# Data Poisoning Attacks against Autoencoder-based Anomaly Detection Models: a Robustness Analysis

Giampaolo Bovenzi, Alessio Foggia, Salvatore Santella, Alessandro Testa, Valerio Persico, Antonio Pescapé
University of Napoli "Federico II" (Italy)
{giampaolo.bovenzi, valerio.persico, pescape}@unina.it, {ales.foggia, sa.santella, alessandro.testa}@studenti.unina.it,

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838942

***Abstract*—The Internet of Things (IoT) is experiencing a strong growth in both industrial and consumer scenarios. At the same time, the devices taking part in delivering IoT services—usually characterized by limited hardware and software resources—are more and more targeted by cyberattacks. This calls for designing and evaluating new approaches for protecting IoT systems, which are challenged by the limited computational capabilities of devices and by the scarce availability of reliable datasets. In line with this need, in this paper we compare three state-of-the-art machine-learning models used for Anomaly Detection based on *autoencoders*, i.e. shallow Autoencoder, Deep Autoencoder (DAE), and Ensemble of Autoencoders (viz. KitNET). In addition, we evaluate the robustness of such solutions when Data Poisoning Attack (DPA) occurs, to assess the detection performance when the benign traffic used for learning the legitimate behavior of devices is mixed to malicious traffic. The evaluation relies on the public *Kitsune Network Attack Dataset*. Results reveal that the models do not differ in performance when trained with unpoisoned benign traffic, reaching (at $1\%$ FPR) an F1 score of $\approx 97\%$. However, when DPA occurs, DAE proves to be the more robust in detection, showing more than $50\%$ of F1 Score with $10\%$ poisoning. Instead, the other models show strong performance drops (down to $\approx 20\%$ F1 Score) by injecting only $0.5\%$ of the malicious traffic.**

## I. Introduction

In the last few years, the Internet of Things (IoT) has experienced a serious growth, resulting in a global network of interconnected devices, and gaining a paramount role among future technologies [1]. IoT is particularly employed in industrial scenarios, with machine-to-machine connections that are expected to grow up to $14.7$ billion by $2023$ ($2.4\times$ w.r.t. 2018) [2]. However, IoT devices often suffer by limited computing, storing, and networking capabilities, thus turning out to be more vulnerable compared with other endpoint devices such as smartphones and tablets [3]. According to the widespread of IoT devices a rise of novel and diverse attacks is observed, targeting IoT infrastructures, and possibly compromising IoT systems and services [3, 4]. Hence, because traditional defense mechanisms for known attacks may be efficient only in specific scenarios, detecting and preventing various types of malware (e.g., botnet, adware) has become extremely important even for the IoT infrastructures, requiring the adaptation of underlying methodologies to the IoT context constraints to proper act as a second line of defense beyond the base security (e.g., encryption, authentication) [5].

To address these challenges, many Artificial Neural Network (ANN)-based techniques have recently been proposed to improve Anomaly Detection (AD) capabilities of Network Intrusion Detection Systems (NIDS), also tackling the constraints imposed by the limitations characterizing IoT devices [5]. However, despite ANN-based approaches have proved capable of learning complex patterns in data (e.g., fostering network traffic classification and attack signature detection [6, 7]), they $(i)$ typically require huge hardware and time resources for their training, and $(ii)$ rely on supervised learning techniques, thus requiring labeled traffic (which in turn often demands experts to manually intervene to associate traffic objects to the class of traffic they belong to) and the update of the model when a new traffic class (e.g., a new attack) has to be considered [8]. The scientific literature provides a number of examples where autoencoder-based models are used to learn the legitimate behavior of (IoT) devices [8, 9, 10, 11], and are able to improve the classical outlier detector models (e.g., Isolation Forest, and One-Class Support Vector Machine) reaching near-ideal detection rate and incurring very low error. In fact, the adoption of autoencoders as building blocks for NIDS jointly tackles and mitigates the issues mentioned above: $(i)$ autoencoders are inherently simple ANN models, which are composed by low-complexity layers, demanding limited computing resources; $(ii)$ autoencoders allow one to leverage unsupervised learning paradigm, thus not requiring labeled traffic for training.

