Learning to Detect Anomalous Wireless Links in IoT Networks

Gregor Cerar\textsuperscript{1,2}, Halil Yetgin\textsuperscript{1,3}, and Carolina Fortuna\textsuperscript{1}
\textsuperscript{1}Department of Communication Systems, Jožef Stefan Institute, SI-1000 Ljubljana, Slovenia.
\textsuperscript{2}Jožef Stefan International Postgraduate School, Jamova 39, SI-1000 Ljubljana, Slovenia.
\textsuperscript{3}Department of Electrical and Electronics Engineering, Bitlis Eren University, 13000 Bitlis, Turkey.
\{gregor.cerar \ halil.yetgin | carolina.fortuna\}@ijs.si

Abstract—After decades of research, Internet of Things (IoT) is finally permeating real-life and helps improve the efficiency of infrastructures and processes as well as our health. As massive number of IoT devices are deployed, they naturally incurs great operational costs to ensure intended operations. To effectively handle such intended operations in massive IoT networks, automatic detection of malfunctioning, namely anomaly detection, becomes a critical but challenging task. In this paper, motivated by a real-world experimental IoT deployment, we introduce four types of wireless network anomalies that are identified at the link layer. We study the performance of threshold- and machine learning (ML)-based classifiers to automatically detect these anomalies. We examine the relative performance of three supervised and three unsupervised ML techniques on both non-encoded and encoded (autoencoder) feature representations. Our results demonstrate that; i) automatically generated features using autoencoders significantly outperform the non-encoded representations and can improve F1 score up to 500% and ii) among the best performing models based on F1 score, supervised ML models outperform the unsupervised counterpart models with about 18% on average for anomaly types SuddenD and SuddenR, and this trend also applies to SlowD and InstaD anomalies, albeit with a tiny margin.

Index Terms—Anomaly Detection, Internet of Things (IoT), Machine Learning (ML), Wireless Links, Wireless Networks

I. INTRODUCTION

The Internet of Things (IoT) has received a plethora of attention from both industry and academia due to the market release of a variety of smart devices on a regular basis, e.g. the devices retrofitted in home appliances, wearables, healthcare, vehicles and industrial machinery, just to name a few \cite{1}. To this end, extensive research efforts have been put forward for their active deployment and development to enable increasingly efficient and more automated operations in manufacturing, agriculture, transportation and healthcare, but also due to their massive economic contributions \cite{2}.

Valid business cases \cite{3} and successful real-world large-scale IoT deployments are emerging as a way to improve existing business processes as well as enable new applications \cite{2}. However, once the network of sensors is deployed, it becomes part of the operational infrastructure of a business, and needs to be maintained and serviced similar to any other infrastructure, such as legacy IT infrastructure, robots and machines just to name a few. Minimizing maintenance costs while ensuring the reliability of IoT network \cite{4} becomes prohibitive when the number of sensors are in their thousands or tens of thousands.

To efficiently manage such massive IoT networks, automatic IoT network monitoring \cite{5} and malfunction detection \cite{6} solutions that automatically report relevant malfunctions and filter them out without influencing the business process are required.

IoT network or node malfunctioning can also be referred to as network or node anomaly and to date, it has been defined in various ways, often from the perspective of monitored networking aspects. For instance, Sheth et al. \cite{6} define and identify anomalies from the IEEE 802.11 physical layer perspective, namely, hidden terminal, capture effect, noise and signal strength variation anomalies, whereas Gupta et al. \cite{7} define anomalies from multihop networking perspective with the aspects, such as black hole, sink hole, selective forwarding and flooding. Alipour et al. \cite{8} define the anomalies from IEEE 802.11 link layer security perspective with the focus on aspects, such as injection test, deauthentication attack, disassociation attack, association flood and authentication flood. Generally speaking, anomaly detection research in IoT networks can be found in the form of intrusion, fraud and fault detection, system health monitoring, event detection in sensor networks and detecting ecosystem disturbances \cite{9}, where most studies mainly concerned with a certain type of anomaly within a specific scenario.

In this paper, motivated by a real-world experimental IoT deployment, we introduce four types of IoT anomalies that can be identified at the link layer, namely sudden degradation, sudden degradation with recovery, instantaneous degradation and slow degradation. Rather than focusing on the cause of an anomaly as realized in \cite{6} and \cite{7}, we focus our attention on the observable symptoms of link measurements, namely the changes in the expected received signal. Based on the type of anomaly, we identify possible root causes that may be related to hardware, firmware and the channel, and develop models for automatically classifying the introduced anomalies. The main contributions of this paper are as follows.

1) Based on the gained knowledge while operating the LOG-a-TEC wireless experimentation testbed \cite{10}, we provide an analysis on real-world operational measurements that further stresses the need for automated anomaly detection in massive IoT networks.

2) We introduce four types of plausible anomalies gleaned from our experimental observations, identify their symptoms from the application perspective and potential
underlying causes.

3) We study the performance of threshold- and machine learning (ML)-based classifiers to detect the four types of anomalies introduced. To achieve this, we train the proposed classifiers with standard manually-engineered features (data representations) and with an autoencoder-based automatic feature generation approach, which outperformed the former.

4) We also analyse the relative performance of three supervised and three unsupervised ML techniques. More explicitly, we consider regression-based, tree-based and kernel-based methods as part of our supervised techniques, while nearest neighbours, tree- and kernel-based methods are leveraged as their unsupervised counterpart techniques.

5) We produce a publicly available anomaly detection toolset\(^1\) including entire procedures, e.g., anomaly injection into trace-sets, feature generation out of data representations, and model training and development.

This paper is structured as follows. Section II summarizes the related work and Section III introduces the four types of IoT network anomalies. Then, Section IV elaborates on various data representations that can be used to generate features for training the proposed ML models, whereas Section V describes the threshold-based approach as well as the selected supervised and unsupervised ML techniques. Section VI discusses the relevant methodological and experimental details, while Section VII provides thorough analyses of the results and discusses the limitations. Finally, Section IX concludes the paper.

II. RELATED WORK

We provide related work to the main contributions of this paper as follows. First, we discuss related works that define anomalies in wireless and IoT networks, then we stress on the use of autoencoders for improving various aspects of wireless networks including anomaly detection, and finally, we focus on ML models that support for improved operations of wireless networks.

A. Anomaly definitions in wireless networks

Generally speaking, an anomaly is defined as an outlier, a distant object, an exception, a surprise, an aberration or a peculiarity, depending on the domain, research community and specific application scenario \(^9\), \(^11\)–\(^15\). A widely used classification of anomalies, including in wireless sensor network research is provided in \(^9\), \(^16\), where three classes of anomalies are defined based on their nature; point anomalies, contextual anomalies and collective anomalies. In \(^14\), Gupta et al. classify relevant studies on outlier detection for time series data, one of which is the point outlier as defined in \(^9\), and others are subsequence outliers, global and local outliers. More recently, Lavin et al. \(^17\) introduce a benchmark for anomaly detection, and target mainly at cloud networks and associated services, where they provide reference datasets to be used when evaluating the performance of anomaly detection algorithms. While they do not specifically define the type of anomalies, their benchmark datasets include several anomalies.

Due to the spatio-temporal nature of wireless sensor network monitoring and data collection, Jurdak et al. \(^18\) introduce temporal, spatial and spatio-temporal anomalies as well as node, network and data anomalies, followed by even finer grained anomalies, such as node resets, node failures, etc. A number of studies then introduce more focused and application specific anomalies. For instance, Sheth et al. \(^6\) define and identify anomalies from the IEEE 802.11 physical layer perspective namely; hidden terminal, capture effect, noise and signal strength variation anomalies. Moreover, Gupta et al. \(^7\) define anomalies with the aspects of multihop networking, such as black hole, sink hole, selective forwarding and flooding, whereas Alipour et al. \(^8\) define anomalies from IEEE 802.11 link layer security aspects, such as injection test, deauthentication attack, disassociation attack, association flood and authentication flood. For further details, motivated readers are referred to \(^18\) for the diagnosis and detection of wireless network anomalies.

B. Autoencoders for improving wireless network operations and anomaly detection

With the advent of deep learning, one class of techniques belonging to this class of ML, referred to as autoencoders, has been proven to be particularly useful at performing automatic feature engineering also for time series data \(^19\). Autoencoders attempt to learn a lossless compression of the data and the code resulting from that compression represents a superior feature set.

Generally in wireless, autoencoders have been successfully applied by \(^20\) and their subsequent works, such as \(^21\) to accurately reconstruct physical layer signals and \(^22\) signal denoising for more accurate localization. For anomaly detection in wireless and IoT networks, Wang et al. \(^23\) proposed autoencoders for more accurate identification of faulty parts of WSNs, as well as faulty antennas in antenna arrays, whereas Shahid et al. and Chen et al. \(^24\), \(^25\) proposed autoencoders for identifying anomalies in wireless and IoT networks based on transport layer traces, and recently, Yin et al. \(^26\) proposed recurrent autoencoders for time series anomaly detection for IoT networks. However, they used a synthetic dataset with metrics derived from several Yahoo services. Unlike the state-of-the-art, this work proposes autoencoders as an automatic feature generation method for link layer anomaly detection and uses a real-world wireless dataset in which the introduced four types of anomalies are synthetically injected.

