Supervised and Semi-Supervised Learning for Failure Identification in Microwave Networks

Francesco Musumeci, *Member, IEEE,* Luca Magni, Omran Ayoub, Roberto Rubino, Massimiliano Capacchione, Gabriele Rigamonti, Michele Milano, Claudio Passera, and Massimo Tornatore, *Senior Member, IEEE*

***Abstract*—Automated failure-cause identification in communication networks allows operators to reduce service unavailability. Once the most likely failure root-cause is identified, appropriate countermeasures can be effectively put in place (e.g., by choosing an in-field intervention vs. a remote equipment reconfiguration). In this paper, we describe a successful application of Machine Learning (ML) for automatic failure identification in microwave networks based on the real-field data. On microwave links, different heterogeneous causes (e.g., adverse atmospheric conditions, or obstacles) lead to service unavailability and produce not easily-distinguishable degradation effects on the transmission parameters. Hence, failure identification is traditionally accomplished by domain experts via direct inspection of transmission-parameter logs. As a first contribution, we identify six categories of failure causes in microwave networks and show that supervised ML enables very accurate failure identification, hence significantly simplifying failure troubleshooting. Comparing various ML algorithms, we find that up to 93% classification accuracy is obtained using real-field labeled datasets with 2513 points. One main hindrance to the application of supervised learning is that, in real network deployments, limited amount of labeled data is available for training, as manual labeling is performed by domain experts based on their knowledge and experience. On the other hand, collecting unlabeled data is relatively simple as network management systems retrieve large amounts of unlabeled information automatically. As a second contribution, we investigate an automated labeling procedure, based on autoencoders-like Artificial Neural Networks, to combine the knowledge of the few manually-labeled data with large unlabeled data. Results show that our data augmentation based on autoencoders can slightly improve failure-cause identification only when Artificial Neural Networks or Support Vector Machines are used, while accuracy slightly decreases when adopting Random Forest.**

***Index Terms*—Microwave Networks, Failure Identification, Root-Cause Analysis, Machine Learning, Data Augmentation, Semi-Supervised Learning.**

I. Introduction

Machine Learning (ML) can be effectively applied for automation of failure management in communication networks. ML potential has been already demonstrated in optical networks [1], [2], however, little attention has been so far devoted to application in the context of microwave radio networks. Microwave networks represent a promising field as free-space signal transmission is substantially affected by unpredictable variations of the radio channel state, which can occur over time and with different severity levels, resulting in radio link unavailability event whose causes are not easy to discriminate. In this work, we concentrate on the identification of failure causes (a.k.a. root-cause analysis) in microwave networks using supervised and unsupervised ML learning methodologies [3].

In general, in microwave networks, different failure causes trigger different countermeasures from the network operator. For example, common failures causes in microwave links can be 1) a sudden increase in channel attenuation (e.g., due to atmospheric events or presence of physical obstacles), typically causing only temporary link unavailability (no repair action is triggered in this case), 2) misconfiguration/limited damage of radio equipment, which leads to permanent link degradation link (a reparation is required in this case), or 3) equipment hardware failure which leads to severe degradation or total link disruption (which requires in-field intervention for substitution). Nowadays, failure-cause identification is carried out by human experts, who "manually" analyse and correlate the behaviour of various link measures (e.g., transmitted/received power values, modulation format, equipment alarms, etc.) and, based on their experience, identify possible failure causes and devise proper countermeasures to restore the service.

As this manual analysis is very time-consuming, it is nowadays performed by domain experts in an *offline* manner (e.g., once per day). To the best of our knowledge, nowadays no automated failure-cause identification solution/models for microwave networks exist. Therefore, ML is regarded as a promising technique to automate failure-cause identification and enable automated and fast root-cause analysis. The basic underlying idea for ML application to failure identification is that different causes produce different "*signatures*" on the link transmission parameters, and that ML models can learn this *signature* by observing historical data, and hence become able to identify failure causes in future similar situations. To this end, in this paper we design different ML algorithms to perform automatic failure-cause identification in microwave links by exploiting observation of link transmission parameters at both sides of the link. The various ML algorithms are then evaluated and compared in terms of classification accuracy and training duration.

However, such approaches rely on the availability of significant amounts of labeled data, i.e., they need to be trained using several observations coming from microwave links (along with their *signature*) affected by a failure and for which a root-cause has been identified and "labeled", typically by a human expert. As manual labelling is costly and time consuming, another possible direction is to make use of the

Francesco Musumeci, Omran Ayoub and Massimo Tornatore are with Politecnico di Milano, Italy. Luca Magni, Roberto Rubino, Massimiliano Capacchione, Gabriele Rigamonti, Michele Milano and Claudio Passera are with SIAE Microelettronica, Italy.

Corresponding author email: francesco.musumeci@polimi.it.

large amount of unlabeled data typically collected by current network management systems in microwave networks. To make use of this unlabeled data, in this work we also propose to use unsupervised learning to automate the labelling of failed instances. Such unsupervised approaches are typically referred to as *data augmentation*. The objective of this further analysis is to understand in which conditions the automatic labeling improves the performance of supervised ML algorithms for failure identification.

We can summarize the contribution of this paper as follows: 1) we model the problem of failure-cause identification in microwave networks as a ML classification problem; 2) we propose a classification of main categories of failure causes; 3) we develop different supervised ML algorithms to perform the failure-cause identification, and we numerically evaluate them, using real-world data, in terms of classification accuracy and algorithm complexity; 4) we propose a data augmentation approach to perform automatic labeling of unlabeled data, and evaluate the performance improvement provided by automatic-labeling. Our numerical results show that satisfactory accuracy, up to 93%, can be achieved even with a relatively limited training data set of around 2000 data points.

The paper is organized as follows. In Sec. II relevant existing work is discussed. Sec. III provides background information on microwave networks. Sec. IV discusses the proposed categorization of failures in microwave networks. The supervised and semi-supervised failure identification problems are formally defined in Sec. V. Data preprocessing procedures are then presented in Sec. VI, whereas the supervised and semi-supervised methodologies adopted in this paper are presented in Sec. VII. Finally, we provide numerical results in Sec. VIII and draw paper conclusion in Sec. IX.

II. Related Work

In the last 20 years, failure management in communication networks has been investigated in both academia and industry. Similarly to our work, Refs. [4] and [5] are among the earliest works focusing on failure root-causes analysis in radio networks. In Ref. [4] authors applied correlation techniques based on causality graphs and associate failure root-causes to alarm sequences. Similarly, Ref. [5] adopts artificial neural networks to correlate the presence of different alarms in mobile network equipment to an initial cause generating the alarms sequence. Authors in [6] designed a framework for automatic anomaly detection and cause identification in mobile networks, based on observation of alarms and employing a decision process which emulates the human reasoning. However, unlike our paper, these works do not consider signal propagation phenomena and direct observation of signal transmission parameters, such as transmitted/received power values.

Although they are not directly focused on root-cause analysis, other recent works have used statistical and ML techniques for anomaly detection and prediction in radio networks, to understand if/when a failure is occurring. For example, Ref. [7] proposed a supervised learning approach to detect anomalies and predict failures so as to quickly intervene and repair any imminent microwave link failures. Decision trees have been used in [8] to perform anomaly detection considering synthetically generated data drawn from realistic microwave network statistics. Moreover, Ref. [9] used supervised learning to detect and localize failures in cellular networks, focusing on the observation of end-to-end (i.e., application level) service performance indicators. Ref. [10] proposed an anomaly detection framework considering time-series for various performance indexes and applying a regression algorithm in cellular networks. Unlike our paper, these works do not focus on the identification of the particular event leading to the failure (i.e., they do not perform root-cause analysis), but concentrate on a binary prediction to detect if a link will suffer an anomaly or not. Moreover, although some of these works adopt supervised techniques, they do not apply any semi-supervised learning, i.e., they do not consider the possibility of leveraging information from unlabeled data as we do in our study.