Specifically, in [8] authors proposed a deep autoencoder for AD, achieving a very high accuracy in detecting malicious traffic. On the same extent, in [9] a double-stage ensemble of shallow autoencoders (viz. KitNET) is proposed, where each autoencoder belong the first stage is fed with a subset of the features, and then reconstruction errors (i.e. RMSE) from the first stage autoencoders are fed to a single autoencoder at stage two (voting). Recently, in [10] a variant of autoencoder-based approach for AD is proposed, named M2-DAE (Multi-Modal AutoEncoder). This model is evaluated against a deep autoencoder, showing no performance degradation, but reduced model size. Finally, the usage of a shallow autoencoder to detect DDoS attacks is evaluated in [11], achieving $82\%$ detection rate with 0 false positive rate.

To the best of our knowledge, existing works barely optimize model hyperparameters without providing hints about the model generalization capabilities [9, 11]. Moreover, they do not evaluate neither the training and the inference time [10, 11] or the size of detectors [8, 9, 11].

Due to the unsupervised training nature of autoencoders, the injection of malicious data in the training phase, named

*Data Poisoning Attack* (DPA), is more effective than in the supervised scenario. In fact, no labeling is required and malicious traffic could be injected with no knowledge of the underlying model and the monitored features (i.e. black-box attack). In this direction, several works have evaluated the robustness of autoencoder-based anomaly and intrusion detection solutions [12, 13].

Based on the numerous benefits that directly come from the adoption of autoencoders as core components of network AD engines, in this paper we investigate and fairly compare the performance of three state-of-the-art autoencoder-based solutions in the context of IoT systems, namely a *shallow autoencoder*, a *deep autoencoder*, and an *ensemble of autoencoders* (i.e. *KitNET*), Moreover, we test the robustness of these models against DPAs, emulated by the injection of malicious (attack) traffic into that used for the unsupervised training. This scenario strongly adheres with real deployments of similar NIDS solutions, where AD models are trained assuming that no infected devices are active in the monitored network [8]. To foster reproducibility, perform our study relies on the public *Kitsune Network Attack Dataset* which contains 9 different network attacks on a commercial IP-based surveillance system and an IoT network. Also, we release the code used for experimental evaluation.[1]

In the following, in Sec. II the models under investigation are detailed and the DPA treat model is discussed. Results both in clean- and adversarial-training scenarios are reported in Sec. III-C together with the details of the experimental setup to foster reproducibility. Finally, Sec. IV concludes our work with an eye on future perspectives.

## II. Autoencoder-based Anomaly Detection and Data Poisoning Attacks

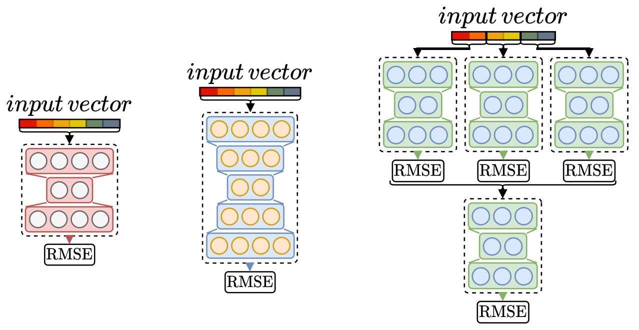
In this section we present the three autoencoder-based models for AD we consider and the threat model to evaluate their robustness when DPA occurs. First, we introduce the pre-processing steps the network traffic must be subjected to, in order to feed an anomaly detector model.

### A. Network Traffic Pre-Processing

Pre-processing steps parse raw network traffic to obtain the information in the format to feed the Machine Learning (ML) algorithms. First, network traces are split into *traffic objects*—which are collections of network packets sharing common characteristics. In this work we select biflows as traffic objects (i.e. set of packets sharing the same quintuple {`Src IP`, `Src Port`, `Dst IP`, `Dst Port`, `Proto`} where source and destination addresses and ports of the quintuple can be swapped [10]), in order to capture the bidirectional communication pattern between sender and receiver.

Then, *feature extraction* is performed computing (experts-defined) characteristics related to traffic objects. Selected features can encompass different aspects of interest, such as a 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 the 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 subjected to encryption. In addition, it is worth to underline that the features can be either extracted considering the entire traffic object (viz. post-mortem detection), or from incremental snapshots. Although the effectiveness of the model is impacted by which/how features are selected, it is worth to notice that their architecture (discussed in the next section) does not directly depend upon the specific selection but on their number.