C. ML techniques for wireless and IoT network anomaly detection

In the literature, it is often a good practice that when a ML solution to a specific problem is considered, several

\(^1\)Script for the design and development of anomaly detection models: https://gist.github.com/gcuerar/0b03e55f41147a7b723d0f455d1f120946
counterpart ML models are evaluated against each other for performance analyses. For instance, Kieu et al. [19] compare the performance of ten different ML techniques, such as Support Vector Machines, Local Outlier Factor, Isolation Forest, just to name a few, on six different datasets that are suitable for anomaly detection.

With respect to wireless and IoT network anomalies, Thing et al. [27] evaluate the relative performance of four deep learning and one decision tree models for anomaly detection and attack classification in IEEE 802.11 networks, whereas Chen et al. [25] evaluate the relative performance of principal component analysis, standard and convolutional autoencoder for detecting anomalies in transport layer traces, i.e., TCP, UDP and ICMP of wireless networks. Moreover, Ran et al. [28] evaluate the relative performance of their proposed semi-supervised approach of IEEE.802.11 anomaly detection, and similarly Salem et al. [29] evaluate the relative performance of five ML techniques, i.e., SVM, decision trees (J48), logistic regression, Naïve Bayes, and Decision Table for anomaly detection in WSNs. Additionally, the previous authors [30] also evaluate the performance of their proposed algorithm against selected three ML techniques, namely linear regression, additive regression, and J48 decision tree for anomaly detection in WSNs. However, in most of the ML-based network anomaly detection research discussed in this section as well as in [31] provide only limited relative performance evaluation results. To the best of our knowledge, this paper is the first attempt to provide relative comparisons between three supervised and three unsupervised ML techniques based on various data representations and their encoded counterpart features.

III. MOTIVATION

Our lab runs the LOG-a-TEC [3] testbed that has empowered wireless experimentation for more than ten years. The first version of the testbed comprised of our custom embedded platform [32] was mounted on public light poles in a small municipality of Slovenia [33]. It included more than fifty nodes, most of which were situated in hard-to-reach locations. A sensor management system [10] is used to keep the record of each node for its hardware and software versions, configurations, and locations. This system also performs a number of management and diagnosis related tasks to monitor the operation of the devices.

Over time, the users of the testbed had difficulties in reaching some of the nodes or noticed unexplainable measurements collected during their testbed experimentation. For instance, the transceivers on some of the nodes were degraded significantly for their receiver sensitivity and transmit power performances, and in some cases to such a degree that they became inoperative. As depicted in Fig. 1a third node (ID-3) sensed transmissions from fifth node with received signal strength indicator (RSSI) of about -70 [dBm] on average till 2nd February of 2013. Following that, either fifth node’s transmit power or third node’s receiver sensitivity was degraded significantly, which was reduced to about -90 [dBm] on average. After investing a good amount of time and effort in understanding and reproducing the anomaly, the fifth node was diagnosed with a hardware failure, and it could only be restored to normal operation by replacing the integrated circuit for transceiver (TI CC2500).

Similarly, another anomaly type is experienced in Fig. 1b with a sudden degradation and there were several recovery attempts between February 15th and March 9th 2013. In this particular case, we figured out that the sixth node was accidentally downgraded in February to an older version of the firmware that had a bug in the spectrum sensing code, which directly affected the operations of the sixth node and degraded its transmit power. Fig. 1c presents several spike-like instantaneous degradation anomalies between nodes 12 and 15. We were not able to discover anything technically wrong with these respective nodes. Therefore, we assumed that these anomalies were probably due to weather and/or large objects moving around the radios, since these two devices were mounted in an industrial zone, where moving large trucks and massive long-term standing objects were not an uncommon occurrence, which can indeed incur spikes due to the instantaneous non-line-of-sight channels experienced. Finally, Fig. 1d also exhibits two distinguishable rapid drops and climbs, but most importantly, on average, shows a slightly degrading performance in sensitivity and/or transmit power between nodes 4 and 26 after December 2012. We were not able to readily justify such behaviour of the device, but ageing of electronic components may induce such behaviour, which is a well-known issue [34].

IV. WIRELESS NETWORK ANOMALIES

Wireless networks are designed to exchange data between two communicating parties, e.g., video, voice and sensor measurements. As long as the network remains functional and is not interrupted, all the devices within the network are considered ordinarily operable. When the devices are compromised as exemplified in Section III, then a degradation in the service quality is experienced. The way how anomalies affect the user’s service quality experience is stringently associated with the type of anomaly. Therefore, in this section, we introduce four types of anomalies that can be observed in communication links of wireless networks, which were mainly discovered in our evaluation of a real-world experimentation, as discussed in Section III: a) sudden degradation, b) sudden degradation with recovery, c) instantaneous degradation (spike) and d) slow degradation.

a) Sudden degradation (SuddenD): The sudden degradation anomaly can be mathematically represented by a step function with decreasing slope, as depicted in Fig. 2a. In our case, this represents a sudden persistent change in the state of a link. While this sudden change with an increasing slope is also possible in theory, typically it will only lead to a more reliable link, therefore they are not accounted as an anomaly.

Symptom: From the perspective of a user, services may become unavailable, offline and unreachable. From the perspective of a network, either the transmitter stops generating electromagnetic field or the receiver is unable to receive data.

2LOG-a-TEC testbed with sensor platforms http://log-a-tec.eu
(a) Sudden degradation with no recovery between Node 5 and Node 3.

(b) Sudden degradation with recovery between Node 6 and Node 13.

(c) Spike-like instantaneous degradation between Node 13 and Node 15.

(d) Slow degradation between Node 4 and Node 26.

Figure 1. Anomalies observed in operational environment, where solid black lines represent average RSSI and greyed areas show maximum/minimum values.

**Possible causes:** Such sudden degradation can be induced by a transceiver failure as discussed in Section III and depicted in Fig. 1a, a significant and sudden change in the position of one or both of the communicating parties leading them to remain disconnected, moving from line-of-sight to a non-line-of-sight environment with obstacles preserving electromagnetic shielding materials, and a significant hardware or software failure where built-in recovery mechanisms, such was watchdogs cannot be triggered.

**b) Sudden degradation with recovery (SuddenR):** The sudden degradation with recovery anomaly can be mathematically represented by a step function with decreasing slope, as depicted in Fig. 2b. In this case, the state of a link suddenly changes, stays in the new state for a longer period of time and ultimately returns to the previous state. In sudden degradation with recovery, communication is interrupted for a certain period of time.

**Symptom:** From user’s perspective, provided services may become sluggish and unavailable for a certain period of time and later resume back to their regular operations. From the perspective of the network, in the case of sudden degradation with recovery, either transmitter temporarily stops generating electromagnetic field or the receiver temporarily is unable to receive it.

**Possible causes:** This type of degradation can be caused by buffer congestion and software bug, as discussed in Section III and depicted in Fig. 1b, where watchdog performs reboot after a certain timeout, a radio remaining in excessive active state and requiring recalibration, an obstacle blocking the communication for some time, and a signal jammer equipped on a military vehicle that is passing by.

**c) Instantaneous degradation (InstaD):** The instantaneous degradation anomaly can be mathematically represented by a step function with steeply decreasing slope, forming a sudden spike, as depicted in Fig. 2c. In this case, the state of the link changes suddenly, but instantaneously returns to its previous state. The instantaneous degradation anomaly may appear as an information loss.

**Symptom:** From user’s perspective, a real-time service may experience instant lags, while other non-real-time services may work unaffected. From the perspective of the network, either transmitter experiences a deep fading instance or the receiver...
becomes unable to receive data due to an instant exposure to excessive noise or interference.

Possible causes: This type of degradation can be caused by an instant interference, collision, quantization errors, value reading errors or sudden saturations in the transceiver’s electronic components, as discussed in Section III and depicted in Fig. 1c where anomaly can be stringently induced by the issues related to the propagation environment, such as an external device communicating on the same frequency, excessive background noise and multipath fading, just to name a few.

d) Slow degradation (SlowD): The slow degradation anomaly can be mathematically represented as a normalized linear function with slightly decreasing slope, as depicted in Fig. 2d. In this case, the state of the link undertakes slight and unnoticeable changes for a longer period of time and it may never resume to its original state. The slow degradation anomaly may commence triggering information loss and interruptions after a certain amount of time.

Symptom: Slow degradation anomaly could go unnoticed for a very long time, where users may not even notice any difference in service quality immediately. When relevant thresholds are triggered, users commence experiencing deteriorated service quality. After employed compensation methods are exhausted (e.g., buffers, queues, bandwidth preservation strategies), communication may be interrupted and intended services may become unavailable. From the perspective of the network, either transmitter gradually stops generating sufficient electromagnetic field to satisfy a received signal-to-noise ratio threshold or the receiver is not able detect or collect enough electromagnetic radiation to decode the information, which can also be induced by the aging of electronic components.