Semi-supervised and unsupervised approaches for failure and anomaly detection in contexts with scarce labeled data have been investigated in optical networks [11], [12]. In [11] authors use single-class Support Vector Machines for anomaly detection. This implementation is justified by the fact that anomalies in optical networks are rare and so the algorithm is trained with the N-centermost filter weights of samples where the failures are not present. Ref. [12] proposes a density-based unsupervised clustering algorithm to group and label the points in possible classes, identifying the outliers as network failures, and then the previously labeled data are used to train a neural network to identify failures via binary classification. Furthermore, Ref. [13] focused on comparing the performance of supervised, semi-supervised, and unsupervised learning approaches for attacks detection and localization in optical networks. Ref. [14] proposed an unsupervised learning technique which detects malicious attacks at the optical layer as anomalies without having prior knowledge of the signature of the attacks and their effects. Similarly, Refs. [15], [16] propose a hybrid unsupervised-supervised ML approach for anomaly detection in optical networks without prior knowledge of abnormal network behaviors. The approach is based on utilizing unsupervised data clustering module that self-learns abnormal network behaviors and detects anomalies, and then on utilizing a supervised data regression and classification modules based on deep neural networks.

To the best of our knowledge, no existing work has considered failure-cause identification in the specific context of microwave networks combining supervised and semi-supervised approaches, i.e., leveraging information from both labeled and unlabeled data to improve classification performance. Another valuable aspect in our paper is that we use real data collected from an operational microwave network consisting of more than 10 thousand point-to-point microwave links observed for a period of around 18 months. This manually-labeled dataset is then used both to develop supervised ML classifiers and to enable automatic labeling of an unlabeled dataset.

III. Background on Microwave Networks

A. Hardware Components

The basic structure of a microwave link with its building blocks is shown in Fig.1. At both transmitter and receiver

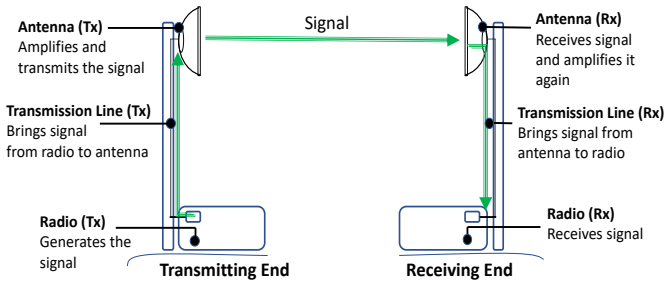


Fig. 1: Basic components of a microwave link.

sites, three main elements are present, i.e.:

1) *Microwave radio* is present at both Transmitter (Tx) and Receiver (Rx) sides, where it generates (resp., demodulates) the digital signal. At both sides, the microwave radio can be placed at different locations, i.e., either inside a building or shelter (*full-indoor*), in proximity of the antenna (*full-outdoor*), or by adopting a hybrid solution, called *split-mount*, where the electronic devices are distributed between an outdoor unit (ODU) and indoor unit (IDU).
2) *Transmission line* connects the microwave radio to the directional antenna. It is typically implemented via coaxial cables, suitable for frequency up to around 2 GHz, or waveguides, used for higher frequency up to around 13 GHz (and, in some rare cases, up to 40 GHz). Transmission lines are responsible for non-negligible signal losses, depending on the signal frequency, and may strongly affect the quality of transmission in case of physical medium deterioration.
3) *Directional antenna* is usually parabolic-shaped and is characterized by its, gain, size and directivity function, i.e., the capability of concentrating the transmitted/received power to/from specific directions.

Note that microwave links usually work in a bidirectional manner, i.e., at both ends, say site A and site B, of the link transmitting and receiving equipment is present for the communication in both directions A-to-B and B-to-A.

Microwave links are usually deployed with **Adaptive Code Modulation** (**ACM**), i.e., links can be configured to operate at different modulation formats, e.g., ranging from QPSK to 1024 QAM. Depending on channel quality, ACM enables automatic configuration of modulation format, based on measurements of the received power, signal-to-noise ratio (SNR), and BER.

B. Performance Metrics

The most common approach to monitor the performance of microwave links is to evaluate the number of errored bits in a certain time span. Consequently, link *unavailability* is defined in terms of *Unavailability Seconds*, or *UAS*, which represent the amount of time (expressed in seconds) when the number of errors exceeds a certain threshold.

Link unavailability metrics are specified in ITU-T Recommendations G.826 and G.828 [17], and include, among the others, the following definitions.

- *Errored Block (EB)*: a block (i.e., group of consecutive bits) in which one or more bits are in error.
- *Errored Second (ES)*: a one-second period with one or more errored blocks.
- *Severely Errored Second (SES)*: a one-second period which contains 30% errored blocks.

Based on these definitions, one bidirectional microwave link is considered as entering an *unavailable* state when at least ten consecutive SES are observed in at least one of the two directions A-to-B or B-to-A. Then, the link is considered again available if, after the block of consecutive SES, for at least ten consecutive one-second periods the link is not severely affected by errors (i.e., these periods do not contain more than 30% errored blocks) in both directions. As an example, starting at time $t = 0s$, if the link is characterized by 10 SES in $t = [0 \div 10)s$, 5 non-SES in $t = [10 \div 15)s$, 2 SES in $t = [15 \div 17)s$ and 10 non-SES in $t = [17 \div 27)s$, the total amount of seconds of unavailability, i.e., the total UAS, is equal to 17, as the link enters the unavailability state at $t = 0s$ and exits the unavailability state at $t = 17s$.

IV. Categories of Failures in Microwave Links

The availability of microwave links can be influenced by various factors, as atmospheric factors (e.g., fog, rain, humidity), temporary obstacles, or hardware failures. In this paper we concentrate on ML-based failure-cause classification, in order to automatically identify the root-cause phenomenon leading to microwave links unavailability. Note that this operation is carried on offline (e.g., once per day), based on data collected by the management systems. Although different countermeasures can be implemented, according to operator needs, after failure cause has been identified, here we do not focus on a specific failure handling architecture, which is not in the scope of the paper, but evaluate how ML algorithms perform in automatizing the failure-cause identification. As a first preliminary step to build a ML-based classifier for automated failure identification, in this section we propose a categorization of six different failure causes[1]. For each cause, different countermeasures are usually adopted to restore link availability, as described in the following.

1) **Deep Fading** consists of a strong increase of channel attenuation, and can be due to many factors such as, e.g., seasonality, geographical position or radio frequency in use. It can be caused by the presence of new obstacles (e.g., growth of vegetation) or adverse meteorological phenomena, such as heavy rain, snow or fog, leading to multipath and shadowing effects. To contrast deep fading, normally, no physical intervention is required on the radio equipment to restore link availability. In particular, in microwave links with ACM, deep fading is typically contrasted automatically by a temporary reduction of the modulation format, or in cases when the

[1]Note that, although in real microwave links many causes can be simultaneously present and concur to a link failure, here we assume that failures (i.e., unavailability) on a given link is caused by one event at a time. Investigation on correlations between different simultaneous causes on radio propagation is left for future work.

degradation is more persistent (e.g., due to the presence of new stable obstacles), via re-designing the radio link budget.

2) **Extra Attenuation** occurs when received power is well below (e.g. 6 or more dB lower) the minimum power threshold, even considering the lowest-order modulation format in the link[2]. Extra attenuation is a persistent effect, which cannot be contrasted in general through ACM. It can be caused, e.g., by path obstruction (due to the presence of permanent obstacles), antenna misalignment, mounting/screwing issues, water infiltration into waveguide used in the transmission line or damaged antenna/coupler. To solve this problem, human intervention is typically needed, either remotely or in field.
3) **Interference** occurs when other transmissions overlap at the receiving antenna at the same frequency, so that the multiple received bit streams are undistinguishable. The interfering channels can be, e.g., unexpected reflections from other links or frequency misconfigurations. Typically, interference does not change over time, so human intervention is needed to solve it, e.g., by turning off the interfering link or changing its carrier frequency.
4) **Low Margin**: we include in this class of failure causes the cases when the link configuration parameters have not been chosen adequately, i.e., they do correspond to the ones recommended by the manufacturer. In such cases, UAS events may occur although they could have been avoided though a more accurate link configuration. As an example, low margin may occur when ACM is disabled, or when the minimum modulation format configured on a link is not adequate (e.g., 16QAM is configured as minimum allowed on a link, but QPSK would be sufficient to guarantee a tolerable BER at the receiver even under a degraded propagation scenario). To address low margin failure types, remote human intervention is typically needed, such as, e.g., adjustment the lowest modulation formats allowed by ACM.
5) **Self-Interference** occurs when the link is operated in full-duplex, and the transmission line, which is shared between the two streams and connects the antenna to two radio components, creates local signal reflections and spurious signals which are propagated to the receiver radio component. Such failures can be caused by the degradation of the hardware (e.g., amplifiers and/or filters) used to eliminate signal reflections. It causes sporadic/random UAS as well as link instability (e.g. ACM switching, packet dropping) even though the link is working at nominal received power level and no fading event is occurring. In this case, human in-field intervention is usually needed, e.g., to substitute hardware components.
6) **Hardware Failure**: we include in this category all the cases when link unavailability is not directly related to evident propagation problems. Moreover, we include here the cases when multiple causes are concurring to produce link unavailability. This category includes both permanent and temporary failures, for which a more accurate troubleshooting is needed to identify the ultimate failure cause and implement proper countermeasure. We are currently investigating different features, such as the alarms issued in radio equipment, and on how to relate such features to different possible causes in this class.

