Machine Learning for Failure Management in Microwave Networks: a Data-Centric Approach

Nicola Di Cicco*, *Graduate Student Member, IEEE*, Memedhe Ibrahimi*, *Member, IEEE*, Francesco Musumeci, *Senior Member, IEEE*, Federica Bruschetta, Michele Milano, Claudio Passera, and Massimo Tornatore, *Fellow, IEEE*

***Abstract*—We consider the problem of classifying hardware failures in microwave networks given a collection of alarms using Machine Learning (ML). While ML models have been shown to work extremely well on similar tasks, an ML model is, at most, as good as its training data. In microwave networks, building a good-quality dataset is significantly harder than training a good classifier: annotating data is a costly and time-consuming procedure. We, therefore, shift the perspective from a Model-Centric approach, i.e., how to train the best ML model from a given dataset, to a Data-Centric approach, i.e., how to make the best use of the data at our disposal. To this end, we explore two orthogonal Data-Centric approaches for hardware failure identification in microwave networks. At training time, we leverage synthetic data generation with Conditional Variational Autoencoders to cope with extreme data imbalance and ensure fair performance in all failure classes. At inference time, we leverage Batch Uncertainty-based Active Learning to guide the data annotation procedure of multiple concurrent domain-expert labelers and achieve the best possible classification performance with the smallest possible training dataset. Illustrative experimental results on a real-world dataset show that our Data-Centric approaches allow for training top-performing models with ∼4.5x less annotated data, while improving the classifier's F1-Score by ∼2.5% in a condition of extreme data scarcity. Finally, for the first time to the best of our knowledge, we make our dataset (curated by microwave industry experts) publicly available, aiming to foster research in data-driven failure management.**

***Index Terms*—Microwave Networks, Machine Learning, Failure Management**

I. Introduction

Machine Learning (ML) has found a broad set of applications in the design and management of modern communication networks [1]–[3]. Traditionally, failure management in microwave networks is dealt with via human inspection of telemetry data, which is an extremely time-consuming process. This requires hiring a team of trained domain experts who spend a significant amount of time identifying the causes of failure. In the specific context of microwave networks, modern ML-based approaches for failure management outperform static expert-driven decision rules, achieving close-to-human performance within a scalable decision-making framework [3].

*Nicola Di Cicco and Memedhe Ibrahimi are co-first authors.

Nicola Di Cicco, Memedhe Ibrahimi, Francesco Musumeci, and Massimo Tornatore are with the Department of Electronics, Information and Bioengineering (DEIB), Politecnico Di Milano, Italy. E-mail: {name}.{surname}@polimi.it

Federica Bruschetta, Michele Milano, and Claudio Passera are with SIAE Microelettronica, Cologno Monzese, Milan, Italy. E-mail: {name}.{surname}@siaemic.com

However, when training and deploying ML models for failure management, all network operators must face the challenge of building a good training dataset. For operators, the process of gathering and labeling data from the field is extremely expensive, as it requires continuous interaction with costly human annotators, causing significant time delays (e.g., for our dataset, it takes two full workdays for an annotator to label a hundred samples.)

Failure management in microwave networks is no exception to the aforementioned rule, as it requires *domain experts* to manually analyze and correlate the behavior of various link measures, i.e., Tx/Rx power and the corresponding failure alarms. Moreover, such experts must identify failure causes and failure types to take the *corresponding countermeasures* for restoring the disrupted services. Such countermeasures can imply something simpler, such as a decision to re-route traffic on another link until the quality-of-transmission in the original link is retrieved, or more complex decisions, such as sending a team in the field to identify and fix faulty equipment at a microwave site. While state-of-the-art ML-based approaches are known to automate such tasks with close-to-human accuracy, e.g., over 90% [3], the premise to reaching such performance is based on the assumption that there are sufficient labeled data for the ML model to learn robust failure identification rules.

In this work, in contrast to previous literature on failure management in microwave networks, we shift the perspective from *model-centric* (i.e., "How do we design a good ML model for failure identification?") to *data-centric* (i.e., "How do we build a dataset of good quality for training ML failure classifiers?"). We opt for a *data-centric* approach motivated by the fact that, in practice, improving the quality of a dataset brings greater benefits to an ML model than engineering its architecture and hyperparameters [4]. We, therefore, explore two data-centric approaches for hardware failure identification in microwave networks, i.e., identify a failure cause that occurs in the hardware component of a microwave network. Specifically, we investigate the following data-centric Research Questions (RQs), addressing two crucial domain-specific challenges of ML-based failure identification in microwave networks: a) high annotation costs, and b) data scarcity due to infrequent failures.

RQ1) (Online phase) What is the optimal strategy for collecting and annotating data? To answer this question, we propose an Uncertainty-based Active Learning (UAL) approach. The objective of UAL is to leverage the predictive

uncertainty of the model for labeling the most informative samples only. By doing so, we can identify a small subset of data to be labeled by domain experts, thus leading to significant cost savings without compromising the state-of-the-art classification performance of our ML model.

RQ2) (Offline phase) How can we overcome data scarcity from infrequent failure classes? To deal with this challenge, we leverage data augmentation with synthetic data. Specifically, we propose using a Conditional Variational Auto-encoders (CVAE) for generating synthetic samples with per-class, per-sample granularity. This is especially useful when the available dataset is highly imbalanced, i.e., in case a specific hardware failure presents itself only a few times during the data collection campaign. We illustrate that in a scenario characterized by extreme class imbalance, data augmentation with our CVAE ensures a fair classification performance for all hardware failure classes.

Last but not least, we make our dataset publicly available. To the best of our knowledge, ours is the first publicly available dataset comprising telemetry data from *real, in-production* microwave networks, collected over several years and painstakingly labeled by domain experts. We hope that our contribution will serve as a benchmark for future, exciting research on the microwave domain.

We summarize our key contributions as follows:

- We make a case for Data-Centric ML for failure management in microwave networks, and we focus on hardware failures (vs. mostly propagation failures in past work).
- We leverage the predictive uncertainty for quantifying the information gained by labeling batches of data points via Batch Uncertainty-based Active Learning (BUAL). Our approach allows training a top-performing model requiring only $\sim$15% of the full dataset, thus granting proportional savings in annotation times and costs.
- We leverage CVAEs for generating synthetic data to enrich poorly represented failure classes, with the goal of improving the average classification performance among all failure classes.
- We make our high-quality dataset publicly available[1]. Code is available upon request to the authors.