Possible causes: This type of degradation may be caused by easier aging of electronic components in extreme working conditions (e.g., high moisture and heat) as it is discussed in Section III and depicted in Fig. 1c where it reflects a gradual but permanent impairment to the hardware or, slowly increasing obstacle such as a building being slowly built or vegetation growing.

V. DATA REPRESENTATION

Sections III and IV provided real-world anomaly examples and formalized wireless link anomalies, respectively. In the following, we provide five distinct ways to represent data for a better understanding.

a) Time-value representation: The anomalies appearing in time series of RSSI values and in Figs. 1a and 2a are recorded as raw time-ordered values, thus forming a time series. We refer to this time-ordered values as time-value representation. In Figs. 3a, 4a, 5a and 6a, the time-value representation of an ordinary link is depicted with solid black lines and its anomaly injected counterpart, as per the definition from Section IV is depicted with dashed red lines.

However, through mathematical transformations, time series can be represented in other domains that, in some cases may be more suitable for the analysis of anomaly or pattern recognition. Motivated readers are referred to [35] for a comprehensive taxonomy of time series representation. In addition to the time-value representation, in this study, we also consider an aggregated representation, a histogram representation, a frequency domain representation and an automatically encoded representation.

b) Aggregated representation: This representation contains seven statistical aggregates computed from the time-valued representation, namely average, standard deviation, and all five quantile (Q) values, such as zeroth quantile (minimum), first quantile, second quantile (median), third quantile, and fourth quantile (maximum). This representation is depicted in Figs. 3b, 4b, 5b and 6b for each anomaly type, where they present values belonging to middle quantiles (Q1-Q3) as a box shape, first quantile (Q0-Q1) and third quantile (Q2-Q3) are marked as separate whiskers on top and the bottom, median value (Q2) is shown as a red bar within the box shape (–), and finally, average is portrayed as a blue triangle shape (▲).

c) Histogram representation: The histogram representation observed in Figs. 3c, 4c, 5c and 6c is performed via splitting the range between (global) minimum and maximum values into ten equally-sized bins. More explicitly, this representation exhibits the percentage of values allotted in each bin.

d) FFT representation: The frequency domain representation provided in Figs. 3d, 4d, 5d and 6d utilizes absolute value of complex transformation, which is presented using log-scale for better contrasting "with anomaly" scenario against the "no anomaly" one.

e) Encoded representation: A recent revolution of deep learning techniques, namely autoencoders, exhibits great performance returns in a diverse set of problems. To contrast against the above-mentioned traditional representations, we
propose automatically generated encoded (autoencoder) representations for all anomaly types introduced in Section IV.

Autoencoders [16], [36], [37] are neural networks which are trained to generate a representation from the reduced encoding that is very similar compared its original input. The middle layer of an autoencoder is depicted with the purple circles in Fig. 7 containing the reduced version of the input data and is referred to as a code \( h \) whose size is expected to be smaller than the size of the input data. As portrayed in Fig. 7 an autoencoder is composed of two parts; i) an encoder function \( h = f(x) \), and ii) a decoder function producing a reconstruction \( \hat{x} = g(h) \). The autoencoders thus learn to include only the most useful signals from the input data, while mitigating the unnecessary signal noise.

An undercomplete autoencoder, where code size is smaller than input size, with nonlinear activation functions presents a generalized form of principal component analysis (PCA). Through the training process, the error between input \( x \) and output \( \hat{x} \) becomes negligible. Consequently, neural network learns a new representation of the input data, within a re-
VI. APPROACHES FOR AUTOMATED DETECTION OF ANOMALIES

Considering the link anomalies defined in Section IV and their corresponding representations depicted in Figs. 3 and 6, it is clear that setting predefined thresholds for the investigated data would enable the detection of abnormal measurements and aid in treating them as an outlier. However, it has been proven that since fixed threshold-based approaches do not adapt to fluctuating behaviour of the data, selecting a threshold becomes consequential and thus may lead to poor performance, especially in real-time prediction applications [38]. On the contrary, adaptive and proactive approaches, such as deep learning neural network (DNN) and recurrent neural network (RNN) [38], can learn from regular patterns of the data and accurately identify abnormal behaviours to enable more accurate anomaly detection.

A. Threshold based detection

Considering Fig. 2a, detecting SuddenD requires the diagnosis of steep falling slopes that do not recover for a relatively long, possibly predefined, period of time. Detecting SuddenR amounts to the identification of a sudden drop and later a boost in signal that resumes back to the original strength level within a predefined time window. SuddenR and InstaD are somewhat similar from application perspective. However, the distinction lies in the length of the time window at which the signal recovers back to its original levels within an instant of the time for InstaD. Detecting SlowD requires the diagnosis of a slowly but rather consistently falling slope for a relatively long, possibly predefined time window.

The time-value rules are a straightforward way to approach link-level anomaly detection. These rules may either be set based on an experienced arbitrary threshold or they can be identified using a theoretical or numerical method. However, as discussed in Section VII, there are various possible ways to detect anomalies. For instance, it can be seen on Figs. 3b and 4b that RSS distribution of an average healthy link is significantly different than the RSS distribution of the same link when anomaly is injected, which is readily distinguishable for SuddenD, SuddenR and SlowD anomalies at a glance. More explicitly, the spread of RSS for the anomaly injected link is wider, and its mean and median values are overwritten accordingly. Similar conclusions can be made for the respective histograms in Figs. 3c, 4c and 6c. However, abnormal distributions in SlowD anomaly can only be detected with long-term observations. Moreover, sudden changes in time series can also be detected in frequency domain, which in our case, are readily observable for SuddenD and SuddenR anomalies as larger magnitudes at lower frequencies in Figs. 5b and 4b, respectively. Changes due to injected anomalies are almost indistinguishable in the case of InstaD and SlowD while leveraging frequency domain.

Details of the threshold strategy are provided in Section VIII. For time-value perspective, we consider D’Agostino-Pearson’s normality statistical test [39], [40]. The test assesses whether certain set of points come from normal distribution or not. If the p value is below threshold, it is likely that the measurements do not come from normal distribution. Notice that Pearson’s normality test is not sufficient condition for normality claims. Although, the approach may work fine for our limited line-of-sight scenario, it will not work for mobile or non line of sight scenario. For aggregated perspective, we consider for a link to have an anomaly two separate criteria. One criterion is based on the difference between mean and median values, which (if we assume normal distribution) are fairly close. The second criterion is how much can values deviate in standard deviation. Either of them has to be true for a link to be marked to have an anomaly. For histogram perspective, we define and arbitrary threshold. Anything below that is marked as an anomaly.

B. Machine learning-based detection

A ML model is expected to distinguish between anomalous and ordinary behaviours of a link, thus requires to solve a binary classification problem. There are two ways to train a ML model to identify such distinctions. The first one is based on a supervised training approach where all anomaly data are labelled, although in many practical applications, producing a reliable training dataset is expensive and it can inevitably cover only the type of anomalies that are present in the training dataset, which then cannot cope with the abnormal link behaviours in a comprehensive manner. For this reason, training a ML model in an unsupervised way is more practical, where learning from patterns of the overall link operations so as to distinguish the abnormal behaviours of a link from the

Figure 7. Illustration of autoencoder configuration during training process.
anticipated behaviours is provoked, which is referred to as the automated detection of an outlier \[41\] or an anomaly \[16\] using ML models.

In addition to baseline threshold-based approach discussed in Section VI-A, we also consider three supervised and three unsupervised ML techniques as elaborated in the following sections.

1) Supervised approaches: To evaluate the performance of selected supervised ML techniques against each other and against the threshold-based approach, we opt for a set of candidate supervised approaches leveraging one representative technique from three different classes: i) Logistic Regression from Regression Analysis \[42\], ii) Random Forest from tree ensemble class \[43\] and iii) Support Vector Machines (SVM) from kernel-method class \[43\].

Logistic Regression \[42\] is a modified linear regression able to work on classification problems. In linear regression the goal is to fit a line to data samples and minimize loss. Similarly, logistic regression aims for fitting sigmoid function with the goal to minimize loss at predicting any two classes. Logistic regression also includes a generalized form suitable for high-dimensional input data and multi-class rather than binary classification.

Random Forests \[44\] is an ensemble method that uses a number of decision tree classifiers followed by a voting mechanisms to perform multi-class classification. The trees are learnt by randomly splitting a relatively large feature space into smaller subspaces. Each tree provides a class in which a specific data point falls into, the class corresponds to the "vote" of that tree. The final outcome of the classifier then uses a mechanism, such as majority voting to provide the final result.

Support Vector Machine \[45\] is a learning algorithm that belongs to the family of kernel methods. Roughly speaking, SVMs attempt to learn a hyperplane that best splits a set of data into two classes. The shape of the hyperplane depends on the type of kernel function selected for the algorithm. When the kernel function is linear, so is the learnt hyperplane. When non-linear kernels are chosen, for instance RBF kernel \[46\], then the hyperplane is non-linear therefore better suited to approximate or discriminate non-linear random variables.