[2]Note that, for a given modulation format, a corresponding minimum received power threshold can be set to guarantee a predefined maximum BER at the receiver.

V. Failure-Cause Identification: Problem Definition

In this section we first describe problem inputs and outputs and then provide formal problem definitions, and identify the research questions addressed in this study.

A. Problem Inputs and Outputs

Our dataset is collected over a set of 10841 links of a real microwave network. For both sites of each network link (i.e., site A and site B), several performance metrics have been collected at fixed time-steps of 15 minutes via a network management system (NMS) developed by SIAE Microelettronica. The overall data collection period spans between Nov. 26th 2017 and May 5th 2019.

For each microwave link in the dataset and for each 15-minutes slot, the SIAE NMS collects 39 different measures, which constitute the inputs of our problem and which we group in the following categories:

- **General link information** $(x_1 \div x_7)$ include information that uniquely identifies link, i.e., they include an identification number (ID-Link), the date and time when the measures have been collected, the geographical location, the network operator and the IP addresses used at the transmitter/receiver equipment at the two sides of the link.
- **Design information** $(x_8 \div x_{17})$ relates to those parameters that are set during the design phase; they include the protection technique[3] (x_8), the lowest and highest modulation formats configured in the link $(x_9 - x_{10})$, a flag indicating if the ACM is enabled on the link (x_{11}), and the type of equipment used in the link (x_{12}). Three types of equipment are considered in our dataset, namely, *AlcPlus2e* and two devices of *SM-OS* family, all provided by SIAE Microelettronica [18]. We also include 5 other metrics, namely, the nominal (i.e., expected) received and transmitted power associated to the lowest modulation format set in the link $(x_{13} - x_{14})$, the nominal received power associated to the highest modulation format which can be used in the link (x_{15}), the carrier frequency (x_{16}) and the bandwidth (x_{17}) associated to the link.
- **G.828 performance measures** $(x_{18} \div x_{20})$ include the availability and error performance metrics described in Sec. III-B, i.e., for each link and each 15-minutes slot we include ES, SES and UAS.
- **Propagation measures** $(x_{21} \div x_{29})$ include, for both sides of the link, the minimum and maximum received

[3]Note that in our study we always consider unprotected links, i.e., 1+0 configurations

and transmitted power values as measured along the 15-minutes slot. Furthermore, the lowest modulation adopted during the 15-minutes slot is also included.

- **Quality flags** $(x_{30} \div x_{32})$ are set by the NMS to indicate if the measurements collected in the 15 minutes are *reliable* or not (e.g., due to lack of communication between the NMS and the equipment) and they refer to the G.828 performance measures $(x_{18} \div x_{20})$, the transmitted power measures $(x_{21} \div x_{24})$ and the received power measures $(x_{25} \div x_{29})$, respectively. In other words, if flags $x_{30} \div x_{32}$ are set to 1, the management systems is informing if the communication between the link equipment and the data collection agent is alive, otherwise the flags are set equal to 0.
- **Other measures** $(x_{33} \div x_{39})$. In this category, we include measurements that are not related to the entire 15-minutes slot, but represent instantaneous measurements performed at the end of the slot. Specifically, x_{33} and x_{34} are the SNR values at both sites of the link, $x_{35} = min\{x_{33}, x_{34}\}$, x_{36} and x_{37} are the modulation formats used in the two directions, x_{38} and x_{39} are the received power values at both sites.

In Sec. VI we describe how we elaborate these 39 inputs to obtain the features used as inputs to the ML failure identification algorithms. In particular, to capture how these characteristics vary over time, each data point in the dataset is constituted by a 45-minutes *window*, i.e., three consecutive slots of 15-minutes and all their collected metrics, where at least one UAS event is present at the third 15-minutes slot[4]. This filtering operation, applied on the entire dataset of 15-minutes slot, has led an elaborated dataset consisting of around 10 millions 45-minutes *windows* in total. Out of these, a total of 2513 45-minutes windows have been then manually labeled by domain experts with a label representing one of the failure causes described in Sec. IV. We identify the following labels associated to the 45-minutes windows: y_1**: Deep Fading**, y_2**: Extra attenuation**, y_3**: Interference**, y_4**: Low margin**, y_5**: Self-interference**, y_6**: Hardware failures**. To give a rough idea of the effort required for the manual data labeling, two domain experts have spent around two weeks to label all the 2513 windows. In the following, we assume that all manually-labeled windows are correct and provide the **ground truth**.

B. Problem Statement

In this paper we address two problems related to the identification of failure causes in microwave networks.

Problem 1: Failure-cause identification (supervised ML). We model this problem as a multi-class classification problem. More specifically, **given** a 45-minutes window observation on a microwave link where the 39 inputs $(x_1 \div x_{39})$ described in Sec.V are available for each of the constituting 15-minutes slots, and for which at least one UAS event is present at the last 15-minutes slot (i.e., a failure event has been detected and the link is considered unavailable, at least in this slot), we **estimate/identify** the failure cause among those described in Sec.V $(y_1 \div y_6)$[5]. In our numerical analysis we will use different amount of labeled data taken from our complete dataset, and select different existing ML classifiers for failure-cause identification in order to address the following research questions:

RQ1.1. *Which is the most suitable ML algorithm in terms of accuracy and time complexity?*

RQ1.2. *How much labeled data do we need to get acceptable performance for each algorithm?*

Problem 2: Data augmentation for failure-cause identification (semi-supervised learning). In this case, our objective is to investigate how unlabeled data can be effectively used to improve failure-cause identification. To this end, we propose a procedure based on semi-supervised learning, which we use to perform automatic data labeling of massive amounts of unlabeled data. So, the automatically-labeled data will be included in the training set of new ML classifiers developed via a supervised-learning approach. More formally, we are **given** a *labeled dataset* and an unlabeled dataset. Both sets include 45-minutes windows where, for each window, the 39 inputs $(x_1 \div x_{39})$ described in Sec.V are available for each of the constituting 15-minutes slots, and for which at least one UAS event is present at the last 15-minutes slot. In the labeled dataset, each 45-minutes window is assigned a label (i.e., a failure cause) chosen among the causes $(y_1 \div y_6)$. The **output** of this problem is an automatic labeling procedure to **decide** the labels associated to the points in the unlabeled dataset, and identifying their failure causes. This way, we are able to develop new ML classifiers using a much larger labeled dataset, constituted by the manually-labeled dataset and automatically-labeled dataset. The goal of the analysis is to address the following research questions:

RQ2.1. *Does the automated labeling procedure provide an improvement on the classification accuracy and for which ML algorithm the improvement is most significant?*

RQ2.2. *How much data, labeled automatically, do we need to improve the classification performance?*

C. Supervised and Semi-Supervised Failure-Cause Identification Framework

Fig. 2 shows the overall framework of the supervised and semi-supervised ML approaches considered in this paper, which are detailed in the next sections. For both frameworks, **Data Preprocessing** is performed at first (see Sec. VI).

Then, considering different ML algorithms, we perform model through k-fold cross-validation (explained in Sec. VII), and evaluate ML algorithms performance in terms of accuracy and training duration. Note that, in the semi-supervised learning case, before ML model selection, we perform **Data Augmentation**, which consists of i) feature engineering and ii) automatic labelling of unlabeled data (see Sec. VII-B).

[4]Note that we do not consider windows where the last 15-minutes slot is not characterized by UAS as such situation is representative of disappearing degradation phenomena.

[5]Note that our objective is to automate the failure cause classification, which is typically addressed by domain experts in an **offline** manner in current microwave networks.