The remainder of the manuscript is organized as follows. In Section II, we discuss relevant related work. In Section III, we provide some background on microwave networks and describe the main characteristics of the dataset. In Section IV, we describe our proposed data-centric solutions and discuss how they fit in the context of failure management in microwave networks. In Section V, we discuss illustrative numerical results on our dataset. Section VI concludes the manuscript.

II. Related Work

We overview recent literature investigating applications of ML to wireless networks, with emphasis on microwave networks. Additionally, we describe several recent works applying Active Learning (AL) and synthetic data generation in wireless networks.

[1]https://github.com/bonsai-lab-polimi/tnsm2024-data-centric

A. ML for failure management in wireless networks

Failure management is a well-known problem in wireless networks, and it has been extensively investigated. Among the earliest works that address the problem of root-cause failure analysis in radio networks are [5] and [6]. Kliger *et al.* [5] have applied correlation techniques based on causality graphs and associate failure root causes to alarm sequence. Wietgrefe *et al.* [6] have adopted a neural network to correlate the presence of different alarms in mobile network equipment to an initial cause generating the alarm sequence. In [7], Szilágyi *et al.* designed a framework for automatic anomaly detection and cause identification in mobile networks, based on observations of alarms and employing a decision process that emulates human reasoning. Casas *et al.* [8] use decision trees for anomaly detection considering synthetically generated data drawn from realistic microwave network statistics. Ahmed *et al.* [9] used supervised learning to detect and localize failures in cellular networks, focusing on the observation of end-to-end service performance indicators. Wu *et al.* [10] proposed an anomaly detection framework considering time series for various performance indexes and applying a regression algorithm in cellular networks.

In recent years, our research group has actively investigated the application of ML-based approaches to microwave networks. Musumeci *et al.* [3] applied several ML approaches for automatic failure identification in microwave networks, in particular, proposing new models for the identification of failure causes by supervised ML and a semi-supervised automated labeling procedure based on auto-encoders. We address the problem of hardware failure as a supervised-learning classification problem as addressing similar problems in [3] via self-supervised pre-training was found to work overall poorly, therefore making the ground-truth information fundamental. Moreover, in microwave networks, different features can help capture different failure causes. For example, propagation failures can be discriminated from features associated with radio performance (e.g., transmitted/received power statistics and modulation format [3]), while equipment hardware failures can be discriminated from features associated with equipment alarms. Lateano *et al.* [11] and Ayoub *et al.* [12], [13] investigated several ML-based approaches for failure root-cause prediction and application of eXplainable Artificial Intelligence (XAI) in microwave networks, respectively.

Compared to previous model-centric works, we propose data-centric solutions allowing for cost-efficient data collection without compromising on the performance achieved in the state-of-the-art literature. Specifically, we leverage CVAEs for synthetic data generation and Batch AL to identify the most informative data points to label.

B. Active Learning applied to wireless networks

From an operator's point of view, gathering unlabeled data from the field is often a by-product of network monitoring. However, labeling said data (especially for hardware failures in microwave networks) can be an extremely costly and time-consuming process. To address the challenge of data labeling

with the objective of reducing operational cost and time, we leverage a Batch AL approach.

Pan *et al.* [14] make use of AL in microwave networks to continuously update the detection model at low cost. They show that, using AL, only 7% of the training data is required to achieve comparable results with the full dataset. Shahraki *et al.* [15] investigate the application of AL to query labels during training in the context of network traffic classification. Xu *et al.* [16] propose an AL-based solution to minimize the prediction error for classification-based mobile crowd-sensing subject to upload and query cost constraints.

In contrast to prior literature, we make a case for an AL framework leveraging the predictive uncertainty in tree ensembles. The motivation is twofold: first, tree ensembles are known to be the best-performing models in tabular datasets [17], [18]; second, tree ensembles allow for efficient epistemic (or knowledge) uncertainty sampling in AL, which is the current state-of-the-art for uncertainty sampling [19], [20]. Moreover, current AL applications in communication networks focus on querying one sample at a time, exacerbating annotation and training times. To solve this problem, we develop a Batch AL algorithm to simultaneously query up to 50 data points per iteration, showing minimal F1-Score degradation (~ 0.025 in the worst case) compared to the single-query optimum on the same annotation budget. This grants an order-of-magnitude reduction in annotation and model (re)-training times.

C. Synthetic data generation in wireless networks

A common challenge when applying ML to wireless networks is data scarcity, as it is often expensive to have access to operating networks, and, even when access is granted, it is expensive to gather and label data. Most existing research relies on mathematical models for data generation, however, such models are based on several assumptions and simplifications and may not fully represent realistic scenarios.

In the following, we cover related works that have applied Generative Adversarial Networks (GANs), Synthetic Minority Oversampling Technique (SMOTE), and Variational Auto-Encoders (VAEs) to address data scarcity.

Ayanoglu *et al.* [21] provide an overview of applying GANs to next-generation networks, and show several case studies of synthetic data generation. Navidan *et al.* [22] review the application of GANs to computer and communication networks, including mobile networks, network analysis, the internet of things, the physical layer, and cyber-security. Yang *et al.* [23] make use of GANs for autonomous wireless channel modeling without any domain-specific knowledge or technical expertise.

Clark IV *et al.* [24] investigate the problem of data augmentation in the context of radio frequency systems. Davaslioglu *et al.* [25] investigate the application of GANs to generate additional synthetic data to improve classifier accuracy and adapt training data to spectrum dynamics applied to spectrum sensing. Tang *et al.* [26] investigate the application of GANs to automated modulation classification in cognitive radio networks and evaluate the classifier's performance when the dataset is enriched with synthetically generated data.

An alternative, simpler approach for synthetic data generation is SMOTE. Massaoudi *et al.* [27] utilize it to handle class imbalance for intrusion detection in smart grid operations. Sun *et al.* [28] utilize SMOTE to address class imbalance in the context of self-organizing cellular networks to address the problem of failure diagnosis. Abdulkareem *et al.* [29] address class imbalance in an Internet-of-Things (IoT) context to perform a classification task for intrusion detection. Similarly, Tesfahun *et al.* [30] has implemented SMOTE to address class imbalance to perform an intrusion detection classification task using a Random Forest classifier.

Several works have used VAEs for generating synthetic data. Razghandi *et al.* [31] propose a VAE-GAN as a smart grid data generative model able to learn several types of data distributions and generate samples from the same distributions without performing any prior analysis on the data before the training phase. Qu *et al.* [32] propose a network data reinforcement method based on the multiclass VAEs to complete training tasks with a limited amount of data. The proposed solution can control the proportion of different classes by adjusting parameters, thus solving the imbalance problem in network data. Khan *et al.* [33] use VAEs for data augmentation for failure management in optical networks.