2) Unsupervised approaches: The cost of producing labels for supervised learning is discussed in Section VI-B. As a countermeasure, we also consider a set of candidate unsupervised approaches for developing anomaly detection models \[43\], where we leverage one representative technique from three different classes: i) Local Outlier Factor from Nearest Neighbour (NN) class \[43\], ii) Isolation Forest from tree ensemble class \[43\] and iii) one-class Support Vector Machines (SVM) from kernel-method class \[43\].

Local Outlier Factor \[47\] belongs to the k-Nearest Neighbour (kNN) family of algorithms, which rely on the computation of the distance between data points of the feature space. The feature vectors with smaller distance are alike and thus clustered together. One drawback for this family of algorithms is that as the dimensionality of the training data grows, the computational complexity evolves exponentially. However, there have been attempts in circumventing this exponential complexity, e.g., Ball Tree.

Isolation Forest \[48\] belongs to tree-based ensemble methods, and works in a roughly similar way as Random Forests as described above. Essentially, it represents a Random Forest adapted so that it optimizes outlier detection rather than multi-class classification of majority of data it sees. Based on certain metrics and distinct criteria, the algorithm decides whether particular subspaces contain any abnormal samples, namely anomalies.

Support Vector Machine, as described at the end of supervised approaches, can also be used in an unsupervised mode for anomaly detection. In fact, most ML techniques can be used in both supervised and unsupervised mode. With this one-class approach, the model is expected to distinguish data as negative or positive instances. Then, the model can learn the boundaries of the data so as to detect the points that lie outside the boundary exposed as anomalies or outliers.

VII. METHODOLOGY AND EXPERIMENTAL DETAILS

Before we proceed with the analysis of the relative performance of the wireless link anomaly detection approaches proposed in this paper, we provide relevant methodological and experimental details.

A. Training dataset generation

For our experimental evaluation, we consider a real-world measurement dataset, i.e., Rutgers \[49\], which contains measurements from 29 nodes at 5 different noise levels and each record has 300 measurements. Although every link is measured at five different noise levels, we consider each recording as a different link and we assume that there is no correlation.
On this existing real-world dataset we synthetically inject the four types of anomalies proposed in this paper as follows. First, we only pick the links without packet loss. This reduces our dataset from 4,060 to 2,123 (≈ 52%) of independent links. Second, by means of applying one anomaly type at a time, we randomly pick 33% of these links, at which the anomaly is injected according to guidelines in Table I while the remaining is left intact.

The suddenD anomaly, observed in Fig. 2a on the affected link appears arbitrarily between 200th and 280th packet and it persists indefinitely. In case of suddenR, observed in Fig. 2b anomaly applied on the link appears only once with a random start from 25th to 275th packet, where it persists for an arbitrary duration between 5 to 20 measurements. For InstaD of Fig. 2c the anomaly can appear anywhere in the entire series with 0.01 probability, which means that each anomaly on the affected link appears three times on average. Finally, SlowD anomaly of Fig. 2d appears arbitrarily between 1st and 20th measurements, where it commences with a random degrading pace of duration between 150 and 280 packets. In a nutshell, anomaly injection details are provided in Table I.

### Table I

| Type      | Links | Affected | Appearance | Persistence |
|-----------|-------|----------|------------|-------------|
| SuddenD   | 2,123 | 33% (700)| once, [200th, 280th] | for [5, 20] |
| SuddenR   | 2,123 | 33% (700)| once, [25th, 275th] | for [5, 20] |
| InstaD    | 2,123 | 33% (700)| on ≈ 1% of a link | for 1 datapoint |
| SlowD     | 2,123 | 33% (700)| once, [1st, 20th] | for [150, 180] |

1 RSSI(x, start) ← RSSI(x) + min(0, −rand(0.5, 1.5) · (x − start))

### B. Computing standard and encoded representations

Once anomalies are injected as specified in Table I we compute four different data representations described in Section VI-A.

The first one, namely time-value representation of Section VI-a, converts each link into a single feature vector containing 300 features. The second one, the so-called aggregated feature, summarizes each link with 7 features, which are described in Section VI-b. The third one, namely histogram feature discussed in Section VI-c, defines ten equally spaced bins, which are then presented to a model as a feature vector containing 10 features. The forth one, namely frequency feature elaborated in Section VI-d, gives the model a large feature vector of frequency-domain representation summing up to nearly 150 features. As we compute four representations for each of the four types of anomalies, we generate 16 candidate datasets.

Next, we also consider autoencoders for each anomaly scenario and each of the four standard representations. As any other deep neural network, autoencoder also requires many iterations of training. To produce credible results with autoencoder, we build the generic model in two steps. In the first step, we split the dataset into training and test groups with a 60:40 ratio, respectively. In the second step, when the weights of the autoencoder are converged, we perform an end-to-end evaluation on the test group. Relevant autoencoder configurations are provided in Table II where the layers and their required parameters are outlined for the encoder and the decoder. Although recent trends in DNNs go towards the use of convolutional layers, a convolution layer would make sense only in case of time-value and frequency perspective, due to their reasonable size and correlated neighbouring vector values. Therefore, our decision is to go with fully connected (dense) layers. For the activation part, we use batch normalization (BN) followed by Leaky Rectified Linear Unit (leaky ReLU, or LReLU) with $\alpha = 0.2$ coefficient for negative values. While plain ReLU is most widely used non-linear activation function, its leaky version has shown several benefits and minor overall improvements.

To produce the encoded representations, we feed the 16 datasets corresponding to the representation provided in Sections VI-(a),(b),(c),(d) into the autoencoder, resulting in additional 16 candidate datasets. Therefore, to continue with the anomaly detection, we train both supervised and unsupervised ML models on a total of 32 datasets, 16 corresponding to the four standard representations of each anomaly and the other 16 corresponding to the encoded representations.

### C. Performing automatic anomaly detection

Next, we compute the performance of the threshold, three supervised and three unsupervised ML techniques described in Section VI on the 32 generated datasets corresponding to the proposed anomalies and representations. Each approaches' output is compared to a label to identify whether the link actually contains anomalies or not.

1 a) Threshold approach: Descriptive details of leveraging certain thresholds for each anomaly can be found in Section VI-A. The utilized experimental threshold parameters are

### Table II

| Role | Layer | Notes |
|------|-------|-------|
| Input(*) | Dense(128) | |
| Encoder | Dense(64) | |
| BN + LeakyReLU($\sigma = 0.2$) | |
| BN + LeakyReLU($\sigma = 0.2$) | |
| BN + LeakyReLU($\sigma = 0.2$) | |
| BN + LeakyReLU($\sigma = 0.2$) | |
| Dense(4) | no activation | |
| Decoder | Dense(32) | |
| BN + LeakyReLU($\sigma = 0.2$) | |
| Dense(64) | |
| BN + LeakyReLU($\sigma = 0.2$) | |
| Dense(128) | |
| BN + LeakyReLU($\sigma = 0.2$) | |
| Dense(*) | no activation | |

* input/output size depends on feature vector

Implementation of autoencoders in TensorFlow/Keras is available at: [https://gist.github.com/gcerar/5e4e53902493632a3cfb5cc06c3317b7](https://gist.github.com/gcerar/5e4e53902493632a3cfb5cc06c3317b7)
Table III
PREDETERMINED ANOMALY THRESHOLDS.

| Features         | Anomaly thresholds                                      |
|------------------|---------------------------------------------------------|
| Time-series      | Normality test \cite{39, 40}, when \( p < 10^{-3} \)    |
| Aggregated       | (mean - median) > 3 dB OR (2-stdev > 2.5 dB)            |
| Histogram        | RSSI < -85 dBm                                          |

![Figure 9. Regularization parameter (C) search for selecting the best performing model that is, for example, trained using LR on time-value representation for SuddenD anomalies and based on robust scaler.](image)

listed in Table III. The threshold for the time-series representation that uses the D’Agostino-Pearson’s normality statistical test \cite{39, 40} is \( p < 10^{-3} \). The threshold for the aggregated representation assumes the absolute difference between mean and median is higher than 3dB or that the double of the standard deviation is higher than 2.5dB. The threshold for the histogram representation is set at RSSI < -85dBm while threshold selection for the FFT and encoded representations were infeasible to find using our trial-and-error approach. The differences in the FFT representation are not easily visible or detectable using simple methods while the encoded representations cannot be easily interpreted, therefore also deriving an appropriate threshold is not possible.

b) Machine learning-based approaches: For each of the six selected ML techniques, we use standard ML cross-validation\footnote{Stratified K-Fold cross validation is implemented by using StratifiedKFold parameter in Python Scikit Learn toolbox \url{https://scikit-learn.org/stable/}}. We train the models using shuffled data split into training and test sets with a 80:20 ratio, respectively. Model is trained with the training set and evaluated using the test set in order to ensure credible results. We use standard metrics for evaluating classifiers: precision, recall and F1 score. Precision measures how many of the instances detected as class A actually belong to class A, expressed as:

\[
\text{Precision} = \frac{TP}{TP+FP},
\]

whereas recall measures how many of the instances belonging to class A were actually detected, expressed as:

\[
\text{Recall} = \frac{TP}{TP+FN},
\]

where TP, FP and FN stand for true positives, false positives and false negatives, respectively. F1 score is quantified by the harmonic mean of the precision and the recall, where larger values indicate better classifiers with balanced and higher precision and recall performances.