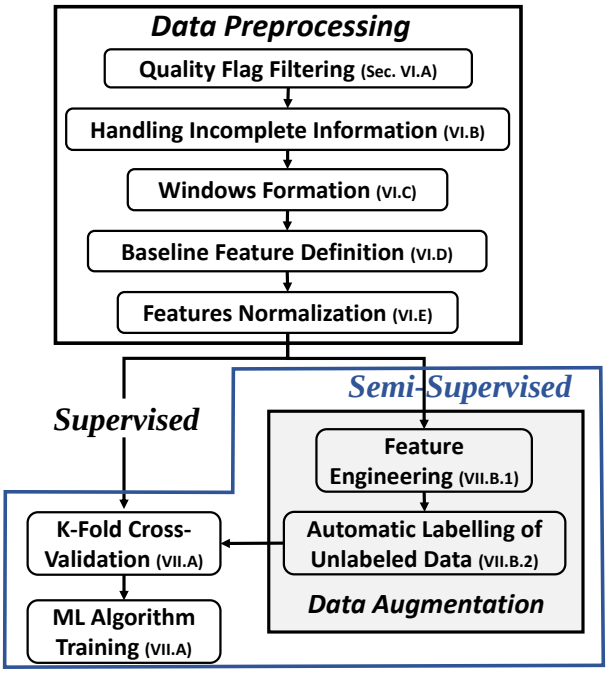


Fig. 2: Overall framework of the Supervised and Semi-Supervised ML approaches

VI. Data Preprocessing

Data used in this work is obtained from a real microwave network. To transform the collected raw data in data suitable for training of our ML algorithms, we must first perform significant data preprocessing as described in the following.

A. Windows Formation

To capture the variation of links characteristics over time and limit computational complexity, we consider 45-minutes windows constituted by three consecutive slots of 15 minutes, each featured by the inputs $x_1 \div x_{39}$ described above.

Each 45-minutes window in our dataset is characterized by at least one UAS event in the last 15-minutes slot, as shown with an example in Fig. 3, where we show a sequence of 15-minutes slots for one link along a 2.5 hours observation period, and the corresponding 45-minutes windows included in our dataset out of this overall time span. Assuming that UAS events have occurred in two consecutive slots (i.e., 2:30-2:45 and 2:45-3:00), two 45-minutes windows are formed, i.e., 2:00-to-2:45 and 2:15-to-3:00. Note that, observing historical data and thanks to the knowledge of domain experts, a 45-minutes time span is deemed sufficient to capture temporal dynamics of failure causes in microwave links, also in cases when the 3rd slot is the only one affected by UAS (such as the third 15-minutes slot in "Window 1" in Fig. 3).

B. Quality Flags Filtering

In some cases, data retrieved from the field can be inconsistent, e.g., because the communication channel between the NMS and the device performing the measurement is temporarily interrupted. In these cases, the NMS cannot collect

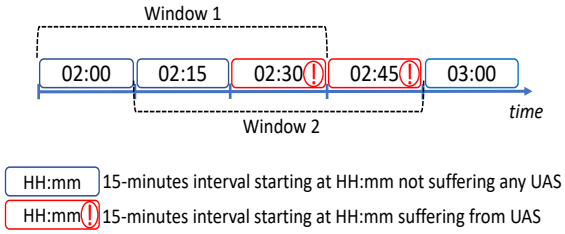


Fig. 3: Example of 45-minutes windows related to a failure (i.e., unavailability) event occurred on a link within a period of 30 minutes.

any measurement, so one or more of the quality flags $x_{30} \div x_{32}$ described above, is set to 0 by the NMS when providing the measurements. In our analysis, we consider only the windows where data is consistent along the entire 45-minutes period.

C. Handling incomplete information

In real scenarios, data can be partially available, so it is important to properly manipulate missing information to avoid negative impact on the ML classifiers' accuracy. Different approaches can be used to handle incomplete datasets [19], e.g., discard data points with partial information, provide an estimation for the missing information, or enforcing static numerical values.

In our case, due to its scarcity in the dataset, we completely neglect input x_{29} indicating the lowest modulation adopted during the 15-minutes slot. Concerning the values of minimum and maximum received and transmitted power at both sides of the microwave link ($x_{21} \div x_{28}$), we enforce "out-of-range" values in case of missing information, exploiting the knowledge of the distributions of such metrics. Specifically, considering that, for the equipment in our dataset, the received and transmitted power values range between [-98,-28] dBm and [-20,28] dBm, respectively, we force numerical values of -150 dBm and 100 dBm for minimum/maximum received and transmitted power, respectively. These values have been chosen as they are numerical values (hence, they can be handled by ML algorithms), but, since they are far enough from the expected values, they still capture unequivocally the difference with the cases when information is not missing.

D. Choice of Input ML Features

We consider data points as 45-minutes windows consisting of three consecutive 15-minutes slots. However, for each 45-minutes window, we do not consider all the $39 \cdot 3$ features as inputs of the ML algorithms but define an overall set of 35 features by elaborating the 39 inputs $x_1 \div x_{39}$ for each 15-minutes slot constituting a window.

The features, described in Tab. I, include: ($f_1 \div f_5$) the link characteristics, which do not depend on the 15-minutes slots and represent the design parameters of the links; ($f_6 \div f_{11}$) the G.828 performance measures ES and SES for the three 15-minutes slots; ($f_{12} \div f_{35}$) the minimum/maximum received and transmitted power values for each side of the link (i.e., site A and site B) and for each of the three 15-minutes slots.

Tab. I: Features describing a 45-minute window of the radio link. Feature names with '*' are measurement features with three different values, one for each 15-minutes slot.

Type	Feature	Name	Description
Link Characteristics	f_1	LowThr	Minimum received power tolerated on the link with any modulation format used (dBm)[6]
	f_2	Ptx	Nominal transmitted power when the minimum modulation format is used (dBm)
	f_3	Thr_min	Minimum received power threshold tolerated by the link with its current modulation format (dBm)
	f_4	RxNominal	Nominal received power at the maximum modulation format (dBm)
	f_5	acmEngine	A flag which indicates if the ACM is enabled on a given microwave link
G.828 metrics	f_6, f_7, f_8	ES*	Number of one-second periods with at least one ES in the 15-minutes slot
	f_9, f_{10}, f_{11}	SES*	Number of one-second periods with at least one SES in the 15-minutes slot
Power values	f_{12}, f_{13}, f_{14}	txMaxA*	Maximum power transmitted from site A in in the 15-minutes slot (dBm)
	f_{15}, f_{16}, f_{17}	txminA*	Minimum power transmitted from site A in the 15-minutes slot (dBm)
	f_{18}, f_{19}, f_{20}	rxmaxA*	Maximum power received at site A in the 15-minutes slot (dBm)
	f_{21}, f_{22}, f_{23}	rxminA*	Minimum power received from site A in the 15-minutes slot (dBm)
	f_{24}, f_{25}, f_{26}	txMaxB*	Maximum power transmitted from site B in the 15-minutes slot (dBm)
	f_{27}, f_{28}, f_{29}	txminB*	Minimum power transmitted from site B in the 15-minutes slot (dBm)
	f_{30}, f_{31}, f_{32}	rxmaxB*	Maximum power received at site B in the 15-minutes slot (dBm)
	f_{33}, f_{34}, f_{35}	rxminB*	Minimum power received from site B in the 15-minutes slot (dBm)

E. Features Normalization

A typical operation performed before training ML algorithms is features normalization, and is useful to improve the performance of the ML classifier as it makes the algorithm less sensitive to scale of data, and allows to speed-up ML algorithm convergence [20].

Formally, for each feature f_i $(i = 1, 2, ..., 35)$ we calculate the mean $\bar{f}_i$ and the standard deviation σ_i considering all the data points in the dataset. Then, for each data point $j \in X$, characterized by features $(f_i^j,\ i = 1, 2, ..., 35)$ we obtain the the standardized features by updating their values as follows:

$$f_i^j \leftarrow \frac{f_i^j - \bar{f}_i}{\sigma_i} \tag{1}$$

VII. Supervised and Semi-supervised Failure-Cause Identification

A. ML Algorithms

Failure-cause identification is finally performed using various multi-class ML classifiers, each based on a different ML algorithm, namely, Artificial Neural Network (ANN), Support Vector Machine (SVM) and Random Forest (RF). Various combinations of hyperparameters have been tested to obtain, for each algorithm, the classifier with highest classification accuracy.

