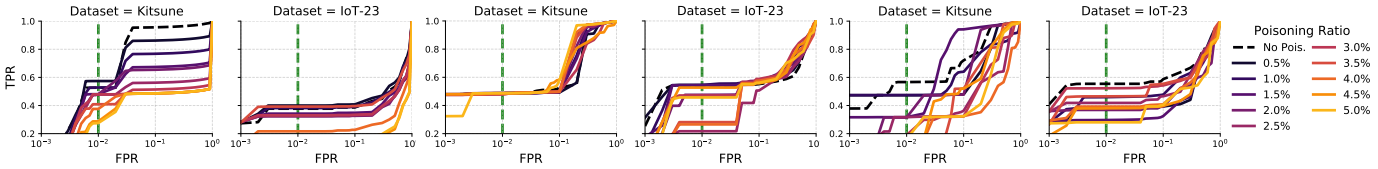


(j) KitNET-2 +RMSE_{EXT}. (k) KitNET-2 +RMSE_{INT}. (l) KitNET-2 +SAP.

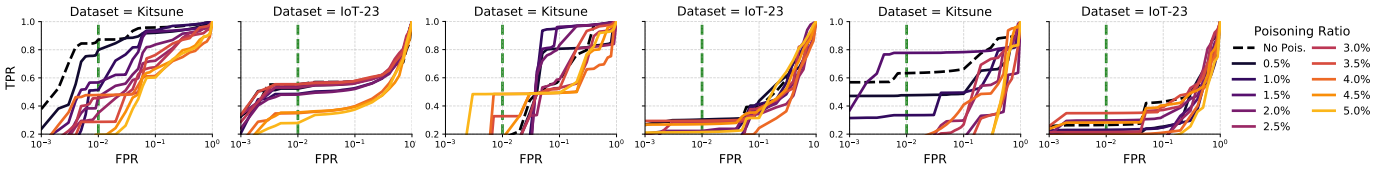


(m) KitNET-2 +MHLN_{EXT}. (n) KitNET-2 +MHLN_{INT}. (o) KitNET-2 +NAP.

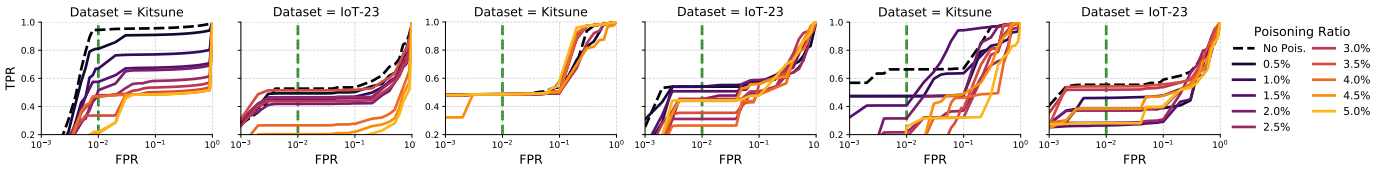
Figure A.12: ROC curves for ML models, AE-2 and KitNET-2 varying the distance score, showing robustness to poisoning. Results are shown as *avg*. over 10 folds.



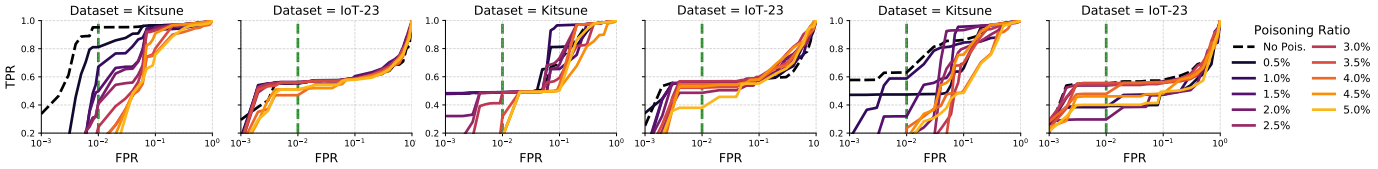
(a) KitNET-2 +RMSE_{EXT} +Equ. (b) KitNET-2 +RMSE_{EXT} +Nor. (c) KitNET-2 +RMSE_{EXT} +Equ+Nor.



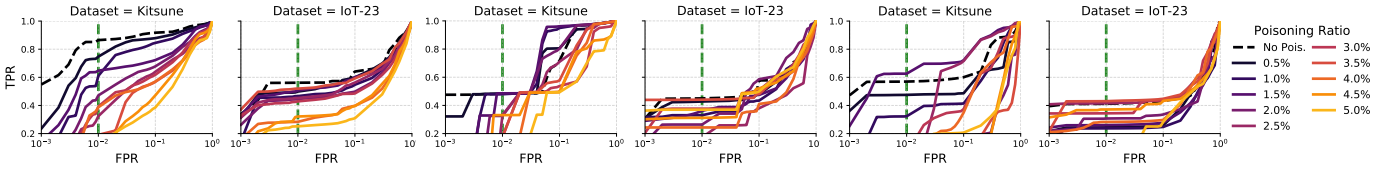
(d) KitNET-2 +RMSE_{INT} +Equ. (e) KitNET-2 +RMSE_{INT} +Nor. (f) KitNET-2 +RMSE_{INT} +Equ+Nor.