[1]https://github.com/jmpr0/dpa-autoencoders.



(a) AE. (b) DAE. (c) KitNET.

Fig. 1. Autoencoder-based models for Anomaly Detection.

### B. Autoencoder-based Solutions for Anomaly Detection

An *autoencoder* is a generative neural network made of two principal components, an encoder and a decoder. The former computes a compression (encoding) of the input vector (viz. mapping of the feature space to a latent space) and the latter tries to reconstruct the input vector starting from the encoder output (viz. inverse mapping from the latent space to the feature space). In the following, we describe the autoencoder-based models we leverage for AD. We take into account three state-of-the-art autoencoder-based anomaly detectors. In particular, we consider two models which significantly differ for their complexity (in terms of number of layers): *shallow Autoencoder* (AE) [11] (Fig. 1a), and *Deep Autoencoder* (DAE) [8] (Fig. 1b). In addition, we evaluate an *Ensemble of Autoencoders* (Fig. 1c) named KitNET [9] that is composed of two stages: ($i$) the ensemble stage, which is constituted of several autoencoders, each one reconstructing a portion of the used features, and ($ii$) the output stage, which is an autoencoder that enforces a non-linear voting mechanism by reconstructing reconstruction errors from the ensemble stage.

If one trains an autoencoder to reconstruct benign traffic, the error it commits in reconstructing the input vector can be used as a measure of anomaly, and therefore adopted for AD. In fact, this error can be measured via arbitrary metrics, e.g., Root Mean Squared Error (RMSE) as shown in Fig. 1, and when it is bigger than a given threshold (e.g., $\mu_{\mathrm{RMSE}_{benign}} + \sigma_{\mathrm{RMSE}_{benign}}$, where $\mathrm{RMSE}_{benign}$ refers to the error in reconstructing (training) benign traffic [8]), one

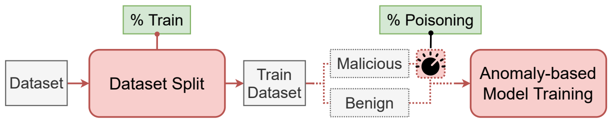


Fig. 2. Threat model workflow. The arrows follow the flowing of data (gray boxes) and the dots connect functions (red rounded boxes) to related attributes (green boxes). The dotted lines denote info which are available only in controlled scenarios.

assumes that an anomalous event is occurring—i.e. the IoT device has been potentially compromised because it exhibits a behavior that deviates from the one expected and allowed.

It is worth to notice that although AD solutions are based on the unsupervised learning paradigm, where the labeling of the training dataset is no longer required for modeling, their training phase bases on the *cleanliness* assumption: i.e. the network traffic adopted for the training phase, although not labeled, is assumed to be *non-anomalous* (i.e. benign). In practice, the effectiveness of the trained model can be undermined in the case this cleanliness assumption does not hold, e.g., when the traffic generated by one or more infected devices is injected in the network and is mistaken for benign traffic, as its presence would shift the learned behavior. On this basis, in the following we discuss the treat model detailing data poisoning attacks.

### C. Threat Model – Data Poisoning Attack

In this section, we focus on DPA and describe the *threat model* and the application scenario for the evaluated unsupervised-learning AD solutions (Fig. 2). The majority of recent works proposing autoencoder-based AD assume the cleanliness of the traffic used for training the models [8, 9, 10].

However, this assumption introduces some non-trivial practical constraints to be guaranteed, e.g., it requires that IoT network's normal traffic is collected immediately following the installation of devices in the network [8]. In addition, this assumption might not be guaranteed in practice, for instance when an attacker is able to inject malicious traffic into that leveraged for training the models, thus introducing potential flaws in the training phase of autoencoder models.

In general terms, a DPA is an attack from the Adversarial ML family and happens when an attacker properly crafts input feature samples and injects them in the training dataset to deviate the training phase of ML models. The goal of such attacker is to fool the ML model, with the aim of causing prediction errors, e.g., mistaking legitimate behavior for malicious and vice versa. Interestingly, DPA could evolve in a DoS for legitimate users [14] when benign traffic is misclassified (i.e. *availability disruption*), or could assure evasion when malicious traffic is confused with benign (i.e. *integrity disruption*). In detail, DPA is a *causative* attack, which means that the attacker can alter the training data used by the detector, ranging from the direct control of portion of these data to the influence in the data generation path [15].