Specifically, the combinations of hyperparameters which have been evaluated are described in the following (see, e.g., [3] and the references therein).

- ANN: we consider multi-layer ANNs, where we tune the number of hidden layers between 1 and 10 and the number of neurons per layers in the set {10, 50, 100}, and vary the activation function used in each neuron considering the {*Identity*, *Sigmoidal*, *Tanh*, *Relu*} activation functions.
- SVM: for the *slack* variable, which drives the SVM margins between the various classes, we consider multiples of 5 in the range $[5 \div 100]$; moreover, we consider different SVM kernels, chosen from the set {*linear*, *polynomial*, *radial basis function*, *sigmoid*} and evaluate both One-vs-One and One-vs-All strategies to define the optimum multiclass SVM classifier.
- RF: we consider different number of trees, ranging between 10 and 250 with step of 10, and splitting criterion, chosen between *Gini Index* and *Cross-Entropy* [3].

B. Data Augmentation for Failure-Cause Identification

As the amount of available labeled data is usually scarce, due to time consuming and costly labeling operations, a second objective of this paper is to evaluate the benefit of using unlabeled data to augment the labeled dataset, and so to improve failure classification performance. To this end, we investigate a data augmentation methodology to automatically label unlabeled data.

Note that a trivial way to perform automatic labeling could be re-using the classifiers developed with manually-labeled data. However, after testing this approach for all the classifiers (i.e., based on ANN, SVM and RF) and using the original 35 features, we did not find any improvement in the classification performance for all the ML algorithms evaluated. Therefore, we resort to a more sophisticated data augmentation procedure, consisting of two main steps, i.e., 1) *features engineering*, and 2) *automatic labeling of the unlabeled data*, which are described in detail in the following.

1) Features engineering: To perform automatic labeling of the unlabeled data, i.e., of the 45-minutes windows $k \in X_U$, we exploit the available manually-labeled data points, i.e., the dataset $[X_L, Y_L]$, where each 45-minutes window $j \in X_L$ is characterized by features $(f_i^j,\ i = 1, 2, ..., 35)$ and is assigned a label $y_j \in \{1, 2, 3, 4, 5, 6\}$, corresponding to a failure cause as described in Sec. IV.

However, to perform accurate automatic labelling, it is important to find a proper features space, which captures similarities between data points in X_L and X_U. To do so, we test various features spaces by checking if they verify the *clustering assumption* on the labeled dataset $[X_L, Y_L]$ [21]. For each features space, the clustering assumption is verified if, by performing unsupervised clustering on dataset X_L, we obtain groups which are in accordance with labels Y_L.

To quantitatively evaluate the clustering assumption, we use a numerical metric called Adjusted Rand Index (ARI) [22], which is generically used to calculate the similarity between two different partitions of the same dataset. More specifically,

ARI is a real number ranging between -1 and 1, where values close to 1 correspond to high similarity, i.e., they indicate that, in the two partitions, two given points are more likely to be grouped in the same cluster. In our problem, the first partition $\mathcal{P}_1$ is provided with labels as it is taken from the know manually-labeled dataset $[X_L, Y_L]$, which contains 6 groups identified by labels $y_1 \div y_6$. The second partition $\mathcal{P}_2$ is a clustering on the set X_L, performed considering the features space $\mathcal{F}$ under test. Our objective is to find the features space $\mathcal{F} = \mathcal{F}*$ such that the $ARI(\mathcal{P}_1, \mathcal{P}_2)$ is maximized.

As preliminary approach, we considered the basic features space $\mathcal{F} = \{f_1 \div f_{35}\}$, but found values of ARI close to 0.3, which was not satisfactory.

Moreover, we also tried simple features engineering adding linear combinations of the original features $(f_i, i = 1, 2, ..., 35)$ described in Sec. VI. We evaluated several features spaces, where, besides the original features $(f_i, i = 1, 2, ..., 35)$, we included around fifty linear combinations of these features, obtained by leveraging on SIAE experience and background knowledge on microwave networks, such as the difference between the nominal received power *RxNominal* and the minimum received power at sites A and B[7]. However, such simple features engineering approaches, based on adding linear combinations of original feature, did not give satisfactory results in terms of ARI, which was well below 0.5 for all the tested cases.

Therefore, a more sophisticated features engineering is proposed in this paper, which leverages the power of ANNs with particular symmetrical structures, namely, the *Autoencoders*.

Autoencoders are often used to perform dimensionality reduction and find compressed representation of the data, that can be used both for help the classification task and to execute data visualization procedure. The structure of an autoencoder is symmetric and composed by three main elements, i.e.: 1) the *encoder* is constituted by the ANN input layer (say of size F) and the initial set of hidden layers, and it is used to compress the data, i.e., to reduce the number of raw features ($F = 35$ in our case) into a smaller number $D < F$; 2) the *bottleneck layer*, which is the intermediate hidden layer constituted by D nodes where the D coded features are collected and which will represent the new features space in our problem; and 3) the *decoder*, that is symmetric with respect to the encoder, is constituted by the remaining hidden layers and the output layer in the ANN, where the number of output neurons is equal to the number of input neurons F. The goal of the decoder is to recompute the input data starting from the encoded features provided by the encoder. When autoencoders are used in unsupervised learning problems, they are trained by considering, for each training point j with features f_i^j $(i = 1, ..., F)$ *fictitious* labels of type $y_i^j = f_i^j$ $(i = 1, ..., F)$ for all neurons at the output layer. Therefore, the final goal of autoencoders is to provide a new set of *encoded features vector* $\Phi = \{\varphi_1, ..., \varphi_D\}$ at the bottleneck layer. These D features are non-linear (i.e., *encoded*) combinations of the original raw features, and can be used to reconstruct an approximated representation of the input data through the decoder part of the autoencoder.

[7]For sake of conciseness, we do not list all the linear combinations of features included in this evaluation as they did not provide significantly good results in terms of ARI.

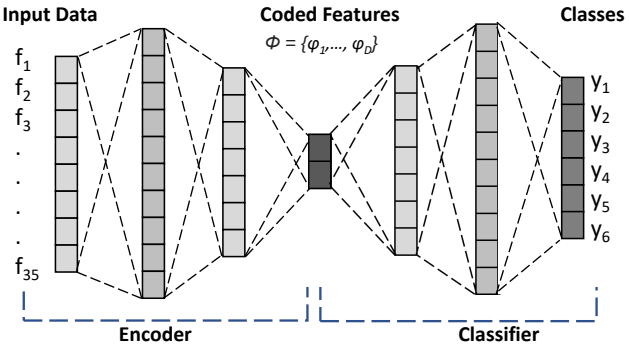


Fig. 4: Example of autoencoder structure used in the feature engineering phase.

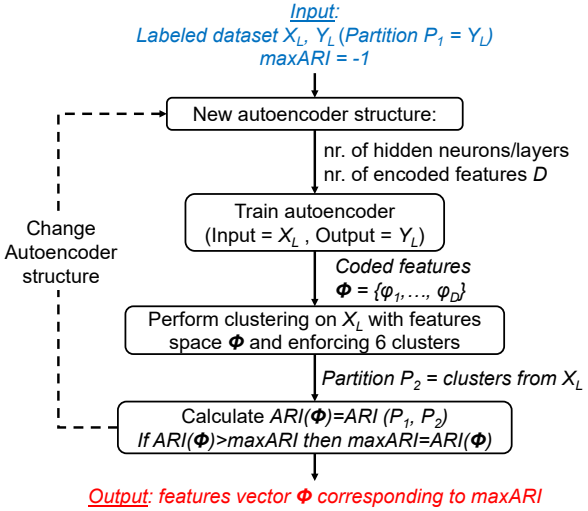


Fig. 5: Flow-chart of the features engineering used to obtain features space Φ with maximum ARI.

Inspired by autoencoders abilities and leveraging the knowledge provided by the available labels for data points in the labeled set $[X_L, Y_L]$, we consider a slightly modified approach, where the final output layer is characterized by 6 output neurons (instead of 35), representing the classes, i.e., the labels, of the 6 failure causes (see Fig. 4). Then, after finding the best autoencoder structure (i.e., the one characterized by a number of layers, number of neurons in each layer, number of neurons in the *bottleneck layer* providing the highest ARI), the obtained autoencoder is trained using the 45-minutes labeled windows in set $[X_L, Y_L]$.