Compared to previous works that make use of other approaches for synthetic data generation, such as Conditional Tabular Generative Adversarial Networks (CTGANs) [34] and SMOTE [35], a CVAE-based solution proves to be more appropriate for our purpose. GAN-based solutions are unsuitable for our use case because i) GANs thrive in the presence of large volumes of data, but in our case, the initial dataset is of modest size (i.e., less than 1700 observations), and ii) GANs require a delicate balance between the generator and discriminator networks, and in small datasets this balance can be harder to achieve, leading to training instability, oscillations, and difficulties in convergence. We opted for VAEs as we empirically found them more reliable and easier to train than GANs for our application. We underline that the empirical studies of [36] illustrate different pros and cons of using VAEs as opposed to GANs for tabular data generation. VAEs achieve better performance metrics and are easier to train, but GANs make it easier to achieve differential privacy for the training data (which is presently not a concern in this work.) As for simpler approaches such as SMOTE, there are several shortcomings [37], such as: i) in case a minority class is extremely sparse, SMOTE could lead to overfitting and to a synthetic data generation that does not adequately represent the underlying distribution of the real data; ii) in high-dimensional spaces, SMOTE's effectiveness decreases as the curse of dimensionality makes it difficult to find meaningful neighbors for interpolation.

We do not claim that CVAE-based approaches are always better than other alternatives, e.g., GANs, as such choice depends on the case study. However, for our use case study, we show that adding synthetically generated data improves the per-class performance, especially for the under-represented failure classes. Compared to the related works for data generation, namely, CTGAN and SMOTE, we provide a qualitative comparison in the numerical results (see Section V).

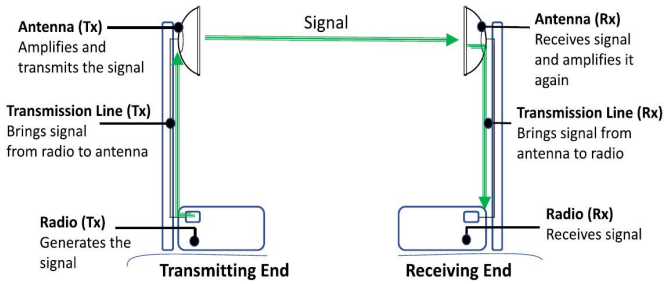


Fig. 1. Basic components of a microwave link [3]

III. Background on Microwave Networks and Dataset Description

In this Section, we provide a brief overview of the main components in a microwave network, and we describe the SIAE Microelettronica hardware failures dataset.

A. Microwave Networks

Figure 1 shows the basic structure of a microwave link, highlighting the transmitting end and the receiving end. Each end is composed of three main elements:

1) A Microwave radio that generates the signal (at the TX site) and receives the signal (at the RX site). The microwave radio can be placed at different locations, i.e., either inside a building or shelter (full-indoor), in the 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 an Indoor Unit (IDU).
2) A transmission line brings the signal from/to radio to/from the antenna. It is typically implemented via coaxial cables, suitable for frequencies up to around 2 GHz, or waveguides, used for higher frequencies 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 signal frequency, and may strongly affect the quality of transmission in case of physical medium deterioration.
3) A directional antenna, usually parabolic-shaped, is characterized by its gain, size, and directivity functions.

In this paper, we focus on hardware failures that can impact the hardware radio, i.e., IDU failure, ODU failure, cable failure, and power failure. In the following, we provide a detailed description of our hardware failures dataset.

B. SIAE Microelettronica hardware failures dataset

Microwave-link unavailability is typically defined in terms of *Unavailability Seconds* (UAS), representing the number of seconds over which the number of errored bits exceeds a given threshold. The UAS can be caused by various phenomena, such as *propagation failures*, due to, e.g., atmospheric factors such as rain, fog, temporary obstacles, or *hardware failures* due to equipment malfunction or aging. Hence, to automate the identification of failure causes leading to UAS and to avoid the continuous involvement of domain experts, we develop a set of ML-based solutions. To this end, we constructed a high-quality dataset of hardware failures from a real microwave network via a Network Management System developed by SIAE Microelettronica.

Our dataset comprises four non-overlapping classes of failure types: 1) *Class-0*: IDU failure, 2) *Class-1*: ODU failure, 3) *Class-2*: cable failure, and 4) *Class-3*: power failure. Each failure type is identified based on alarms issued by the radio equipment, serving as input features to the ML-based classifier.

We collected and labeled 1669 data points (observations) in total. The ground-truth labels were attributed and extensively validated by domain experts. Specifically, the frequency of each failure class in our dataset is as follows:

- Class-0 (IDU failure): 515 observations
- Class-1 (ODU failure): 611 observations
- Class-2 (Cable failure): 207 observations
- Class-3 (Power failure): 336 observations

The skew of our dataset, i.e., the ratio between the highest and the lowest class frequencies, is approximately 3, showing a moderate degree of class imbalance.

Each observation in the dataset aggregates data from 15-minute non-overlapping windows. For each microwave link and 15-minute window, the input to the classification problem consists of 164 features, each one corresponding to the number of seconds in which a specific alarm was active in the window. Specifically, the features take values between 0 and 900, with 0 indicating that the alarm is *OFF* and 900 indicating that the alarm is *ON* during the whole 15-minute window. To provide a rough idea of the human effort required for the manual data labeling, two domain experts from SIAE Microelettronica have spent more than 2 weeks[2] to label all 1669 data points.

To preserve confidentiality, we have performed the following operations to anonymize the dataset: 1) We masked all alarm names to [alarm_0 - alarm_{163}], 2) We removed all timestamps, 3) We randomly permuted the rows and columns of the original dataset (i.e., there is no semantic correlation between adjacent columns and no temporal correlation between adjacent rows), 4) We removed all sensitive information that could be used for tracing the specific microwave link.

For both the synthetic data generation and AL, we initially perform a *feature preprocessing* step, and then we estimate the performance of several ML models in terms of F1-score via stratified K-fold cross-validation.

C. Feature preprocessing

We devise three strategies for representing the alarm features based on when an alarm in a link is ON:

1) *Mode-1: Binary*, in which alarms are binarized, i.e., "0" if an alarm is OFF for the full 15-minute window, and "1" if the alarm is ON for at least 1 second.
2) *Mode-2: Categorical*, in which alarms are represented by categorical features.[3] We consider four categories: "0" if an alarm is OFF for the full 15-minute window, "1"

[2]For our specific task, domain experts were able to annotate approximately 50 data points for each full workday.

[3]Note that the identification of categories is done in collaboration with domain experts, to distill the most useful information for decision-making.