For each of the ML techniques selected in Section VI, Table IV lists the respective implementations and parameters used in the experiments. For instance, for logistic regression we use the LogisticRegression implementation available in the Python Scikit Learn toolbox\footnote{Script for the design and development of anomaly detection models excluding data preprocessing is available at: \url{https://gist.github.com/gcerar/0b03e55f41147a7b7230f45d1f12094d}}. As the LogisticRegression implementation enables setting 12 different parameters that influence the final model, we generally select standard values that have been proven to work on large number of cases and datasets. However, we identify selected parameters that should be optimized, such as the regularization strength \( C \) in this case. We search for the best configuration by adapting an array of possible values \( C \in [10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2] \) and ultimately select the best performing regularization factor \( C \) among them. For instance, Fig. 9 presents the scenario where a model is trained using LR on time-value representation for SuddenD anomalies and based on robust scaler. For this particular scenario, the best F1 score of this model is attained by means of setting \( C \) to any value that is larger than 1. For the results presented in the next sections, we only account for the best F1 scores obtained after searching for such near-optimal regularization parameter values.

The implementations chosen for the remaining algorithms also include over ten possible input parameters. For LOF, we vary the number of neighbours, algorithm and leaf size for finding the best performing model. For RForest and IForest, we vary the number of base estimators, whereas for SVM and OC-SVM, we vary the regularization factor \( C \), the kernel and the kernel coefficient \( gamma \) for the \( rbf \) kernel, respectively.

As some of the models are sensitive to scaling, we also consider training on data that is: i) not scaled, ii) scaled by using mean values, iii) scaled using mean and deviation, and iv) scaled using min-max. The entire procedure and parameters can be readily found and used in the existing public open source repository\footnote{https://scikit-learn.org/stable/}. Six selected ML techniques with the associated parameter tuning are trained over the 32 datasets, totalling at more than 40,000 anomaly detection models.

VIII. Evaluation

In this section, we evaluate the relative performance of various data representations discussed in Section V and of approaches discussed in Section VI for detecting four types of anomalies introduced in Section IV. The methodological and experimental details utilized for obtaining the results are elaborated in Section VII.

A. Performance analyses of data representations

The best performing results with respect to the F1 score are presented in Table V for SuddenD and SuddenR, Table VI for InstaD and Table VII for SlowD. The results for SuddenD and SuddenR anomaly types are presented in one table as there are
The encoded representation discussed in Section \[V\] and extracted according to the specifications in Section \[VIII\] a are inserted as rows where the name of the ML technique includes the word “Encoder”. More precisely, when looking at the row corresponding to the ML technique, say IForest, the performance results refer to the four mentioned representations for the IForest ML technique. When looking at the row entitled “Encoder + IF”, the numerical results refer to the IForest ML technique that is applied to the codes generated from the four representations, respectively. Finally, the superscripts identify the scaling methods utilized. The three highest F1 scores for supervised approaches and the three highest F1 scores for unsupervised are emphasized with bold font.

It can be seen from the first two tables, Tables \[V\] and \[VI\] for the first three anomaly types, namely SuddenD, SuddenR and InstaD, that all have in common very steep changes in values that the encoded time-valued representations, followed by the encoded frequency domain representations, yield the best F1 scores. For SuddenD and SuddenR, the highest F1 scores are 0.77 to 0.74 for the supervised approaches and 0.63 to 0.60 for the unsupervised ones. For InstaD, the highest F1 scores are 0.76 to 0.74 for the supervised approaches and 0.74 to 0.64 for the unsupervised ones. All standard non-encoded representations as well as encoded representations computed from aggregated or histogram features yield significantly lower F1 scores of up to 0.44.

The results demonstrate that, for degradations that include steep changes, the autoencoder seems to be able to produce, from the raw data or its harmonic representation, an encoding that preserves the steep change and enables both supervised and unsupervised approaches to detect it with a relatively higher F1 score compared to its counterpart non-encoded representations. However, when the raw data is summarized or transformed into a significantly smaller dimensional representation, such as a 10 bin histogram or an 8 number of aggregated summary, useful information is lost and therefore the autoencoder cannot generate a code with sufficient signal for the learning methods.

From Table \[VII\] it can be observed that for the SlowD anomaly type that introduces very slow changes in the time-series, all representations, both standard and encoded, yield very high F1 scores with values above 0.95 for the supervised approach, while encoded features computed on time-series and histogram features seem to have an advantage over other standard and encoded representations with F1 scores from 0.98 to 0.85.

In a nutshell, autoencoders produce superior features compared to the non-encoded representations for all anomaly types. The results suggest that it is better to use collected data, without any pre-processing, such as summarization or dimensionality reduction (i.e., statistical aggregates or histograms) as an input to the autoencoder which conforms with the findings of anomaly detection on time-series data \[19, 51\]. The trade-off, between that the code compressing the data sufficiently well to be able to reconstruct it, and to preserve enough particularities in the compressed form so that a model is able to efficiently learn from, is crucial. Additionally, for the cases where a time-domain representation is not available, for instance when the hardware produces only a frequency domain representation, our proposed ML techniques using autoencoders will also perform well.

### B. Performance analyses of the approaches

It is demonstrated in Tables \[V\], \[VI\] and \[VII\] that the SlowD can be detected more accurately on our wireless anomalies dataset with very high F1 scores of up to 1.0, while other anomalies can be detected with F1 scores of up to 0.77. Unless stated otherwise, performance analysis are realized based on F1 scores.

**SuddenD and SuddenR anomalies:** According to Table \[V\] the supervised models are able to detect SuddenD and SuddenR anomalies more accurately than the unsupervised models, and most models yield higher performance using the
The best overall performance both on standard and encoded and SuddenR anomalies are 0.63 to 0.6 with SVM yielding the F1 scores of the unsupervised approaches on SuddenD and above the threshold on all four standard representations. SVM performs the best of the three representations and comparably on time-values and frequency domain representations. SVM performs well below the threshold. RForest performs better than LR on aggregated and histogram representations and comparably on time-values and frequency domain representations. SVM performs the best of the three and above the threshold on all four standard representations. The F1 scores of the unsupervised approaches on SuddenD and SuddenR anomalies are 0.63 to 0.6 with SVM yielding the best overall performance both on standard and encoded representations followed by RForest while LOF yields remarkable outcomes.

Further consideration of the precision and recall in Table V for the best performing models with the highest F1 scores for SuddenD and SuddenR anomalies, reveals that in general precision is up to 50% higher than recall. For instance, the supervised encoder+SVM model on time-valued features has a precision of 0.88 and a recall of 0.69. This means that 88% of the anomalous links from the test data have been detected as anomalous while they are actually ordinary links. In other words, the ratio of true positive detection of that model is quite high. However, the lower recall means that only 69% of the anomalous links from the test data have been detected as anomalous while they are actually ordinary links. The ratio of links detected as anomalous, while the remaining is classified as ordinary links. The ratio of links detected as true positive to the overall positives existing in the test data, represented by the links detected as anomalous are actually truly anomalous according to our definition and only the remaining 12% have been detected as anomalous while they are actually ordinary links. In other words, the ratio of true positive detection of that model is quite high. However, the lower recall means that only 69% of the anomalous links from the test data have been detected as anomalous while they are actually ordinary links. The ratio of links detected as anomalous, while the remaining is classified as ordinary links. The ratio of links detected as true positive to the overall positives existing in the test data, represented by the

| Approach | Technique | time-value features | aggregated features | histogram features | frequency domain |
|----------|-----------|---------------------|---------------------|-------------------|-----------------|
|          |           | Prec. | Rec. | F1 | Prec. | Rec. | F1 | Prec. | Rec. | F1 | Prec. | Rec. | F1 |
| Baseline | Threshold (Tab. III) | 0.27 | 0.19 | 0.22 | 0.25 | 0.01 | 0.02 | 0.27 | 0.48 | 0.35 | - | - | - |
| Supervised | LR | 0.28 | 0.18 | 0.22 | 1.00 | 0.00 | 0.00 | 0.80 | 0.01 | 0.01 | - | - | - |
| | encoder + LR | 0.81 | 0.73 | 0.77 | 0.67 | 0.00 | 0.01 | 1.00 | 0.00 | 0.01 | - | - | - |
| | RForest | 0.38 | 0.12 | 0.18 | 0.27 | 0.14 | 0.18 | 0.28 | 0.20 | 0.24 | - | - | - |
| | encoder + RForest | 0.82 | 0.71 | 0.76 | 0.28 | 0.19 | 0.23 | 0.36 | 0.21 | 0.27 | - | - | - |
| | SVM | 0.29 | 0.56 | 0.39 | 0.30 | 0.64 | 0.41 | 0.30 | 0.61 | 0.40 | - | - | - |
| | encoder + SVM | 0.88 | 0.69 | 0.77 | 0.29 | 0.75 | 0.42 | 0.29 | 0.78 | 0.42 | - | - | - |
| | LOF | 0.29 | 0.67 | 0.41 | 0.31 | 0.28 | 0.30 | 0.33 | 0.24 | 0.27 | - | - | - |
| | encoder + LOF | 0.44 | 0.08 | 0.14 | 0.32 | 0.22 | 0.26 | 0.31 | 0.17 | 0.22 | - | - | - |
| | IForest | 0.29 | 0.15 | 0.20 | 0.29 | 0.30 | 0.30 | 0.28 | 0.47 | 0.35 | - | - | - |
| | encoder + IForest | 0.81 | 0.49 | 0.61 | 0.26 | 0.32 | 0.28 | 0.29 | 0.47 | 0.36 | - | - | - |
| | OC-SVM | 0.28 | 1.00 | 0.44 | 0.28 | 0.99 | 0.44 | 0.28 | 0.96 | 0.44 | - | - | - |
| | encoder + OC-SVM | 0.62 | 0.65 | 0.63 | 0.28 | 1.00 | 0.44 | 0.28 | 1.00 | 0.44 | - | - | - |