The steps adopted to perform the ARI-maximized search of features space Φ are summarized in the flow-chart in Fig. 5. Note that the features engineering process using autoencoders might be helpful also in the baseline supervised case. To this end, we have compared the three ML algorithms described above when we use or do not use features engineering (through autoencoders) to transform the features space. We obtained very negligible accuracy differences, with only some small gains for ANN and SVM algorithms. Due to space limitations, we do not show this analysis in the paper.

2) Automatic labeling of unlabeled data: After obtaining the best-performing features space $\mathcal{F}*$ and applying features

normalization to it, we perform an automatic labeling of unlabeled points in X_U, based on the following two different approaches:

- *K-Nearest Neighbours (KNN)*. All the unlabeled points are assigned a label in this case; for a given point $j \in X_U$, its label $y(j)$ is assigned considering considering the K nearest points $k_1, k_2, ..., k_K \in X_L$, based on euclidean distance and considering the normalized features space $\mathcal{F}*$. For each of these K nearest points, the labels are weighted according to its distance from point j, so the label is assigned according to formula:

$$y(j) = argmax_i \sum_{l=1}^{K} 1/d(j,k_l) \cdot \mathbf{1}_{y(k_l)=i}, \quad (2)$$
$$\forall j \in X_U, i = 1, ..., 6$$

  where $d(j,k_l)$ is the euclidean distance between data points $j \in X_U$ and $k_l \in X_L$ considering the normalized features space as defined by the autoencoder structure, and $\mathbf{1}_{condition}$ is equal to 1 if $condition == TRUE$.
- *Gaussian Processes*. in this case, we assume that the points in each class are normally-distributed. To assign a label to an unlabeled point $j \in X_U$, for each class $i = 0, ..., 5$, the model provides in output a probability that point j belongs to class i, say $P\{j|i\}$. Then, we assign to point j label y_i such that

$$y(j) = y_i \text{ s.t. } y_i = argmax_i P\{j|i\}, \quad (3)$$
$$\forall j \in X_U, i = 1, ..., 6$$

  Note that, when adopting the Gaussian-process-based labeling, we define a *confidence threshold* ρ to assign labels, and include in the automatically labeled dataset only the points for which the highest probability provided in eq. 3 is higher than this threshold.

VIII. Numerical Results

In this section we perform numerical evaluation of the supervised and semi-supervised failure identification. First, we provide details on the datasets used for our analysis. Then, considering only the manually-labeled dataset $[X_L, Y_L]$, we compare the performance of the different ML algorithms (ANN, SVM and RF), in terms of classification accuracy and training duration. Finally, we perform the same comparison considering also data augmentation, i.e., after adding to the ground truth dataset the automatically labeled data $[X_U, Y_U]$.

Our experiments have been conducted on a PC with Intel Core i7-6700 processor, 32 GB RAM and 402.5 GB HDD, and ML algorithms have been implemented using Python libraries (*sklearn*, *pandas*, *tensorflow* and *keras*).

A. Datasets

The manually-labeled dataset $[X_L, Y_L]$ consists of 2513 data points (i.e., 45-minutes windows with at least one UAS event in the last 15-minutes slot) in total. Labels are distributed among the 6 classes as shown in Tab. II. A severe unbalance between some classes can be observed, due to the fact that some failure causes, such as, e.g., *interference*, are less frequent.

Tab. II: Distribution of data points in manually-labeled dataset $[X_L, Y_L]$ over failure classes.

Failure Cause	**# of 45-minutes windows**
y_1 - *Deep Fading*	284
y_2 - *Extra Attenuation*	581
y_3 - *Interference*	49
y_4 - *Low Margin*	190
y_5 - *Self-Interference*	187
y_6 - *Hardware Failure*	1222

Tab. III: ML algorithms and selected hyperparameters.

ML algorithm	**Parameter**	**Value**
ANN	Number of Hidden Layers	5
	Number of Nodes per Layer	100
	Activation Function	Relu
SVM	Slack Variable	85
	Kernel	Radial Basis Function
	Decision Function	One-Vs-One
RF	Number of Trees	70
	Cost Function	Cross-entropy

For data augmentation, we use a set of unlabeled data collected in the entire period between Nov. 26th 2017 and May 5th 2019. This dataset, which is referred to as $[X_U, Y_U]$ in the following, consists of a total of 10500 45-minutes unlabeled windows, still characterized by having at least one UAS event at the third 15-minutes slot in the 45-minutes window.

B. Supervised ML Algorithms

First we compare ANN, SVM and RF ML algorithms in terms of classification accuracy and training duration, using only the manually-labeled dataset $[X_L, Y_L]$. The objective is to identify which algorithm is the most suitable to perform failure-cause identification.

We define a fixed 80%-20% training-test partition and, for each ML algorithm, we perform model selection by applying 10-fold cross-validation on the training set only. More specifically, for each algorithm, we identify the best combination of hyperparameters as the one having the highest average accuracy on the 10 folds of the training set, where each time the training is performed on 9 folds (i.e., 90% of the training set) and the validation on the remaining fold (i.e., 10% of the training set). Thus, repeating this procedure for all the combinations of hyperparameters and for all the 10 folds used as validation set one at a time, we obtain the best combination of hyperparameters for each algorithm as shown in Tab. III.

After model selection is completed for each algorithm, the obtained models (i.e., the ML classifiers with the hyperparameters selected as in Tab. III) are retrained on the entire training set, i.e., the 80% of the manually-labeled dataset $[X_L, Y_L]$, and the accuracy for each algorithm is evaluated on the independent fixed test set, constituted by the remaining 20% of dataset $[X_L, Y_L]$.

Note that the performance obtained with this procedure are influenced by the initial 80%-20% split between training and

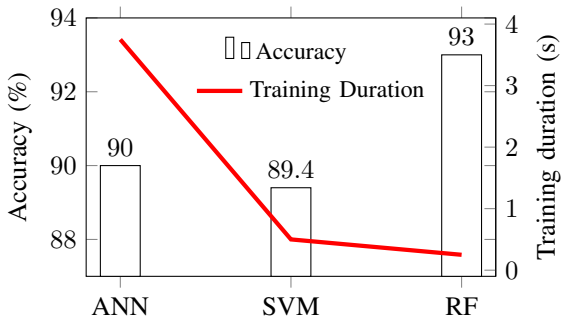


Fig. 6: Failure identification results for the different ML algorithms: cross-validation accuracy and training duration.

test set. So, to provide more generalized results, an "outer" cross-validation might be applied by varying the training-test split and repeating this procedure several times.

Numerical results are shown in Fig. 6 in terms of classification accuracy and training duration. The results show a supremacy of RF in both the accuracy and the computational time. In particular, RF provides more than 3% higher accuracy compared to SVM and ANN, and negligible training duration, especially if compared to ANN. On the other hand, ANN provides around 1% higher accuracy compared to SVM, but a much longer training time, about 7 times higher. The low training time provided by RF can be explained considering that the different trees used in the algorithm are independent, so their training can be parallelized and, consequently, the computational time decreases drastically. Moreover, regarding the highest performance obtained by RF in terms of classification accuracy in comparison to the ANN and SVM, the motivation resides on the fact that RF classification is performed with a similar "human" reasoning as the one adopted when performing data labeling, e.g., based on the observation of windows features and "*if-then-else*" classification.

To see classifiers behaviour in identifying the different failure causes, we show in Tab. IV the confusion matrix of the best RF classifier (i.e., the one with the hyperparameters leading to the highest average accuracy across all the 10 folds used during cross-validation[8]) used on the independent test set with 503 data points, corresponding to 20% of the manually-labeled dataset $[X_L, Y_L]$. In the table, we also show the F1-score for each class, which shows that for some failure causes, e.g., y_5 (*Self-Interference*) and y_6 (*Hardware Failure*), better performance are obtained compared to other causes. This demonstrates that these causes are discriminated more easily with respect to the others, whereas in some cases, e.g., for the *Low Margin* class (y_4), additional information might be needed to improve the failure-cause identification performance.

C. Semi-Supervised Algorithms: Impact of Automatically-Labeled Data

Now we concentrate on data augmentation and on the benefits provided on classification accuracy when leveraging the knowledge of the manually-labeled dataset $[X_L, Y_L]$ to perform automatic labeling of unlabeled points X_U using different

[8]Note that we similar outcomes can be drawn considering RF classifiers with other hyperparameters set.