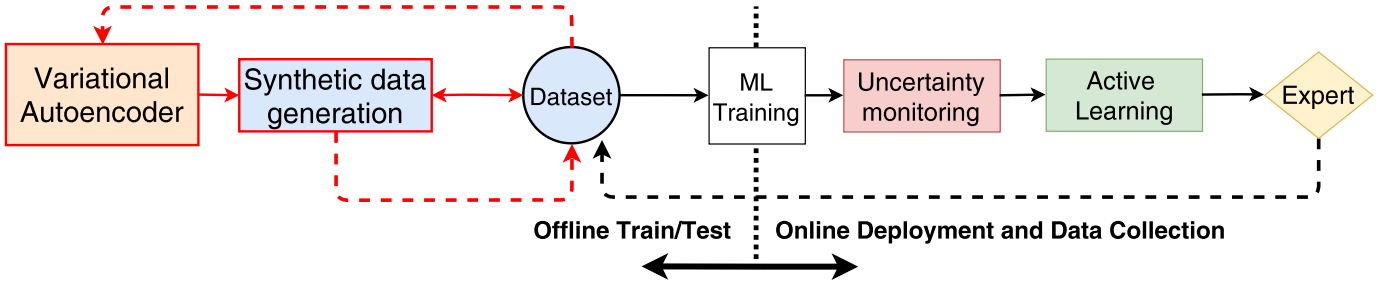


Fig. 2. Data-Centric ML for failure management in microwave networks. Synthetic data generation is performed in the *Offline Train/Test* phase, while AL is leveraged for the *Online Deployment and Data Collection* phase.

if an alarm is ON for no more than 45 seconds (low-severity alarm or false alarm), "2" if an alarm is ON between 45 seconds and 450 seconds (medium-severity alarm), and "3" if an alarm is ON between 450 seconds and 900 seconds (high-severity alarm).

3) *Mode-3: Inherent*, in which alarms take values between 0 and 900, as measured by the radio equipment.

In our numerical results, while we consider all three modes of data representation in the case of UAL, we consider only Mode-1 and Mode-2 features for synthetic data generation via the CVAE, as we have observed that Mode-3 (i.e., not applying any preprocessing) results in worse classification performance. Moreover, we have found the synthetic data generation process to be highly unreliable when generating data with continuous features, compared to either categorical or binary features. Overall, as a generic guideline for all the ML tasks considered in this work, binning the continuous alarm features turns out to be consistently beneficial.

IV. Data-Centric ML for Failure Management in Microwave Networks: Building Blocks

In Fig. 2 we illustrate the core elements in our Data-Centric ML solution for failure identification in microwave networks. Our solution comprises two sections: 1) Offline Train/Test, and 2) Online Deployment and Data Collection. Synthetic data generation is performed during the training phase when an offline dataset is already available. AL is performed during real-time inference, in order to guide the data labeling process towards the most informative samples.

In the following, we provide a detailed description of each building block. We first discuss UAL, as the microwave network we consider is currently operating, and then comment on synthetic data generation, as this process can be performed only once a dataset is available for offline training.

A. Uncertainty-based Active Learning

The goal of AL is to train an ML model with the fewest amount of labeled samples. Our core assumption is that large quantities of unlabeled data are available. This holds true for failure management in microwave networks, as collecting alarm sequences is fully automated and inexpensive. As mentioned before, annotating data in our application scenario is not only very costly but also time-expensive.

At a high level, AL allows the model to actively *query* the ground-truth label from unlabeled samples. The most crucial design choice in AL is defining a query strategy for selecting only the most informative samples.

A simple way to quantify the informativeness of an unlabeled alarm set is to estimate the predictive uncertainty associated with its predicted hardware failure class. Generally, the total predictive uncertainty can be decomposed into *Data Uncertainty* (DU) and *Knowledge Uncertainty* (KU) [38]. DU is caused by the inherent noise in the data, and therefore cannot be reduced by increasing the dataset size. On the other hand, KU is caused by the model's ignorance, i.e., by the model being presented with data from a different distribution than training. In contrast to DU, KU may be reduced by increasing the dataset size. Therefore, we are interested in leveraging KU as an "informativeness score" for labeling new samples.

While there are several strategies for uncertainty decomposition in ML, e.g., Bayesian Neural Networks, our previous works [3], [12] illustrated that the best-performing ML models for failure identification in microwave networks were ensembles of Decision Trees (DTs), such as Random Forests (RFs) or Gradient Boosting Decision Trees (GBDTs). Indeed, the superiority of tree-based models on small to medium-size tabular datasets has been validated by large-scale empirical studies [17], [18]. Therefore, we would like to seamlessly incorporate AL in our failure management framework while keeping our best models. For this reason, we leverage approximate uncertainty decomposition in ensemble models, such as RFs and GBDTs.

Formally, assume that we have trained an ensemble of M models on a dataset $\mathcal{D}$. Let $p(y|\mathbf{x},\mathcal{D})$ be the predictive distribution of the ensemble, $p(y|\mathbf{x},h^{(m)})$ be the predictive distribution of m-th model h in the ensemble (e.g., in RFs and GBDTs $h^{(m)}$ is a DT), and let $p(h|\mathcal{D})$ be the distribution of the ensemble members if trained on dataset $\mathcal{D}$ (e.g., in RFs and GBDTs this represents the probability distribution of generating a certain DT). x denotes the set of alarms while y denotes the failure class. We can define and estimate the KU in Eq. (1), as follows:

$$\underbrace{\mathcal{I}(y,h|\mathbf{x},\mathcal{D})}_{\text{Knowledge Unc.}} = \underbrace{\mathcal{H}\left[p(y|\mathbf{x},\mathcal{D})\right]}_{\text{Total Unc.}} - \underbrace{\mathbb{E}_{p(h|\mathcal{D})}\mathcal{H}\left[p(y|\mathbf{x},h)\right]}_{\text{Data Unc.}} \tag{1}$$
$$\approx \mathcal{H}\left[\frac{1}{M}\sum_{m=1}^{M} p(y|\mathbf{x},h^{(m)})\right] - \frac{1}{M}\sum_{m=1}^{M}\mathcal{H}\left[p(y,|\mathbf{x},h^{(m)})\right],$$

where $H[\cdot]$ and $\mathcal{I}[\cdot,\cdot]$ denote Shannon entropy and Information Gain, respectively [38]. The *Total Uncertainty* (TU) is defined