According to Huang et al. [15], limitations on attackers in performing DPA are related to the knowledge of ($i$) the learning algorithm, ($ii$) the feature space, and ($iii$) the training and evaluation data. However, the first two limitations are faraway to be a problem with the increasing of security-by-design policies which foster the open-source deployment of solutions. Then, the last limitation does not apply to our AD scenario because we assume that training and evaluation data are automatically extracted from the network traffic, which is directly poisoned by the attacker: the same pre-processing is applied to the overall traffic with no means to distinguish the injected one. Defensive solutions to mitigate DPA are practically two [15]: ($i$) removing malicious data from the training dataset and ($ii$) designing the learning algorithm that are resistant to malicious training data. In line with the latter, the usage of multiple classifiers (i.e. ensemble learning) results in higher robustness to training-time attacks [16], like DPA.

According to the issues mentioned, in this paper we evaluate the robustness of the three autoencoder-based anomaly detectors (described in Sec. II-B) against DPA. More specifically, starting from the commonly assumed traffic cleanliness, we consider the scenario where an AD solution is trained assuming that input traffic is benign. Then, we emulate the presence of an attacker performing DPA by purposely injecting in the training dataset an increasing percentage of malicious traffic.

TABLE I
KITSUNE2017 DATASET COMPOSITION (AFTER DUPLICATE DROPPING).

| **Type** | **Category** | **Class** | **# of Biflows** |
|---|---|---|---|
| Benign | | | 13101 |
| Malicious | Botnet | Mirai | 3152 |
| | DoS | SSDP Flood | 65533 |
| | | SYN DoS | 467 |
| | | SSL Renegotiation | 491 |
| | MitM | ARP MitM | 48 |
| | | Active Wiretap | 47 |
| | | Video Injection | 5 |
| | Reconn. | Fuzzing | 21 |
| | | OS Scan | 33 |
| **Entire Dataset** | | | 82898 |

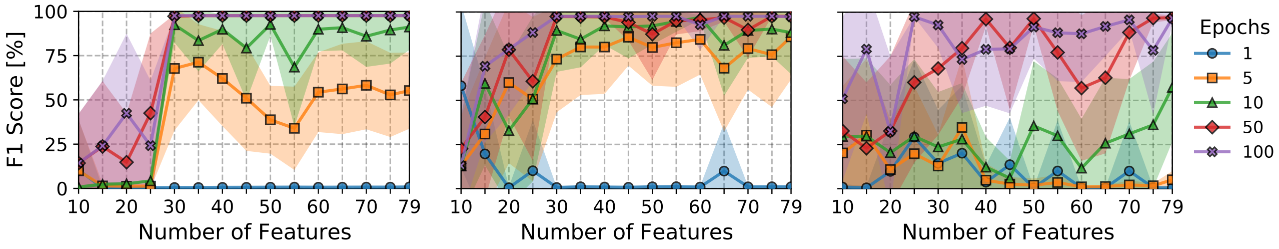
## III. Experimental Evaluation

In this section, we first describe in Sec. III-A the public dataset we leveraged, then we provide details to foster reproducibility in Sec. III-B. Finally in Sec. III-C we present the results.

### A. Dataset

Our experimental evaluation relies on the *Kitsune Network Attack Dataset* [9], which is a collection of benign and malicious network packets, counting 9 attack classes grouped into 4 categories, as listed in Tab. I. The dataset was captured from an IP-based commercial surveillance system and from a network composed by IoT devices. In detail, authors release raw traffic data (PCAP format) and per-packet labeling information. Further details are available in the dataset paper [9].

**Dataset Pre-Processing.** As we consider biflows, we first split PCAP traces according to the network quintuple. Then, we exploit per-packet labeling to label each biflow. To address inconsistencies encountered in case of biflows containing both benign and malicious packets, we adopt a fixed threshold mechanism to consistently label mixed biflows as either benign



(a) AE. (b) DAE. (c) KitNET.