Tab. IV: Confusion matrix and per-class F1-score obtained for the RF classifier with highest cross-validation accuracy.

		Predicted Label					
		y_1	y_2	y_3	y_4	y_5	y_6
True Label	y_1	50	2	0	4	0	1
	y_2	5	106	0	5	0	0
	y_3	0	0	8	1	1	0
	y_4	5	2	0	31	0	0
	y_5	0	0	0	0	36	1
	y_6	0	6	0	1	0	238
F1-score		0.85	0.91	0.89	0.78	0.97	0.98

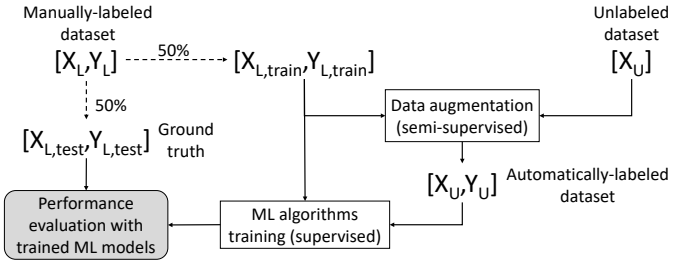


Fig. 7: Datasets usage in the semi-supervised use case.

labelling approaches. After this procedure, we can evaluate the performance of the various ML algorithms using a new training set $[X_{L+U}, Y_{L+U}]$, where $X_{L+U} = X_L \cup X_U$, $Y_{L+U} = Y_L \cup Y_U$ and where labels Y_U have been obtained via automatic labeling as described in Sec. VII-B. To show the high-level difference between the various automatic labeling approaches, we consider a fixed ground-truth test set, $[X_{L,test}, Y_{L,test}]$, which is obtained by randomly choosing 50% of the data points in the manually-labeled dataset $[X_L, Y_L]$ (i.e., $X_{L,test} \subseteq X_L$ and $Y_{L,test} \subseteq Y_L$), constituted by a total of 1257 45-minutes windows. The remaining part, say $[X_{L,train}, Y_{L,train}]$, where $X_{L,train} = X_L \setminus X_{L,test}$ and $Y_{L,train} = Y_L \setminus Y_{L,test}$, is used to perform data augmentation (i.e., features transformation and automatic labeling as described in Sec. VII-B). Note that the split between $[X_{L,train}, Y_{L,train}]$ and $[X_{L,test}, Y_{L,test}]$ is performed by maintaining the classes proportion as in Tab. II. The usage of the various datasets is summarized in Fig. 7.

In Tab. V, we show how the data points in $[X_{L,test}, Y_{L,test}]$ are distributed between the 6 considered failure causes after applying the automatic labeling procedure. We compare the three different automatic labeling procedures, i.e., *KNN*, *GP-60* and *GP-90*, where the last two approaches correspond to the case of Gaussian-process-based automatic labeling with confidence level set to $\rho = 60\%$ and $\rho = 90\%$, respectively.

As observed in Tab. V, all the 1257 points in $X_{L,test}$ are assigned a label in the *KNN* case, as KNN enforces labels to each unlabeled point by weighting the labels of its K nearest neighbours according to eq. 2. It can be observed also that the percentual distribution of data points obtained with *KNN* resembles approximately the one of the points in the entire labeled dataset $[X_L, Y_L]$. This is motivated by the fact that the autoencoders (and therefore, the encoded features vector Φ) used to perform the automatic labeling has been chosen with the objective of maximizing ARI.

On the other hand, in the *GP-60* and especially *GP-90* cases,

Tab. V: Distribution of data points in automatically-labeled dataset $[X_{L,test}, Y_{L,test}]$ over failure classes for the different automatic labeling procedures (*KNN*, *GP-60* and *GP-90*).

Failure Cause	***KNN***	***GP-60***	***GP-90***
y_1 - *Deep Fading*	140	121	0
y_2 - *Extra Attenuation*	302	293	259
y_3 - *Interference*	27	10	0
y_4 - *Low Margin*	105	89	0
y_5 - *Self-Interference*	95	91	0
y_6 - *Hardware Failure*	588	580	550
Total	1257	1184	809

compared to the *KNN* case, a lower amount of data can be labeled with a predefined confidence level. However, for some classes, such as y_2 and y_6, still a significant amount of data can be included, even in the *GP-90* case. This demonstrates that, for the considered features space obtained after applying the features transformation through the best autoencoder obtained for each scenario, these classes are still highly compact and so the transformed features space is helpful in performing reasonable automatic labeling to augment the original dataset with high number of data points.

Now we evaluate how the three data augmentation scenarios, i.e., *KNN*, *GP-60* and *GP-90*, impact on the performance of the ANN, SVM and RF classifiers. Note that, as we want to evaluate how the classification accuracy varies with the number of available manually-labeled windows, we consider the following three cases: A) 100% $X_{L,train}$, B) 75% $X_{L,train}$ and C) 50% $X_{L,train}$, where we assume to have as a starting point 100%, 75% or 50% of the data in the training set $[X_{L,train}, Y_{L,train}]$. In the 75% and 50% cases, data points are always selected by maintaining classes proportions as in Tab. II. Also, we repeat the procedure with 3 different random sampling in both the 75% and 50% cases and provide averaged accuracy results[9]. For each scenario, aiming at obtaining a features set $\mathcal{F}$ which maximizes ARI, we tested several autoencoders structures by considering different combinations of hyperparameters, i.e., we vary the number of hidden layers in the range $[3, 9]$, the number of hidden neurons per layer in the range $[18, 1000]$ and the number of nodes in the *bottleneck layer* (i.e., the number of encoded features D) in the range $[1, 17]$. Moreover, as activation function used in the hidden and output layers, we consider only hyperbolic tangent[10].

In Fig. 8(a) we show classification accuracy of the **Baseline** scenario with supervised dataset only and with original features $f_1 \div f_{35}$ as in Tab. I[11]. Moreover, we show how the classification accuracy changes when the three data-augmentation scenarios are applied:
- ***GP-90***: automatic labeling with *GP-90* and encoded features (Fig. 8b),
- ***GP-60***: automatic labelling with *GP-60* and encoded features (Fig. 8c), and
- ***KNN***: automatic labelling with *KNN* and encoded features (Fig. 8d).

Observing Fig. 8, we note that the best accuracy values are obtained when adopting RF and considering only the labeled dataset $[X_{L,train}, Y_{L,train}]$ and using the original 35 features as in Tab. I, i.e., for the RF case shown in Fig. 8a. This result is obtained independently of the amount of labeled data used, and it reaches around 92.9%, 91.8% and 90.9% for the cases of 100% $X_{L,train}$, 75% $X_{L,train}$ and 50% $X_{L,train}$, respectively. Comparing the cases with different amount of training points (i.e., 100% $X_{L,train}$ vs 75% $X_{L,train}$ vs 50% $X_{L,train}$), as expected, accuracy improves for all the ML algorithms with increasing amount of manually-labeled points.

Figs. 8b-to-d show the benefits/disadvantages of the different data augmentation scenarios (i.e., *GP-90* vs *GP-60* vs *KNN*) with respect to the case when only the manually-labeled points are used, increasing the number of automatically-labeled points. We observe that data augmentation does not always provide improvement in classification accuracy, especially for the RF case, when automatic labeling using encoded features always provides a deterioration of accuracy. The reason RF accuracy decreases is that, while autoencoders used to perform data augmentation transform the inputs of the classifiers into a "hidden" (i.e., "encoded") features space, RF-based classification is more similar to a "human" reasoning, where the outputs (i.e., the failure causes) are decided upon observation of real physical parameters, as the features $f_1 \div f_{35}$ used to perform manual labeling. This aspect is exacerbated for smaller training sets (e.g., for the case 50% $X_{L,train}$), when the number of automatically-labeled samples becomes prominent compared to the amount of data labeled manually).

On the other hand, some improvement can be observed for SVM and ANN cases. Specifically, for ANN, when using 100% of the training set (i.e., in the 100% $X_{L,train}$ case), the *GP-60* case provides around 1% accuracy improvement with respect to *GP-90*, and the *KNN* case provides up to 1.2% improvement. Similarly, for the SVM case, the *GP-60* case provides around 0.7% improvement with respect to *GP-90* case, which is comparable to the improvement provided with *KNN*-based automatic labeling. Overall, concerning ANN and SVM algorithms, there is at least one data augmentation scenario for which classification accuracy can be improved with respect to the case when only the manually-labeled dataset is used.