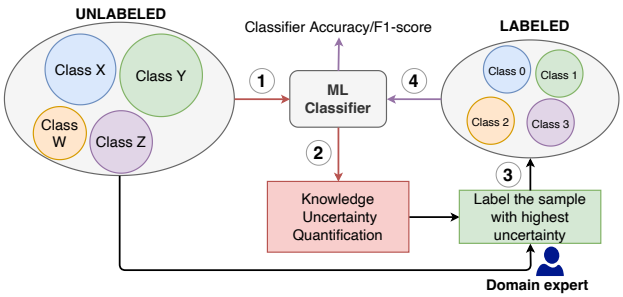


Fig. 3. Single-query Uncertainty-based Active Learning (UAL)

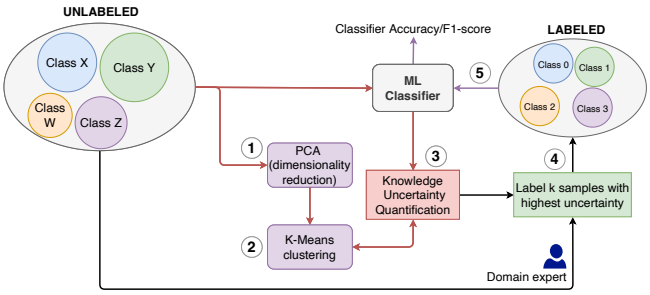


Fig. 4. Batch Uncertainty-based Active Learning (BUAL)

as the entropy of the predictive distribution of the ensemble, obtained by averaging between all ensemble members.[4] The DU is defined as the expected entropy value of the single ensemble member. Finally, the KU is defined as the information that model h gains by revealing the failure class y associated with the alarm set $\mathbf{x}$, and is estimated by subtracting the DU from the TU.

We develop two approaches based on the policy for querying data points to label: i) single-query UAL), and ii) multi-query BUAL. Figure 3 and Fig. 4 illustrate the application of UAL and BUAL, respectively, for guiding the data collection and labeling procedure in our failure management framework. We assume a small initialization pool of labeled data, e.g., 20-40 labeled alarm sets, and a large pool of unlabeled data, e.g., 1000-1500 alarm sets. In the following, we describe the workflow of UAL and BUAL.

B. Single-query Uncertainty-based Active Learning

The framework operates in four steps (see Fig. 3): (1) Data from the pool of *UNLABELED* data are queried via the *ML classifier* which then does (2) *Uncertainty Quantification* of data points in terms of KU. The alarm set with the highest KU values is forwarded to a domain expert to be labeled (3) *Label the sample with highest KU*, which is then added to the pool of labeled alarms. Note that in the case of single-query UAL, one sample is labeled and added to the pool of *LABELED* data. The ML model is retrained from scratch on the updated dataset (4), and the process is iterated until a stopping condition is reached (e.g., a pre-defined budget of data points to label is met or a pre-defined performance metric, such as F1-Score, is met on a held-out validation dataset). In our case, we consider a fixed budget of data points, which can be related to a limited financial budget for compensating domain experts and/or to a time constraint on data collection and labeling.

C. Batch Uncertainty-based Active Learning

In principle, UAL assumes adding one data point per iteration, which guarantees that, at each iteration, we are sampling the alarm set that provides the highest expected information gain for our model. We now consider a Batch AL scenario, where multiple data points are labeled per AL iteration. There are two main practical reasons motivating the need for Batch AL over single-query AL: 1) in real-world scenarios, it is often the case that multiple domain experts are dedicated to data annotation; 2) when annotating one data point per iteration, it is necessary to re-train the ML classifier for each newly-labeled data point. Instead, Batch AL re-trains the ML classifier only once per batch of X data points, e.g., X = 50, granting a proportional reduction in training times. We therefore propose BUAL to solve the aforementioned issues.

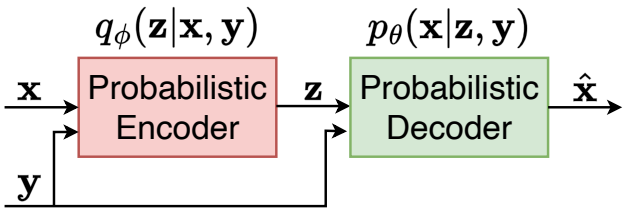
A first, naive query strategy for BUAL is to take the top K data points with the highest KU. We refer to this simple approach as *BUAL-TopK*. However, the top K data points with the highest KU might be highly correlated: for instance, revealing the ground truth for one point only would render all the remaining $K-1$ not very informative. Indeed, our numerical results in Section V-B illustrate that BUAL-TopK, though performing much better than random sampling, is considerably less data-efficient than single-query UAL.

Therefore, we aim for a Batch AL algorithm able to sample points that are both i) highly informative and ii) sufficiently diverse. To balance this delicate trade-off, we leverage dimensionality reduction and K-means clustering to develop *BUAL-KMeans*, which, unlike BUAL-TopK, allows multiple simultaneous queries (up to 50) with negligible performance degradation with respect to single-query UAL.

The core design choice in BUAL-KMeans is to leverage K-means clustering [39] to avoid sampling redundant data points. Previous literature in Batch AL with neural networks illustrated that clustering is an effective strategy for balancing diversification and informativeness [40], [41]. Unfortunately, neural networks are a bad fit for our problem, as it is formulated on tabular data [17], [18]. Moreover, given the relatively high dimensionality of our dataset (164 features) and its moderate size (1669 observations) with respect to the large-scale benchmark datasets in deep learning, distance-based clustering algorithms such as K-means cannot be used out-of-the-box for our problem.

To address these issues, we apply a dimensionality-reduction step to the unlabeled data using Principal Component Analysis (PCA) [42] and keeping only the first three dimensions with the highest variance. This allows filtering out small pairwise distances, enabling effective data separation via K-means clustering. Note that, even though the dimensionality reduction causes significant information loss, the compressed features are used only for clustering, whereas the ML classifier is presented with the full set of alarm features.

[4]In ensemble learning, the KU can also be interpreted as the level of disagreement between different ensemble members.



$$\mathcal{L}_{\text{CVAE}} = -\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) \,||\, p(\mathbf{z}|\mathbf{y}))$$

Fig. 5. Conditional Variational Autoencoder (CVAE). The one-hot encoded hardware failure class vector $\mathbf{y}$ conditions both the encoder and the decoder.