Fig. 3. F1 score (at FPR=1%) of autoencoder-based models by varying the number of selected features (from 10 to 79 with step 5) and the number of training epochs (in $[1, 5, 10, 50, 100]$). Values are reported as $avg. \pm std.$ over 10-folds.

or malicious. In detail, we consider a biflow as malicious when malicious packets are more than the $1\%$ of the benign ones. After that, we extract several per-biflow characteristics, which are ($i$) *Number of Packets*, ($ii$) *(L4) Payload Length* (PL), ($iii$) *Inter-Arrival-Time* (IAT), ($iv$) *Time To Live*, ($v$) *TCP Window*, ($vi$) *TCP Flags* (FLG), and ($vii$) *Byte Rate*. For each time-series characteristic ($ii$-$v$) we compute 17 statistics, namely *minimum*, *maximum*, *average*, *standard deviation*, *mad*, *kurtosis*, *skew*, *variance*, and *percentiles* from $10^{th}$ to $90^{th}$ with step 10. Moreover, for the sole PL we report its *summation* (viz. total number of (L4) payload bytes), instead for the IAT we discard the first value from each biflow (viz. zero IAT values) to compute statistics. Next, to tackle the FLG we use 8 counters, reporting the number of occurrences for each TCP flag. Overall, the dataset we extracted counts 79 unique features. Table I summarizes the composition of the dataset.[2]

## B. Experimental Setup

In this section, we provide details about used tools, procedures, evaluation metrics, and input pre-processing.

First, the experimental evaluation is performed by the application of a 10-fold stratified cross-validation procedure, in order to better assess detection capabilities.

The sensitivity analysis to the number of features is enforced by applying a state-of-the-art feature selection algorithm [17], designed to cope with unsupervised learning scenarios. In detail, it is based on the computation of the ratio between the Arithmetic Mean (AM) and the Geometric Mean (GM) for each feature, by considering the exponential of values, obtaining the so called AMGM (dispersion) measure which is used to rank features: the higher this value, the more relevant the feature.

AD performance is evaluated through the classical True Positive Rate (TPR) and False Positive Rate (FPR) which are the rate of malicious biflows labeled as malicious, and the rate of benign biflows labeled as malicious, respectively. These pieces of information are summarized by mean of the ROC curves and the F1 Score (the harmonic mean of benign and

[2]To conduct an unbiased evaluation of detectors, we clean the obtained dataset by discarding ($i$) duplicate biflows (43.84% of the original dataset), ($ii$) biflows showing different labels (0.02%), and ($iii$) biflows with only one packet (0.06%). This filtering mechanism reduces the number of biflows from 147k to 82k. Noteworthy, we remove 82.37% of benign traffic, but only 5.1% of malicious one.

TABLE II
DETAILS ABOUT USED AUTOENCODER-BASED DETECTORS.

| **Name** | **Configuration** |
|---|---|
| AE | $I(n)$; $D(0.75{\times}n, R)$; $D(n, S)$. |
| DAE | $I(n)$; $D(0.75{\times}n, R)$; $D(0.50{\times}n, R)$; $D(0.33{\times}n, R)$; $D(0.25{\times}n, R)$; $D(0.33{\times}n, R)$; $D(0.50{\times}n, R)$; $D(0.75{\times}n, R)$; $D(n, S)$. |
| KitNET | 5 *ensemble* autoencoders: $I(0.20{\times}n)$; $D(0.75{\times}0.20{\times}n, R)$; $D(0.20{\times}n, S)$. *output* autoencoder: $I(5)$; $D(4, R)$; $D(5, S)$. |

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

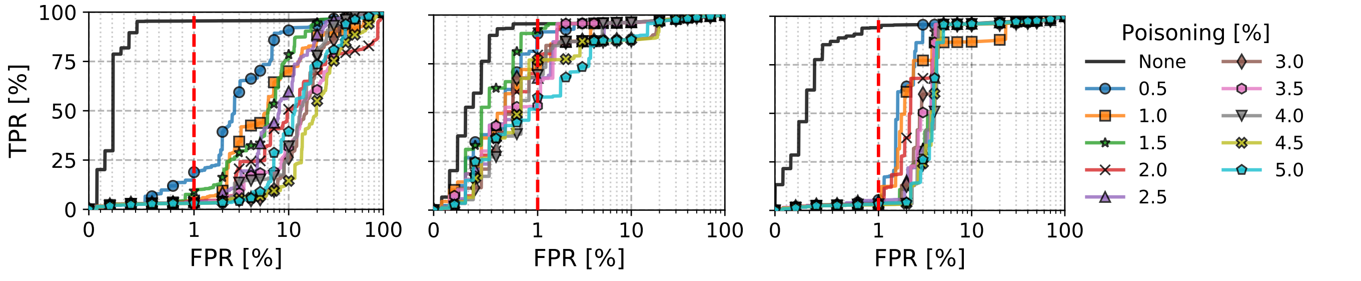
malicious recalls evaluated at $1\%$ of FPR). In the following, when evaluating the F1 Score we imply $1\%$ FPR.