Table V

Performance of detecting sudden degradation (SuddenD) and sudden degradation with recovery (SuddenR) anomalies.

| Approach | Technique | time-value features | aggregated features | histogram features | frequency domain |
|----------|-----------|---------------------|---------------------|-------------------|-----------------|
|          |           | Prec. | Rec. | F1 | Prec. | Rec. | F1 | Prec. | Rec. | F1 | Prec. | Rec. | F1 |
| Baseline | Threshold (Tab. III) | 0.37 | 0.32 | 0.35 | 0.28 | 0.01 | 0.03 | 0.27 | 0.51 | 0.36 | - | - | - |
| Supervised | LR | 0.29 | 0.18 | 0.22 | 0.69 | 0.27 | 0.39 | 0.80 | 0.01 | 0.01 | - | - | - |
| | encoder + LR | 0.81 | 0.68 | 0.74 | 0.72 | 0.28 | 0.40 | 0.75 | 0.01 | 0.01 | - | - | - |
| | RForest | 0.29 | 0.10 | 0.15 | 0.53 | 0.36 | 0.43 | 0.27 | 0.20 | 0.23 | - | - | - |
| | encoder + RForest | 0.80 | 0.69 | 0.74 | 0.53 | 0.37 | 0.44 | 0.28 | 0.20 | 0.24 | - | - | - |
| | SVM | 0.28 | 0.46 | 0.34 | 0.32 | 0.72 | 0.44 | 0.26 | 0.81 | 0.39 | - | - | - |
| | encoder + SVM | 0.86 | 0.69 | 0.76 | 0.63 | 0.33 | 0.44 | 0.26 | 0.73 | 0.38 | - | - | - |
| | LOF | 0.28 | 0.67 | 0.40 | 0.33 | 0.32 | 0.33 | 0.32 | 0.23 | 0.27 | - | - | - |
| | encoder + LOF | 0.35 | 0.11 | 0.17 | 0.27 | 0.22 | 0.24 | 0.31 | 0.18 | 0.23 | - | - | - |
| | IForest | 0.25 | 0.15 | 0.19 | 0.31 | 0.27 | 0.29 | 0.28 | 0.33 | 0.30 | - | - | - |
| | encoder + IForest | 0.84 | 0.67 | 0.74 | 0.55 | 0.34 | 0.42 | 0.26 | 0.52 | 0.35 | - | - | - |
| | OC-SVM | 0.27 | 1.00 | 0.43 | 0.27 | 0.98 | 0.42 | 0.27 | 1.00 | 0.43 | - | - | - |
| | encoder + OC-SVM | 0.67 | 0.73 | 0.70 | 0.31 | 0.76 | 0.45 | 0.27 | 1.00 | 0.43 | - | - | - |
recall, is lower than the precision by 19 percentage points for the supervised encoder+ SVM model on time-valued features. For the case of unsupervised encoder+IForest model on time-valued features, the difference between precision and recall is 32 percentage points meaning that most of the detected anomalous links are actually anomalous, albeit the detected anomalous links over the total anomalous links are only 40%.

On the other hand, Table VII presents that for the case of unsupervised encoder+IForest model on time-valued features, the precision and recall performances exhibit a tiny margin with a precision of 0.62 and a recall of 0.65.

Having faced with the choice of selecting one of the two unsupervised models, one would need to decide what is more important for the application: i) how precisely the model detects anomalous links (high precision) or ii) how many relevant anomalous links were detected (high recall). For a precise model, one would select unsupervised encoder+ SVM model on time-valued features, while for a well-balanced detection performance, opting for unsupervised encoder+IForest model on time-valued features would be plausible.

**InstaD anomalies:** According to Table VII the supervised and unsupervised models exhibit comparable performance in detecting InstaD anomalies, and most models yield higher performance using the encoded representations. The best models, i.e., LR, RF, SVM, IForest, and OC-SVM have comparable F1 scores on encoded time-value and FFT representations that are well above the threshold approach. All models trained for InstaD anomaly show relatively modest performance on encoded aggregated and encoded histogram representations as well as on the four standard representations.

Considering precision and recall in Table VII for the best performing models with highest F1 scores for InstaD anomalies show that the precision is up to 25% higher than recall score. First important notice is that all the best performing models (based on F1 score) are trained either on time-value or FFT representations, while in general aggregated and histogram representations perform poorly for InstaD anomalies. For instance, in general, OC-SVM and encoder+OC-SVM models on aggregated and histogram representations exhibit similar performances. Those four models perform a precision of around 28% and mostly a recall of about 100%. This indicates that models were able to detect almost all anomalous links, however, low precision points that the models have high false positive rate of detection. More explicitly, while models were able to find almost all anomalous links, out of all the marked samples as anomalous, labels were incorrectly placed in 72% of the cases.

If the application requires a selection of one of the two unsupervised models, one would select either unsupervised encoder+IForest or encoder+OC-SVM models on time-valued features for a better precision performance considering that they both perform an F1 score within a tiny margin of 4 percentage points.

**SlowD anomalies:** According to Table VII all three supervised models are able to detect SlowD anomalies with very high F1 scores of 1.0 to 0.98. While LR and SVM attain this performance only on the encoded representations, RF based models are also able to achieve similar performances with the standard representations. The best performing unsupervised ML model is OC-SVM on histogram features.

### Table VII

**PERFORMANCE OF DETECTING SLOW DEGRADATION (SLOWD) ANOMALIES.**

| Approach | Technique | time-value features | aggregated features | histogram features | frequency domain |
|----------|-----------|---------------------|---------------------|-------------------|------------------|
|          |           | Prec. | Rec. | F1  | Prec. | Rec. | F1  | Prec. | Rec. | F1  | Prec. | Rec. | F1  |
| Baseline | Threshold (Tab. III) | 0.17  | 0.10 | 0.13 | 0.49  | 0.03 | 0.06 | 0.31  | 0.57 | 0.40 | -    | -    | -   |
| Supervised | LR | 0.97  | 0.94 | 0.96 | 0.52  | 0.22 | 0.31 | 0.97  | 0.99 | 0.98 | 0.92  | 0.91 | 0.92 |
|          | encoder + LR | 0.98  | 0.98 | 0.99 | 1.00  | 0.00 | 0.00 | 1.00  | 1.00 | 0.00 | 0.96  | 0.97 | 0.97 |
|          | RForest | 0.99  | 0.97 | 0.98 | 0.98  | 0.99 | 0.99 | 1.00  | 0.00 | 0.99 | 0.92  | 0.94 | 0.93 |
|          | encoder + RForest | 0.98  | 0.98 | 0.99 | 1.00  | 0.00 | 0.00 | 1.00  | 1.00 | 0.00 | 0.98  | 0.94 | 0.96 |
|          | SVM | 0.98  | 0.94 | 0.96 | 0.96  | 0.98 | 0.98 | 0.99  | 1.00 | 0.00 | 0.93  | 0.95 | 0.94 |
|          | encoder + SVM | 0.98  | 0.98 | 0.99 | 1.00  | 0.00 | 0.00 | 1.00  | 1.00 | 0.00 | 0.96  | 0.97 | 0.97 |
| Unsupervised | LOF | 0.30  | 0.44 | 0.36 | 0.31  | 0.23 | 0.26 | 0.33  | 0.24 | 0.28 | 0.62  | 0.03 | 0.05 |
|          | encoder + LOF | 0.34  | 0.09 | 0.14 | 0.41  | 0.15 | 0.22 | 0.53  | 0.08 | 0.14 | 0.54  | 0.08 | 0.13 |
|          | IForest | 0.34  | 0.20 | 0.25 | 0.35  | 0.28 | 0.31 | 0.40  | 0.15 | 0.21 | 0.33  | 0.20 | 0.25 |
|          | encoder + IForest | 0.83  | 0.91 | 0.87 | 0.19  | 0.18 | 0.18 | 0.46  | 0.36 | 0.41 | 0.48  | 0.44 | 0.46 |
|          | OC-SVM | 0.31  | 0.98 | 0.47 | 0.30  | 0.96 | 0.46 | 0.97  | 0.99 | 0.98 | 0.36  | 0.88 | 0.51 |
|          | encoder + OC-SVM | 0.56  | 0.99 | 0.72 | 0.72  | 0.77 | 0.74 | 0.75  | 0.99 | 0.85 | 0.49  | 0.88 | 0.63 |

1. No-scaling
2. Only mean scaling (by standard scaler)
3. Mean and deviation scaling (by standard scaler)
4. Only mean scaling (by robust scaler with respect to values between Q25 and Q75)
5. Mean and deviation scaling (by robust scaler with respect to values between Q25 and Q75)
6. Min-Max scaler
7. Linear kernel
8. RBF kernel
shown in Fig. 10. The model was able to accurately detect which was trained on synthetic SuddenD anomaly type, as various technologies and possibly for distinct applications. 802.11 traces is not directly transferable, which indicates that both trace-sets are limited in size, the learnt model on IEEE data from LOG-a-TEC contains IEEE 802.15.4 traces, where included 11 links and was not sufficient to train automatic the utilized dataset as a motivating example in Section III only trace, as discussed in Section VIIa to support our claims, since we synthetically injected anomalies in an available real-world data [26] and none of them uses link layer traces. In this study, of our knowledge, only few related works consider real-world While synthetic data is useful to develop a proof of concept, underlying distribution. Intuition would say that the data that is training examples to be able to accurately approximate the but in general it means that the model needs to see enough application data such as measurements.