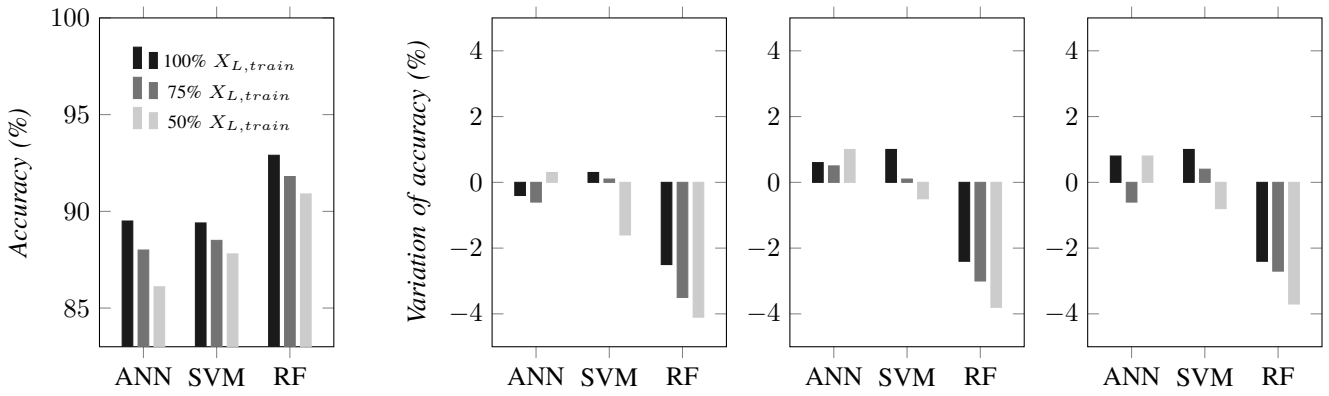
IX. Conclusion

We designed different ML-based classification algorithms to automate failure-cause identification in microwave networks. ML algorithms are trained with real data collected from a set of 10841 microwave links during a period of around 18 months. Considering a labeled dataset of 2513 points, representing periods of 45 minutes when microwave links are unavailable with the associated failure cause, we compare the performance of different ML algorithms, i.e., ANN, SVM and RF, in terms of classification accuracy and training duration.

[9]Note that, in any case, no significant difference has been observed by considering the different random sampling, as well as considering the different $X_{L,train}$-$X_{L,test}$ splitting.

[10]Note that, for each considered scenario, a different combination of hyperparameters can be obtained as the best autoencoder. Here we report results where, in each case, the specific best autoencoder is adopted.

[11]Note that results in Fig. 8(a) and 6 are similar, but they are not directly comparable as they refer to different test sets.



(a) Baseline (b) *GP-90* (c) *GP-60* (d) *KNN*

Fig. 8: (a) Classification accuracy for different ML algorithms and training set sizes (Baseline scenario with training set only and non-encoded features $f_1 \div f_{35}$). (b)-(c)-(d) Accuracy variation for the three data augmentation scenarios and encoded features Φ.

For all tested algorithms, satisfactory accuracy is obtained, reaching 93% when using RF, with limited training duration, which is in the order of few seconds for ANN and below 1 second for SVM and especially RF. Moreover, based on F1-score metric evaluated for RF classifier, we found that some failure causes, such as *Self-Interference* and *Hardware Failures*, are discriminated more easily with respect to the others, whereas in some cases, i.e., *Low Margin*, additional information might be needed to improve the failure-cause identification performance.

Furthermore, to make use of a large amount of additional unlabeled data, we also investigated 1) semi-supervised data augmentation based on autoencoders, which are used to identify a proper features space, and 2) how to automate the labeling of unlabeled data by adopting KNN and Gaussian Processes algorithms. We did not find significant accuracy improvement through data augmentation. In general, the performance improvement obtained using unlabeled data is limited or negligible, and it depends on the specific ML algorithm adopted for failure identification.

REFERENCES

[1] B. Shariati, M. Ruiz, J. Comellas, and L. Velasco, "Learning from the optical spectrum: Failure detection and identification," *Journal of Lightwave Technology*, vol. 37, no. 2, pp. 433–440, 2019.

[2] F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore, "An overview on application of machine learning techniques in optical networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1383–1408, 2018.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics New York, 2008.

[4] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo, "A coding approach to event correlation," in *International Symposium on Integrated Network Management*. Springer, 1995, pp. 266–277.

[5] H. Wietgrefe, K.-D. Tuchs, K. Jobmann, G. Carls, P. Fröhlich, W. Nejdl, and S. Steinfeld, "Using neural networks for alarm correlation in cellular phone networks," in *International Workshop on Applications of Neural Networks to Telecommunications (IWANNT)*. Citeseer, 1997, pp. 248–255.

[6] P. Szilágyi and S. Nováczki, "An automatic detection and diagnosis framework for mobile communication systems," *IEEE transactions on Network and Service Management*, vol. 9, no. 2, pp. 184–197, 2012.

[7] L. Pan, J. Zhang, P. P. Lee, M. Kalander, J. Ye, and P. Wang, "Proactive microwave link anomaly detection in cellular data networks," *Computer Networks*, vol. 167, p. 106969, 2020.

[8] P. Casas, P. Fiadino, and A. D'Alconzo, "Machine-learning based approaches for anomaly detection and classification in cellular networks." in *TMA*, 2016.

[9] F. Ahmed, J. Erman, Z. Ge, A. X. Liu, J. Wang, and H. Yan, "Detecting and localizing end-to-end performance degradation for cellular data services," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[10] J. Wu, P. P. Lee, Q. Li, L. Pan, and J. Zhang, "Cellpad: Detecting performance anomalies in cellular networks via regression analysis," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 2018, pp. 1–9.

[11] S. Varughese, D. Lippiatt, T. Richter, S. Tibuleac, and S. E. Ralph, "Identification of soft failures in optical links using low complexity anomaly detection," in *Optical Fiber Communication Conference*. Optical Society of America, 2019, pp. W2A–46.

[12] X. Chen, B. Li, R. Proietti, Z. Zhu, and S. B. Yoo, "Self-taught anomaly detection with hybrid unsupervised/supervised machine learning in optical networks," *Journal of Lightwave Technology*, vol. 37, no. 7, pp. 1742–1749, 2019.

[13] M. Furdek, C. Natalino, F. Lipp, D. Hock, A. Di Giglio, and M. Schiano, "Machine learning for optical network security monitoring: A practical perspective," *Journal of Lightwave Technology*, vol. preprint, 2020.

[14] M. Furdek, C. Natalino, M. Schiano, and A. Di Giglio, "Experiment-based detection of service disruption attacks in optical networks using data analytics and unsupervised learning," in *Metro and Data Center Optical Networks and Short-Reach Links II*, vol. 10946. International Society for Optics and Photonics, 2019, p. 109460D.

[15] X. Chen, B. Li, R. Proietti, Z. Zhu, and S. B. Yoo, "Self-taught anomaly detection with hybrid unsupervised/supervised machine learning in optical networks," *Journal of Lightwave Technology*, vol. 37, no. 7, pp. 1742–1749, 2019.

[16] X. Chen, B. Li, M. Shamsabardeh, R. Proietti, Z. Zhu, and S. Yoo, "On real-time and self-taught anomaly detection in optical networks using hybrid unsupervised/supervised learning," in *2018 European Conference on Optical Communication (ECOC)*. IEEE, 2018, pp. 1–3.

[17] F. Coenning, "Understanding itu-t error performance recommendations," https://www.julesbartow.com/Pictures/ITS/ITU-T_Errors_ApplicationNote2.pdf.

[18] "SIAE Microwave Product Portfolio," url=https://www.siaemic.com/index.php/products-services/telecommunication-systems/microwave-product-portfolio, accessed May 2020.

[19] G. E. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied artificial intelligence*, vol. 17, no. 5-6, pp. 519–533, 2003.

[20] S. Ali and K. Smith-Miles, *Improved Support Vector Machine Generalization Using Normalized Input Space*. Springer, Berlin, Heidelberg, 2006.

[21] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-supervised learning*. MIT Press, 2006.

[22] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.