The prototypes we used are written in `Python`, leveraging the `tensorflow` library. Details about the configuration of each model are reported in Tab. II. All the models are trained using an Adam optimizer with a Learning Rate of $0.001$ and the MSE (Mean Squared Error) as loss function, the batch size is set to 32, and we tested training lasting for 1, 5, 10, 50, or $100$ epochs. Before feeding the autoencoder-based models, we apply a Min-Max scaling to project each feature in the range $[0, 1]$. It is worth to underline that the scaling is applied by fitting the sole training set. The same scaling is also applied to the intermediate RMSEs which are computed from the KitNET ensemble stage, to feed its output stage.

## C. Experimental Results

In this section, we report the results of our experimental campaign. In particular, we first assess the performance of the model (by evaluating the F1 Score when fixing the AD threshold to meet $1\%$ FPR) and inspect the impact of hyperparameters. Then, we consider the impact of DPA by varying the amount of malicious traffic injected during the training phase of autoencoder-based models. Finally, we report a comparison among autoencoder-based models in terms of time and spatial complexity, showing the per-epoch training time, the per-sample inference time, as well as the number of trainable parameters against the number of features.

**Model Performance and Sensitivity to Hyperparameters.** In this section, we evaluate the AD performance of autoencoder-based models and report their sensitivity to two hyperparameters, namely ($i$) the number of features (which is the input size and directly impact the spatial complexity of the models)



(a) AE. (b) DAE. (c) KitNET.

Fig. 4. ROC Curve for the autoencoder-based models by varying the percentage of poisoning from 0.5% to 5%. "None" represents the training under cleanliness assumption. Values are reported as $avg.$ over 10-folds, to better understand trends.

TABLE III
F1 SCORE FOR THE BEST CONFIGURATIONS OF MODELS.

| Model | # Features | # Epochs | F1 Score [%] |
|---|---|---|---|
| AE | 60 | 50 | $97.61 \pm 0.02$ |
| DAE | 65 | 100 | $97.57 \pm 0.07$ |
| KitNET | 25 | 100 | $97.40 \pm 0.17$ |

and $(ii)$ the number of epochs used for training (the more the epochs the higher the time complexity of the training of models).

Accordingly, we show the achieved F1 Score against the number of features, for AE, DAE, and KitNET, in Figs. 3a, 3b, and 3c respectively. The number of considered features varies from 10 to 79 with step 5 (5 features are not considered because the KitNET results in a degraded model). In addition, the number of epochs considered to train the models varies among 1, 5, 10, 50, and 100. We recall that features are those defined in Sec. II-A and that the feature selection is enforced by means of the AMGM ranking (Sec. III-B).

In detail, both AE and DAE (Figs. 3a and 3b) show a saturation of the F1 Score with just 30 features, while KitNET tends to settle with 25-40 features (Fig. 3c). Considering the number of epochs, a trend emerges for all the models: the larger the number of epochs, the higher the F1 Score. In detail, by focusing on more than 25 features, the F1 Score of AE tends to settle for 50 or more epochs and the F1 Score of DAE shows no significant increment with 10 or more epochs. For KitNET the results show remarkably higher variation, with best F1 Score achieved with 100 epochs.

Finally, we report the best configuration performance, by selecting the best combination (# Features, # Epochs), in Tab. III. It is clear that all the models show a similar F1 Score, with AE requiring the fewest epochs and KitNET requiring the fewest features.