Putting everything together, the BUAL-KMeans algorithm operates in five steps (see Fig. 4): (1) Data from the pool of *UNLABELED* data are fed to the *PCA* block for dimensionality reduction, and then, in step (2), the unlabeled data are clustered via *K-Means Clustering*. In step (3), the *KU Quantification* estimates the information gain through the *ML classifier* for all data points in the pool of *UNLABELED* data. The decision on which data points to pass on to the domain experts is taken based on K-Means Clustering, such that the data point with the highest KU from each cluster is passed on, e.g., we set K = 10 clusters for selecting 10 data points to be labeled in one iteration. In step (4), the batch of alarm set with the highest KU values is forwarded to domain experts for labeling (*Label k samples with highest uncertainty*) and then added to the pool of *LABELED* data. Similarly to single-query UAL, in step (5), the ML model is retrained from scratch on the updated dataset, and the process is iterated until a stopping condition is reached.

D. Synthetic Data Generation with CVAEs

VAEs are deep generative neural networks that can efficiently learn probabilistic models of high-dimensional data [43]. A VAE comprises a probabilistic encoder network and a probabilistic decoder network. The encoder $q_\phi(\mathbf{z}|\mathbf{x})$, parameterized by ϕ, maps an input data sample to a latent space. The decoder $p_\theta(\mathbf{x}|\mathbf{z})$, parameterized by θ, maps a sample from the latent space to the input space. VAEs are trained by minimizing the Evidence Lower Bound (ELBO) loss, given by Eq. (2), as follows:

$$\text{ELBO} = \underbrace{-\log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{MLE reconstruction}} + \underbrace{D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z}))}_{\text{KL regularization}}. \quad (2)$$

The ELBO loss comprises a Maximum Likelihood Estimation (MLE) reconstruction term and a Kullback-Leibler (KL) regularization term [43]. The MLE reconstruction term ensures that the samples reconstructed by the decoder are faithful to the original distribution of the input data. The regularization term forces the variational posterior to be as close as possible to the true posterior. In practice, the KL regularization term forces the latent space to assume the shape of the prior distribution. In this way, after the VAE is trained, one can efficiently generate synthetic data by i) sampling $\mathbf{z}$ vectors from the prior distribution and ii) feeding the $\mathbf{z}$ samples to the probabilistic decoder to produce synthetic $\mathbf{x}$ samples.

While one could use vanilla VAE to generate data, we are interested in sampling from the class-conditioned distribution of the input data. In our application scenario, we would like to have fine-grained control over the number of samples we generate for each failure class, as some classes may be severely underrepresented. While sampling from a specific class could be trivially achieved by training a vanilla VAE and performing rejection sampling, there are two major drawbacks. First, said methodology can become computationally expensive if large amounts of per-class data are queried and/or some classes are less represented in the input data, and are therefore generated less frequently. Secondly, it is well-known that VAEs suffer from mode collapse, that is, the model generates repetitive samples that do not capture the full diversity of the training data. Specifically, in our application scenario, we are interested in modeling 1) the diversity between alarms signaling *different* hardware failure classes, and 2) the diversity of alarms signaling *the same* hardware failure class.

To solve these problems, we employ CVAEs, which learn a probabilistic model of the input data conditioned on some contextual information (in our case, the class information). This is easily achieved by supplying a one-hot vector $\mathbf{y}$ encoding the hardware failure class to both the encoder and the decoder networks. The CVAE ELBO loss [44] then becomes as given by Eq. (3), as follows:

$$\text{ELBO} = \underbrace{-\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y})}_{\text{MLE reconstruction}} + \underbrace{D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) \,||\, p(\mathbf{z}))}_{\text{KL regularization}}. \quad (3)$$

Similarly as before, to generate new data i) we sample $\mathbf{z}$ vectors from the prior distribution, ii) we feed the $\mathbf{z}$ samples and a class vector $\mathbf{y}$ to the probabilistic decoder. In this way, we achieve fine-grained control on the exact per-class quantity of synthetic data we generate with no additional computational effort. Moreover, as the class information is supplied to the model, the CVAE must only model the diversity between samples in the same class, which is arguably an easier learning task compared to a standard VAE.

Figure 5 shows the building blocks of the CVAE used for augmenting our hardware failures dataset, highlighting that *Probabilistic Encoder* and *Probabilistic Decoder* are conditioned on the one-hot encoded class $\mathbf{y}$.

V. Illustrative Numerical Results

In this Section, we first describe the evaluation settings and then discuss our illustrative numerical results for AL and synthetic data generation.

A. Evaluation settings

Here, we discuss our numerical evaluations of the proposed Data-Centric methodologies for hardware failure identification in microwave networks.

We report macro-averaged F1-score (hereafter referred to as macro F1-score) as the main performance metric, as we are interested in fair performance for all classes.[5] As UAL requires

[5]In our simulations, we consider the macro-averaged F1-score, which is computed using the arithmetic mean of all per-class F1-scores. This metric treats all classes equally regardless of their number of data points in the dataset.

retraining the model after a single query, we also report the model training times. Our experiments have been conducted on a Linux PC with an Intel Core i7-6700 processor and 32 GB RAM. Our ML algorithms were implemented using Scikit-Learn [45] for RFs and XGBoost [46] for GBDTs. We opted to use XGBoost and RFs for our application because they have been found to be among the best-performing models for tabular data [17], [18].

B. Uncertainty-based Active Learning

We consider three modes of alarm feature representation and three sizes of the initial dataset.

- Alarm representations are described in Section III, and are: 1) Mode-1: Binary, 2) Mode-2: Categorical, and 3) Mode-3: Inherent.
- We consider three cases of initially labeled datasets: 1) 25 data points, 50 data points, and 3) 100 data points. The number of data points per class in the initially labeled dataset reflects the percentage of each hardware failure class in the complete dataset, i.e., Class-2 comprises 12% of the total dataset, so in 100 data points, 12 data points belong to Class-2.

We evaluate the classification performance in terms of macro F1-score using stratified K-fold cross-validation with K = 5. We assume an available budget for labeling 500 data points, which, in our practical experience, translates to a whole week of work for two domain experts.

Note that all numerical evaluations for data labeling, i.e., the proposed UAL and all baselines, are conducted considering the original dataset described in Section III-B, with no additional synthetically generated data involved.

For Batch AL, we consider querying i) 1 sample, ii) 10 samples, iii) 25 samples, and iv) 50 samples per AL iteration.

We consider three baselines: 1) Virtual Ensembles in Gradient Boosted Decision Tree (GBDT-VE) models as a representative state-of-the-art baseline [38] for uncertainty quantification and Active Learning on tabular datasets. GBDT-VE constructs a "virtual ensemble" by aggregating the predictions of multiple sub-models that compose a single GBDT model. The sub-models are constructed via truncation, i.e., by removing trees from the starting model. In contrast to RFs, where trees are trained independently, GBDT-VE has a high degree of correlation between trees, which adversely affects its uncertainty estimations. 2) Random sampling, i.e., data points to be queried are selected randomly, corresponding to a standard labeling strategy without AL, and 3) Full-Training, i.e., the performance of the best ML classifier when training on the full dataset as an upper bound on the best performance a ML model can achieve. For our problem, RFs and GBDTs roughly achieve the same level of performance when trained on the full dataset.