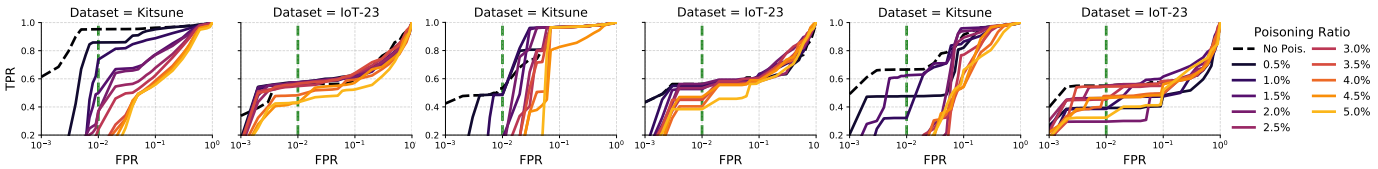
(g) KitNET-2 +SAP +Equ. (h) KitNET-2 +SAP +Nor. (i) KitNET-2 +SAP +Equ+Nor.



(j) KitNET-2 +MHLN_{EXT} +Equ. (k) KitNET-2 +MHLN_{EXT} +Nor. (l) KitNET-2 +MHLN_{EXT} +Equ+Nor.

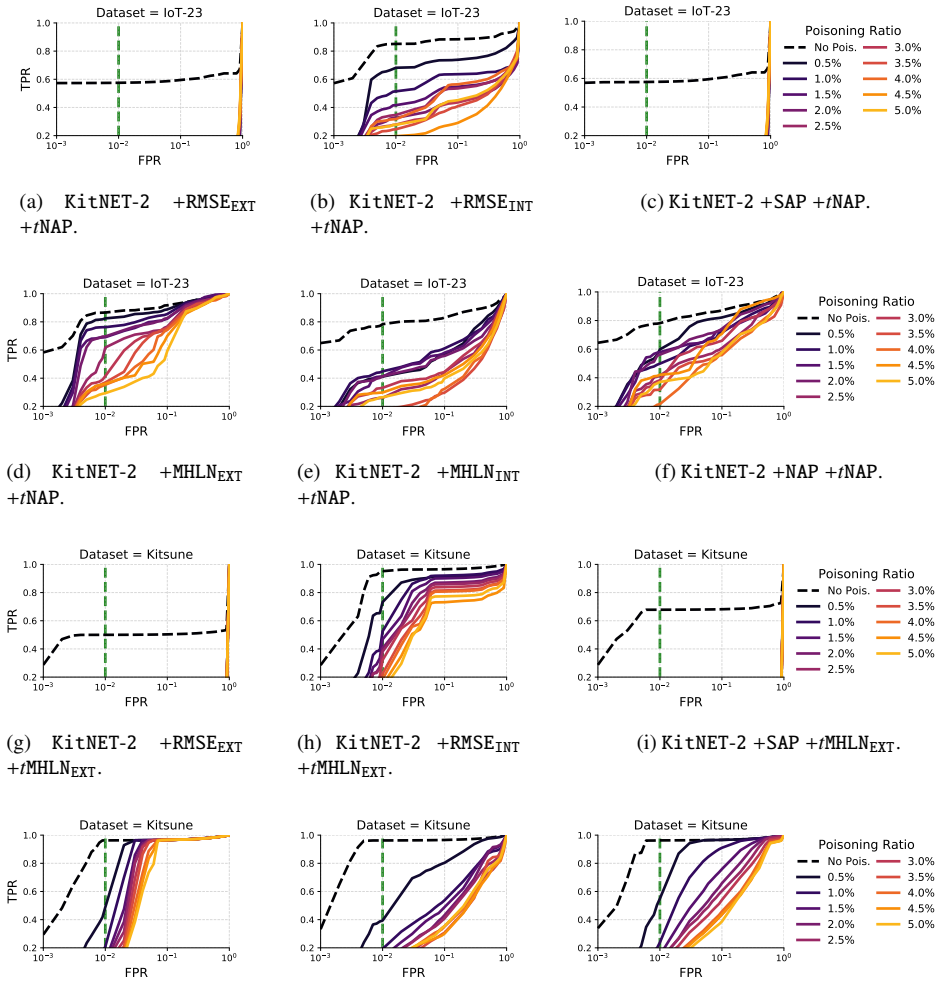


(m) KitNET-2 +MHLN_{INT} +Equ. (n) KitNET-2 +MHLN_{INT} +Nor. (o) KitNET-2 +MHLN_{INT} +Equ+Nor.



(p) KitNET-2 +NAP +Equ. (q) KitNET-2 +NAP +Nor. (r) KitNET-2 +NAP +Equ+Nor.

Figure A.13: ROC curves for KitNET-2 enhancements related to the ensemble layer, namely the equalization and the normalization, varying the distance score, showing robustness to poisoning. Results are shown as *avg*. over 10 folds.



(a) KitNET-2 +RMSE$_{\text{EXT}}$ +tNAP.

(b) KitNET-2 +RMSE$_{\text{INT}}$ +tNAP.

(c) KitNET-2 +SAP +tNAP.

(d) KitNET-2 +MHLN$_{\text{EXT}}$ +tNAP.

(e) KitNET-2 +MHLN$_{\text{INT}}$ +tNAP.

(f) KitNET-2 +NAP +tNAP.

(g) KitNET-2 +RMSE$_{\text{EXT}}$ +tMHLN$_{\text{EXT}}$.

(h) KitNET-2 +RMSE$_{\text{INT}}$ +tMHLN$_{\text{EXT}}$.

(i) KitNET-2 +SAP +tMHLN$_{\text{EXT}}$.

(j) KitNET-2 +MHLN$_{\text{EXT}}$ +tMHLN$_{\text{EXT}}$.

(k) KitNET-2 +MHLN$_{\text{INT}}$ +tMHLN$_{\text{EXT}}$.

(l) KitNET-2 +NAP +tMHLN$_{\text{EXT}}$.

Figure A.14: ROC curves for KitNET-2 enhancement related to the output layer, namely the selection of a different train distance metric (preceded by a t), varying the distance score, showing robustness to poisoning. Results are shown as *avg*. over 10 folds.