**Model Robustness Against Data Poisoning Attack.** In the previous section we confirm that the three autoencoder-based models show similar detection performance, notwithstanding the different best configurations. To further evaluate their effectiveness, in this section we assess their robustness against DPAs. As explained in Sec. II-C, to perform this analysis we remove the network cleanliness assumption, by injecting into the training dataset used for model training an increasing

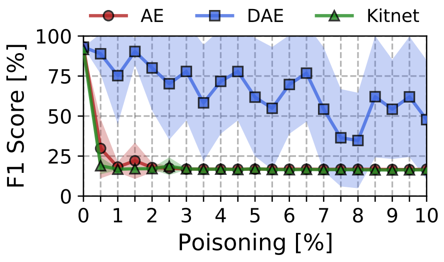


Fig. 5. F1 Score (at FPR=1%) for autoencoder-based models by varying the percentage of poisoning from 0% to 10% with step 0.5%. Values are reported as $avg. \pm std.$ over 10-folds.

percentage of malicious traffic—we randomly drawn malicious biflows from the dataset, maintaining balance between attack classes. In particular, we inject a percentage of poisoning samples varying in 0.5%–10% of the benign biflows number used for training. It is worth to underline that we leverage, for each model, the full set of features by fixing the number of epochs to 50. Despite the last choice does not impact performance, because passing from 50 to 100 epochs does not significantly improve the detection capabilities of such models, the selection of the entire set of features, namely 79, allows us to avoid considering the impact of malicious traffic injection on the features selection procedure, resulting in the unbiased evaluation of model robustness against DPA.

Accordingly, in Figs. 4a, 4b, and 4c we report the ROC curves for the three autoencoder-based models by limiting the poisoning up to 5% (for visualization). From Figs. 4a and 4c, looking at the 1% FPR (red dashed) line it is clear that AE and KitNET are the less resistant, with a very low TPR. On the contrary, DAE (Fig. 4b) shows more resilience, by scoring more than 50% TPR also when the poisoning reaches the 5%. An interesting outcome is the different behavior of evaluated models when the poisoning increases. In detail, both the AE and KitNET detection performance strongly degrades with only 0.5% poisoning, but both DAE and KitNET show little sensibility to the percentage of poisoning, with very close ROC curves when poisoning increases. Finally, Fig. 5 confirms previous findings when the poisoning ranges in 5%–10%, with the F1 Score of DAE that still resists worsening.

**Complexity of Autoencoder-based Models.** In this section we empirically evaluate the time and spatial complexity of the autoencoder-based models. In particular, for the time complexity we evaluate the per-epoch training time and the per-sample inference time by varying the number of features. Concerning the spatial complexity, we resort to the number of trainable parameters against the number of features.

As expected, the *training time per-epoch* increases when increasing the number of features for all the models. We recall that KitNET training time is the maximum training time of the autoencoders composing the ensemble stage (parallel training) plus the training of the sole output stage. It is evident that AE is the fastest model, followed by KitNET and DAE, scoring $2.16 \pm 0.02\,\mathrm{s}$, $2.80 \pm 0.12\,\mathrm{s}$, and $4.54 \pm 0.08\,\mathrm{s}$, respectively (using 79 features). Moreover, passing from 10 to 79 features, AE shows a relative increment of 67.86%, KitNET of 6.43%, while DAE scores 57.19%, electing KitNET as the less sensible to the input size growth.

On the same extent, we evaluate the *per-sample inference time* of each model (by averaging values observed in different folds and varying the input size). A first outcome is the very small impact of feature set size because of the very low standard deviation (less than $0.01\,\mathrm{s}$ for all the models). Moreover, it is clear that AE and DAE models are the fastest (scoring $0.09\,\mathrm{s}$ and $0.11\,\mathrm{s}$, respectively), with AE that slightly improve DAE, but KitNET is the slowest ($0.18\,\mathrm{s}$) because of the two sequential stages of autoencoders.

When focusing on the spatial complexity, the behavior of evaluated models strongly differs when the number of features increases. In fact, the *number of trainable parameters* of AE (DAE) presents a monotonic growing, from 178 (326) using 10 features to 9460 (17511) when the entire set of features is used, resulting in a huge relative growth of 5214% (5271%). On the contrary, the KitNET trainable parameters increase from 109 to 2053, experiencing a relative increment of 1783%. This characteristic would ease the deployment on resource-constrained devices.