First of all, every ML-based tool needs sufficient data for training and evaluation. Quantifying "sufficient" is difficult but in general it means that the model needs to see enough training examples to be able to accurately approximate the underlying distribution. Intuition would say that the data that is "sufficient" to learn a normal distribution would be smaller in size than the data needed to learn an exponential distribution. While synthetic data is useful to develop a proof of concept, for anything more than that real data is required. To the best of our knowledge, only few related works consider real-world data [26] and none of them uses link layer traces. In this study, we synthetically injected anomalies in an available real-world trace, as discussed in Section VIIa to support our claims, since the utilized dataset as a motivating example in Section II only included 11 links and was not sufficient to train automatic feature extractors and classification models. As the models were developed on IEEE 802.11 traces and the motivation data from LOG-a-TEC contains IEEE 802.15.4 traces, where both trace-sets are limited in size, the learnt model on IEEE 802.11 traces is not directly transferable, which indicates that the developed models cannot be readily generalized across various technologies and possibly for distinct applications.

We directly tested our model on the LOG-a-TEC traces, which was trained on synthetic SuddenD anomaly type, as shown in Fig. 10. The model was able to accurately detect two of the anomalous links shown in Figs. 10c and 10h, albeit its accuracy in detecting anomalies for the other four anomalous links, shown in Figs. 10a, 10b, 10f, and 10k was not adequate. Finally, our model falsely detected links as anomalous, as illustrated in Figs. 10e and 10h. Although at a glance they seem to be non-anomalous, they may be misclassified as anomalous by the model.

Secondly, the architecture of the autoencoder that learns the encoded features has been selected for a small number of candidates as a result of the trial-and-error method. Having more data would enable training an autoencoder, which then can be better generalized for even unseen examples. Autoencoder optimization and end-to-end deep learning for the proposed anomaly types might bring further insights into developing better performing and more reliable anomaly detection. However, as hyperparameter search in deep learning is challenging and needs a large amount of training data, we leave such optimization for future work.

Thirdly, in this study we only developed offline models that would need to be periodically retrained in real-world applications in order to account for the dynamically changing environment which is an inherent characteristic of the wireless networks. This leads us to online models that can learn from continuous incoming (streaming) data. Roughly speaking, offline models outperform online counterparts models at the expense of computational power, albeit online models are able to automatically adapt in a faster manner, and also simplify the detection system owing to its reduced storage requirements.

C. Limitations

We identify three main limitations that apply to this treatise, and to the best of our understanding also to other related works in wireless network and IoT anomaly data that do not target application data such as measurements.

First of all, every ML-based tool needs sufficient data for training and evaluation. Quantifying "sufficient" is difficult but in general it means that the model needs to see enough training examples to be able to accurately approximate the underlying distribution. Intuition would say that the data that is "sufficient" to learn a normal distribution would be smaller in size than the data needed to learn an exponential distribution. While synthetic data is useful to develop a proof of concept, for anything more than that real data is required. To the best of our knowledge, only few related works consider real-world data [26] and none of them uses link layer traces. In this study, we synthetically injected anomalies in an available real-world trace, as discussed in Section VIIa to support our claims, since the utilized dataset as a motivating example in Section II only included 11 links and was not sufficient to train automatic feature extractors and classification models. As the models were developed on IEEE 802.11 traces and the motivation data from LOG-a-TEC contains IEEE 802.15.4 traces, where both trace-sets are limited in size, the learnt model on IEEE 802.11 traces is not directly transferable, which indicates that the developed models cannot be readily generalized across various technologies and possibly for distinct applications.

We directly tested our model on the LOG-a-TEC traces, which was trained on synthetic SuddenD anomaly type, as shown in Fig. 10. The model was able to accurately detect two of the anomalous links shown in Figs. 10c and 10h, albeit its accuracy in detecting anomalies for the other four anomalous links, shown in Figs. 10a, 10b, 10f, and 10k was not adequate. Finally, our model falsely detected links as anomalous, as illustrated in Figs. 10e and 10h. Although at a glance they seem to be non-anomalous, they may be misclassified as anomalous by the model.

Secondly, the architecture of the autoencoder that learns the encoded features has been selected for a small number of candidates as a result of the trial-and-error method. Having more data would enable training an autoencoder, which then can be better generalized for even unseen examples. Autoencoder optimization and end-to-end deep learning for the proposed anomaly types might bring further insights into developing better performing and more reliable anomaly detection. However, as hyperparameter search in deep learning is challenging and needs a large amount of training data, we leave such optimization for future work.

Thirdly, in this study we only developed offline models that would need to be periodically retrained in real-world applications in order to account for the dynamically changing environment which is an inherent characteristic of the wireless networks. This leads us to online models that can learn from continuous incoming (streaming) data. Roughly speaking, offline models outperform online counterparts models at the expense of computational power, albeit online models are able to automatically adapt in a faster manner, and also simplify the detection system owing to its reduced storage requirements.

IX. Conclusions

In this paper, we introduce four types of anomalies that can be present in wireless links and are useful for being detected in real-world operational IoT deployments. We demonstrated that these anomalies were exposed on a real-world IoT deployment, namely the LOG-a-TEC testbed, and they significantly affected the expected operations of the testbed. Motivated by this, we develop detection models for each type of anomaly by considering five different data representations and six different ML techniques. We performed an extensive relative evaluation of the models from data representations and ML models perspective, and the limitations of our models are discussed. The resulting tool-set for anomaly injection, feature generation and model development are made publicly available for reproducibility.

We show that the proposed data representations generated by automatic feature learning with autoencoders outperform other standard representations for all four anomaly types namely, sudden degradation (SuddenD), sudden degradation with recovery (SuddenR), instantaneous (InstaD) and slow degradation (SlowD) anomalies. Next, we demonstrate that automatically generated features with autoencoder exhibit the most significant improvement, when we encode time-value and FFT representations. However, encoding aggregated works well only with the case of InstaD anomaly, and it performs poor performance on histogram representation. Additionally, for one specific technique, namely Local Outlier Factor (LOF), encoded features are not conformable, as it significantly degrades the classification performance in all scenarios. For every other ML model, let it be supervised or unsupervised,
Figure 10. Anomaly detection test employed over real-world measurements gleaned from the LOG-a-TEC testbed, where for example, as in (g) N16 → N17 indicates a communication link between nodes 16 and 17.
i.e., Logistic Regression (LR), SVM, Random Forest, 1-class SVM and Isolation Forest, encoded features show significant improvement for all observed metrics, namely precision, recall and F1 score.

We also show that considering all features and the best performing models based on F1 score, supervised ML models, in general, outperform their unsupervised counterpart models, which is, on average, about 18% better on SuddenD and SuddenR (e.g., encoder+ SVM 77% vs. encoder+OC-SVM 63% with time-value features), and 2% better on SlowD (e.g., SVM 98% vs. OC-SVM 100% with histogram features) and about 2.6% better on InstaD (e.g., encoder+ SVM 76% vs. encoder+IForest 74% with time-value features). Furthermore, our analyses have shown that the improvement with automatically generated encoded features can improve F1 score up to 500%, which was observed on encoded time-value representation of supervised Random Forest with InstaD anomaly and of unsupervised Isolation Forest with SlowD anomaly. We also discuss model selection from the perspective of precision and recall emphasizing tradeoffs. For instance, when the application requires a precise detection model, then given the same F1 score, one should choose the model with a higher precision. If the application requires more relevant anomalous link detection, given the same F1 score, one should opt for the model with a higher recall.

REFERENCES

[1] T. Qin, N. Chen, K. Li, M. Ariquzzaman, and W. Zhao, “How can heterogeneous internet of things build our future: A survey,” IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 3011–2027, 2018.

[2] J. Davies and C. Fortuna, The Internet of Things: From Data to Insight. John Wiley & Sons, 2020.