1) UAL vs. Baselines: Figures 6a, 6b and 6c show results for the RF classifier in terms of macro F1-score for Mode-1 with K-queries per AL iteration, where K = 1, 10, 25 and 50.

We observe that single-query UAL meets the Full training F1-score by querying less than 200 data points, compared to the approximately 1400 data points used for Full training. In other words, UAL would require two domain experts to work for two days instead of two weeks. These remarkable time (and hence cost) savings have proven to be very significant for our industrial collaborators for a timely and cost-efficient deployment of the proposed ML-based failure management framework. Moreover, single-query UAL always outperforms GBDT-VE and Random Sampling, the latter being unable to meet the Full training F1-score even after labeling 500 additional data points to the training set. This highlights the practical effectiveness of AL for our application scenario.

In the Batch AL setting, we observe that BUAL-TopK, while outperforming random sampling, is significantly less data-efficient than single-query UAL and GBDT-VE. This is because BUAL-TopK does not consider diversifying the data in the batch. Conversely, BUAL-KMeans follows almost perfectly the trend of single-query UAL for batch sizes K=10 and K=25, while performing comparably to single-query GBDT-VE for K=50.

The practical advantages of BUAL over single-query UAL are exemplified in Fig. 6d, which shows the F1-score as a function of the AL iterations for a batch size K=50. We observe that BUAL-KMeans and BUAL-TopK reach the F1-score of Full training with at most 10 AL iterations, whereas single-query UAL requires around $\sim$25x more AL iterations to reach the same F1-score. In practical terms, the cost savings of BUAL-KMeans over single-query UAL are twofold: first, if multiple domain experts are available, the annotation times can be reduced proportionally to the batch size; second, BUAL-KMeans drastically reduces the number of times the model needs to be re-trained from scratch.

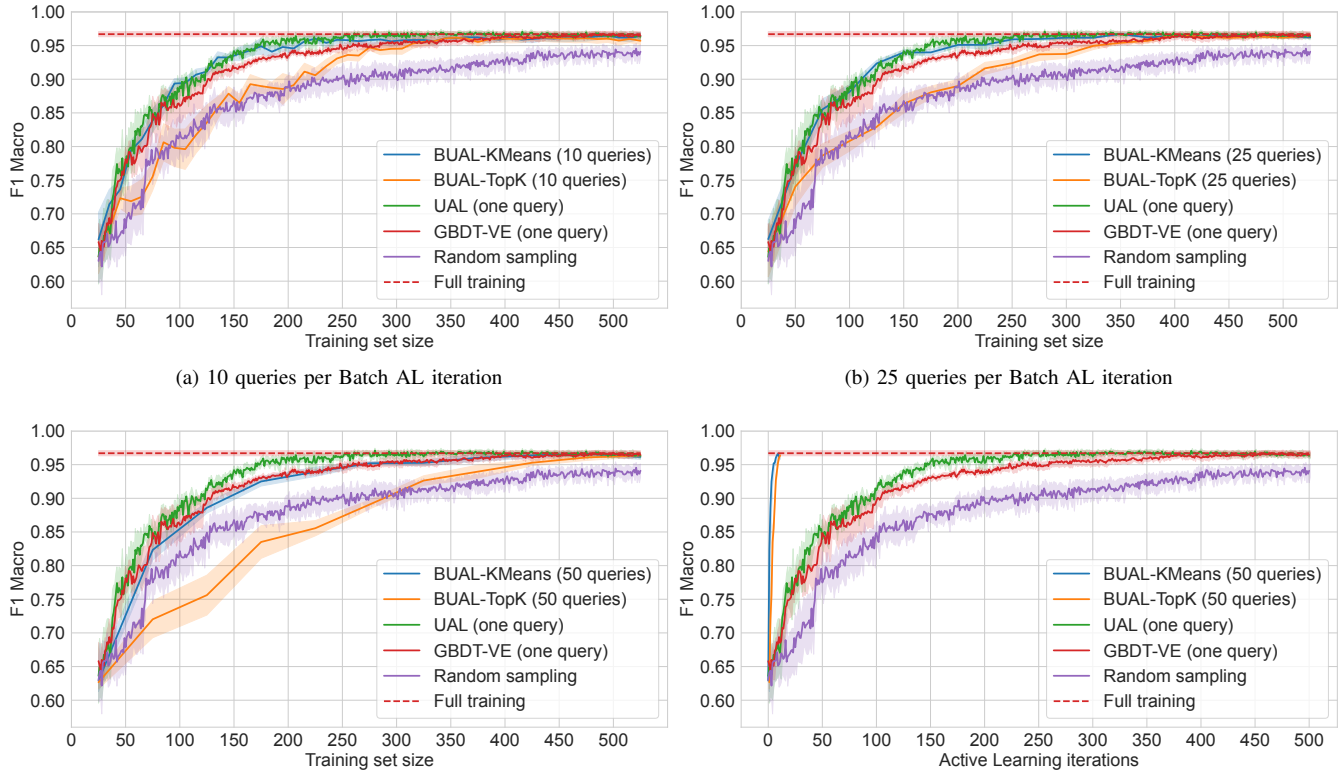
We have performed the same analysis for the other modes of alarm features, namely, Mode-2 (Categorical) and Mode-3 (Inherent), and the main takeaways are consistent with the results shown above. Similarly, the same takeaways highlighted in previous paragraphs hold in case the initially labeled data set has 50 data points and 100 data points.

2) Training times: In the following, we briefly discuss the training times of single-query UAL while varying: 1) the two ML classifiers considered in our work, i.e., RF vs GBDT, and 2) alarm feature modes, i.e., Binary vs Categorical vs Inherent.

We report the training time in the case of UAL single-query, i.e., querying one data point per UAL iteration, as we observed that Batch AL with batch size K follows roughly a proportional reduction in training time by $\frac{1}{K}$. Batch AL also includes PCA and clustering, even though their contribution to the total training time is negligible. Note that, for a scenario with a budget of 500 data points to be labeled, there are 500 iterations of the AL framework.

TABLE I
SINGLE-QUERY UAL TOTAL TRAINING TIMES AS A FUNCTION OF I) CLASSIFIER TYPE, AND II) ALARM MODE REPRESENTATION.