**Discussion on Trained Models.** Based on the loss function behavior on the test set (viz. *test loss*), we can discuss some interesting properties of the models obtained in the considered setup. In detail, we consider training lasting for up to 100 epochs. The AE test loss stops decreasing at 50–100 epochs, resulting in a properly fitted models. On the contrary, the DAE test loss does not reach a plateau within 100 epochs, thus probably resulting in underfitting. Finally, the KitNET test losses (one per autoencoder of the two stages) show different trends: for the ensemble stage, the test loss of the autoencoders seems to depend on the ranking of the respective features, with models coping with best (high dispersion) features which struggle in settle at 100 epochs (underfitting), gradually obtaining a properly trained model when the features are lower in rank (low dispersion); for the output stage, the test loss of the autoencoder smooths in the range 10–20 epochs, but it remains stable up to 100 epochs. These findings suggest that it is worth to investigate the benefits of applying some balancing mechanism of the features fed to the ensemble stage autoencoders (e.g., balanced dispersion), uniforming the training phase of these. This balancing procedure may positively impact the training of the output stage autoencoder. We plan to investigate these aspects in the future.

## IV. Conclusion and Future Perspectives

In this paper we fairly compared and evaluated three state-of-the-art autoencoder-based solutions for AD and provided a robustness analysis against the poisoning of the dataset used for training. In detail, we evaluated the ability of AE, DAE, and KitNET in detecting anomalous biflows, by training them with benign traffic to model the legitimate behavior of IoT devices: when considering their best configuration, all the models show similar performance of $\approx 97\%$ F1 Score (at 1% FPR). Then, we considered DPA and evaluated the impact of malicious biflows injection during the training phase of such models, electing DAE as the more resilient model, disproving in the considered setup the effectiveness of ensemble (i.e. KitNET) in obtaining more robust models. Finally, we empirically evaluated the time and spatial complexity of tested solutions. Summarizing, KitNET results the most scalable model (in terms of training time and number of trainable parameters), despite it nearly doubles the inference time. In future work, we plan to evaluate the effectiveness of countermeasures against DPA, the per-attack poisoning evaluation, and the evaluation of such models in an online deployment.

## References

[1] I. Lee and K. Lee. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Elsevier Business Horizon*, 2015.

[2] Cisco Annual Internet Report (2018–2023) white paper, 2020.

[3] N. Neshenko, E. Bou-Harb, et al. Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations. *IEEE Communications Surveys & Tutorials*, 2019.

[4] M. Hasan, M. M. Islam, et al. Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. *Elsevier Internet of Things*, 2019.

[5] N. Chaabouni, M. Mosbah, et al. Network intrusion detection for IoT security based on learning techniques. *IEEE Communications Surveys & Tutorials*, 2019.

[6] P. M. S. Sánchez, J. M. J. Valero, et al. A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets. *IEEE Communications Surveys & Tutorials*, 2021.

[7] A. Dainotti, A. Pescapè, and K. C. Claffy. Issues and future directions in traffic classification. *IEEE Network*, 2012.

[8] Y. Meidan, M. Bohadana, et al. N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Perv. Computing*, 2018.

[9] Y. Mirsky, T. Doitshman, et al. Kitsune: An ensemble of autoencoders for online network intrusion detection. *Network and Distributed Systems Security (NDSS) Symposium*, 2018.

[10] G. Bovenzi, G. Aceto, et al. A hierarchical hybrid intrusion detection approach in IoT scenarios. *IEEE GLOBECOM*, 2020.

[11] K. Yang, J. Zhang, et al. Ddos attacks detection with autoencoder. In *IEEE/IFIP Network Op. and Man. Symposium*, 2020.

[12] P. Madani and N. Vlajic. Robustness of deep autoencoder in intrusion detection under adversarial contamination. In *ACM HoTSoS*, 2018.

[13] A. Goodge, B. Hooi, et al. Robustness of autoencoders for anomaly detection under adversarial impact. In *IJCAI*, 2020.

[14] B. I. Rubinstein, B. Nelson, et al. Antidote: understanding and defending against poisoning of anomaly detectors. In *ACM IMC*, 2009.

[15] L. Huang, A. D. Joseph, et al. Adversarial machine learning. In *ACM AISec*, 2011.

[16] G. Apruzzese, M. Colajanni, et al. Addressing adversarial attacks against security systems based on machine learning. In *IEEE CyCon*, 2019.

[17] A. J. Ferreira and M. A. Figueiredo. Efficient feature selection filters for high-dimensional data. *Elsevier Pattern Recognition Letters*, 2012.