[3] R. Díaz-Díaz, L. Muñoz, and D. Pérez-González, “Business model analysis of public services operating in the smart city ecosystem: The case of smartSantander,” Future Generation Computer Systems, vol. 76, pp. 198–214, 2017.

[4] U. Wetzker, I. Splits, M. Zimmerling, C. A. Boano, and K. Römer, “Troubleshooting wireless coexistence problems in the industrial internet of things,” in 2016 IEEE Int'l Conference on Computational Science and Engineering (CSE), Paris, France, August 2016.

[5] J. D. C. Silva, J. I. P. Rodrigues, K. Saleem, S. A. Kozlov, and R. A. Rabelo, “M4DN: IoT-A Networks and Devices Management Platform for Internet of Things,” IEEE Access, vol. 7, pp. 53305–53313, April 2019.

[6] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker, “Mojo: A distributed physical layer anomaly detection system for 802.11 wlan,” in Proceedings of the 4th international conference on Mobile systems, applications and services. ACM, 2006, pp. 191–204.

[7] S. Gupta, R. Zheng, and A. M. Cheng, “Andes: an anomaly detection system for wireless sensor networks,” in 2007 IEEE International Conference on Mobile Adhoc and Sensor Systems, 2007, pp. 1–9.

[8] H. Alipour, Y. B. Al-Nashif, P. Satam, and S. Hariri, “Wireless anomaly detection based on ieee 802.11 behavior analysis,” IEEE Transactions on Information Forensics and Security, vol. 10, no. 10, pp. 2158–2170, May 2015.

[9] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM computing surveys (CSUR), vol. 41, no. 3, July 2009.

[10] M. Vucnik, T. Solc, U. Gregorc, A. Hrovat, K. Bregar, M. Smolnikar, M. Mohorcic, and C. Fortuna, “Continuous integration in wireless technology development,” IEEE Communications Magazine, vol. 56, no. 12, pp. 74–81, 2018.

[11] P. Gogoi, D. Bhattacharyya, B. Borah, and J. K. Kalita, “A survey of outlier detection methods in network anomaly identification,” The Computer Journal, vol. 54, no. 4, pp. 570–588, 2011.

[12] A. Zimek, E. Schubert, and H.-P. Kriegel, “A survey on unsupervised outlier detection in high-dimensional numerical data,” Statistical Analysis and Data Mining: The ASA Data Science Journal, vol. 5, no. 5, pp. 363–387, 2012.

[13] C. C. Aggarwal, “Outlier ensembles: position papers,” ACM SIGKDD Explorations Newsletter, vol. 14, no. 2, pp. 58–59, April 2013.

[14] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, “Outlier detection for temporal data: A survey,” IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 9, pp. 2250–2267, September 2014.

[15] X. Xu, H. Liu, and M. Yao, “Recent progress of anomaly detection,” Hindawi Complexity, vol. 2019, January 2019.

[16] A. Cook, G. Mursuli, and S. Fan, “Anomaly detection for IoT time-series data: A survey,” IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6481–6494, July 2020.

[17] A. Lavin and S. Ahmad, “Evaluating real-time anomaly detection algorithms – the numaenta anomaly benchmark,” in 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), Dec 2015, pp. 38–44.

[18] R. Jurad, X. R. Wang, O. Obst, and P. Valencia, Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 309–325. [Online]. Available: https://doi.org/10.1007/978-3-642-17931-0_12

[19] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, “Outlier detection for time series with recurrent autoencoder ensembles,” in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), Macao, PR. China, 2019, pp. 2725–2732.

[20] T. J. O’Shea, J. Corgan, and T. C. Clancy, “Unsupervised representation learning of structured radio communication signals,” in 2015 First International Workshop on Sensing, Processing and Learning for Intelligent Machines (SPLINE). IEEE, 2016, pp. 1–5.

[21] T. J. O’Shea, T. Erpek, and T. C. Clancy, “Deep learning based mimo communications,” arXiv preprint arXiv:1707.07980, 2017.

[22] H. Zhang, K. Liu, Q. Shang, L. Feng, C. Chen, Z. Wu, and S. Guo, “Dual-band Wi-Fi based indoor localization via stacked denoising autoencoder,” in IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, December 2019.

[23] C. Wang, F. Hu, Y. Zhao, and T. N. Guo, “Anomaly detection and anomaly diagnosis in wireless networks with multiple antennas: Framework, challenges and tools,” IEEE Network, vol. 32, no. 1, pp. 152–159, 2018.

[24] M. R. Shahid, G. Blanco, Z. Zhang, and H. Debar, “Anomalous communications detection in IoT networks using sparse autoencoders,” in IEEE 18th International Symposium on Network Computing and Applications (NCA), Cambridge, United States, September 2019.

[25] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, “Autoencoder-based network anomaly detection,” in Wireless Telecommunications Symposium (WTS), Phoenix, AZ, USA, April 2018.

[26] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, “Anomaly detection based on convolutional recurrent autoencoder for IoT time series,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020.

[27] V. L. Thing, “IEEE 802.11 network anomaly detection and attack classification: A deep learning approach,” in IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, March 2017.

[28] J. Ran, Y. Ji, and B. Tang, “A semi-supervised learning approach to ieee 802.11 network anomaly detection,” in 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring). IEEE, 2019, pp. 1–5.

[29] O. Salem, A. Guerra, O. M. Mehaoua, A. Marcus, and B. Furht, “Anomaly detection in medical wireless sensor networks using svm and linear regression models,” International Journal of E-Health and Medical Communications (IJHEMC), vol. 5, no. 1, pp. 20–45, 2014.

[30] ——, “Sensor fault and patient anomaly detection and classification in medical wireless sensor networks,” in IEEE international conference on communications (ICC), Budapest, Hungary, June 2013, pp. 4373–4378.

[31] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, “Machine learning in wireless sensor networks: Algorithms, strategies, and applications,” IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 1996–2032, Fourthquarter 2014.

[32] T. Solc, C. Fortuna, and M. Mohorcic, “Low-cost testbed development and its applications in cognitive radio prototyping,” in Cognitive radio and networking for heterogeneous wireless networks. Springer, 2015, pp. 361–405.

[33] T. Solc and Z. Padrah, “Network design for the LOG-a-TEC outdoor testbed,” in The 2nd International Workshop on Measurement-based Experimental Research, Experimentation and Tools, 2013.

[34] J. K. Mann, S. Perinpanayagam, and I. Jennions, “Aging detection capability for switch-mode power converters,” IEEE Transactions on Industrial Electronics, vol. 63, no. 5, pp. 3216–3227, 2016.
[35] J. Lin, S. Williamson, K. Borne, and D. DeBarr, “Pattern recognition in time series,” Advances in Machine Learning and Data Mining for Astronomy, vol. 1, no. 617-645, p. 3, 2012.

[36] M. A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” AIChE Journal, vol. 37, no. 2, pp. 233–243, February 1991.

[37] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT press, 2016.

[38] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, “Improving traffic forecasting for 5g core network scalability: A machine learning approach,” IEEE Network, vol. 32, no. 6, pp. 42–49, 2018.

[39] R. B. d’Agostino, “An omnibus test of normality for moderate and large size samples,” Biometrika, vol. 58, no. 2, pp. 341–348, 1971.

[40] R. D’AGOSTINO and E. S. Pearson, “Tests for departure from normality, empirical results for the distributions of b 2 and b,” Biometrika, vol. 60, no. 3, pp. 613–622, 1973.

[41] V. Hodge and J. Austin, “A survey of outlier detection methodologies,” Artificial intelligence review, vol. 22, no. 2, pp. 85–126, 2004.

[42] R. Malouf, “A comparison of algorithms for maximum entropy parameter estimation,” in Proceedings of the 6th Conference on Natural Language Learning - Volume 20, ser. COLING-02. USA: Association for Computational Linguistics, 2002, p. 1–7. [Online]. Available: https://doi.org/10.3115/1118853.1118871

[43] C. C. Aggarwal, “Outlier analysis,” in Data mining. Springer, 2015, pp. 237–263.

[44] L. Breiman, “Random forests,” Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[45] V. Vapnik, The nature of statistical learning theory. Springer science & business media, 2013.

[46] L. L. Z. Xiao-long, “Optimization of SVM with RBF Kernel [J],” Computer Engineering and Applications, vol. 29, 2006.

[47] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: Identifying density-based local outliers,” SIGMOD Rec., vol. 29, no. 2, p. 93–104, May 2000. [Online]. Available: https://doi.org/10.1145/335191.335388

[48] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in 2008 Eighth IEEE International Conference on Data Mining. IEEE, 2008, pp. 413–422.

[49] S. K. Kaul, I. Seskar, and M. Gruteser, “CRAWDAD dataset rutgers/noise (v. 2007-04-20),” Downloaded from https://crawdad.org/rutgers/noise/20070420/RSSI, Apr. 2007, traceset: RSSI.

[50] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in Proc. icml, vol. 30, no. 1, 2013, p. 3.

[51] K. Chen, J. Hu, and J. He, “A framework for automatically extracting overvoltage features based on sparse autoencoder,” IEEE Transactions on Smart Grid, vol. 9, no. 2, pp. 594–604, 2016.