Classifier	Alarm mode	Training time (mins)
GBDT	**BINARY**	420
	CATEGORICAL	179
	INHERENT	234
RF	**BINARY**	19
	CATEGORICAL	19
	INHERENT	19



(a) 10 queries per Batch AL iteration

(b) 25 queries per Batch AL iteration

(c) 50 queries per Batch AL iteration

(d) 50 queries per Batch AL iteration

Fig. 6. F1-score of an RF classifier for our proposed AL strategies, UAL and BUAL, against GBDT-VE, Random sampling, and Full training. (a)-(c): F1-score while varying the number of concurrent AL queries. BUAL-KMeans performs nearly as well as single-query UAL. (d): F1-score as a function of the number of AL iterations for 50 queries per batch. BUAL requires very few AL iterations to reach the Full training upper bound.

Table I compares the training time for GBDT and RF. We observe that RFs are significantly more computationally efficient than GBDTs (19 vs. 420 minutes). This is because in RFs the KU can be computed considering each tree in the ensemble, while with GBDTs one needs to train multiple GBDT models, due to the fact that boosted trees are too highly correlated. Though GBDT-VE is more computationally efficient than GBDT ensembles in computing KU, we demonstrated that, for our use case, its AL performance is generally inferior compared to our RFs.

Regarding alarm modes, we notice that the training time of RFs is similar regardless of the alarm mode, while in the case of GBDT we observe that the training times are highly dependent on the chosen alarm representation. Specifically, the Categorical mode has the lowest training time, followed by the Inherent and Binary modes. This behavior is purely dependent on the implementation details of XGBoost, especially on the tree-growth policy. A possible explanation could be that, since Binary features are less informative than Categorical features, XGBoost tends to grow more trees to their maximum allowed depth (6 by default), resulting in longer training times.

Finally, we consider that a total training time of a few tens of minutes is reasonable in practice, as SIAE Microelettronica has adopted our solution in the field.

C. Synthetic data generation

We now illustrate our numerical results for synthetic data generation. To simulate a scenario of extreme data scarcity, we randomly remove 80% of the data points from the training set for Class-2 (i.e., the least represented class), leading to an extreme imbalance in the class representation. This mimics a practical scenario of a rare occurrence of specific failures during the data collection campaign. The goal of this analysis is to show that, even in the case of a highly imbalanced dataset, data augmentation with CVAE allows for a fair classification performance for all hardware failure classes. We consider three dataset rebalancing strategies via CVAE:

1) **Balanced**: starting from an unbalanced scenario, generates synthetic data to ensure an equal number of data points for all classes.
2) **Balanced x2**: doubles the number of data points once having balanced the number of data points for all classes.
3) **Fixed**: generates a fixed number of synthetic data for each class, regardless of their percentage in the total dataset. We consider two cases of adding a fixed number of synthetic data: 1) add 150 data points to each class, and 2) add 400 data points to each class.

We compare the following models: 1) **Base model**: the ML model is trained on real data only, and 2) **Mix model**: the ML model is trained augmenting the real data with synthetic data. For the sake of brevity, we only represent results with Mode-1: Binary alarms. Similar considerations can be drawn for the Categorical representation of alarms.

Note that as *Base* considers training on real data only, the size of the training set is always equal to 1204 data points. On the other hand, the total number of training data in *Mix*

TABLE II
SYNTHETIC DATA GENERATION: ACCURACY, F1-SCORE, PRECISION AND RECALL FOR **BASE** AND **MIX** MODELS COMPARING CVAE, SMOTE AND CTGAN. FOR CVAE, WE REPORT FOUR CASES OF DATA GENERATION: I) BALANCED, II) BALANCED X2, III) FIXED 150, AND IV) FIXED 400. WE REPORT MEANS AND STANDARD ERRORS OVER 5-FOLD CV.

Metric	Base	Balanced	Balanced x 2	Fixed 150	Fixed 400	SMOTE	CTGAN
Accuracy	91.49 (1.03)	93.05 (0.81)	92.39 (0.89)	92.45 (1.01)	92.75 (1.34)	90.71 (1.03)	91.73 (1.11)
F1-score	88.13 (1.31)	91.01 (1.02)	89.97 (0.96)	90.08 (1.31)	90.38 (1.72)	86.48 (1.35)	88.85 (1.46)
Precision	92.57 (1.13)	92.91 (0.87)	92.16 (1.05)	92.74 (1.27)	92.98 (1.54)	90.97 (1.14)	91.23 (1.17)
Recall	86.40 (1.32)	89.85 (1.12)	88.78 (0.94)	88.69 (1.29)	89.00 (1.78)	85.06 (1.31)	87.77 (1.48)

varies depending on the amount of synthetic data added. In particular, for *Balanced*, the total number of training data is 1956 data points, for *Balanced x 2*, the total number of training data is 3904 data points, for *Fixed 150*, the total number of training data is 1804 data points, and for *Fixed 400*, the total number of training data is 2804 data points.

We consider two state-of-the-art baselines, namely, CTGAN and SMOTE, that add synthetic data to the training set and balance the per-class representation. Therefore, in terms of the number of data points in the training set, they correspond to the *Balanced* scenario.

We compare all approaches (CVAE, SMOTE, and CTGAN) in case of *Base* and *Mix* in terms of i) Accuracy, ii) macro F1-score, iii) Precision, and iv) Recall (see Table II), and per-class F1-score (Table III) to quantify the benefit of augmenting real data with synthetic data. We report means and standard errors over 5-fold cross-validation.

In the following, we first compare the different flavors of CVAE, i.e., *Balanced*, *Balanced x2*, *Fixed 150*, and *Fixed 400*. We then compare the best-performing CVAE to two state-of-the-art baselines: CTGAN and SMOTE.

1) CVAE for synthetic data generation: Table II shows that *Mix* outperforms *Base* across all metrics, for all scenarios of CVAE. The main takeaway from this observation is that CVAE improves the classifier's performance by adding synthetically generated data to the training set. In the following, we comment the numerical results in greater detail.

F1-score. Comparing the various scenarios of data augmentation with CVAE, we observe that *Balanced* (ensuring all classes are equally represented) achieves the best performance, e.g., improves the classifier's F1-score by 2.88% in case of *Mix* compared to *Base*.

To illustrate the impact of this improvement, we quantify it in the absolute number of number of correct classifications. For example, in a scenario of 50 microwave links and an alarm report every 15 minutes, there is an upper bound of 72000 samples to be classified daily. Even though it is expected that not every 15-minute window will report an alarm, in practice, such improvement in the F1-score leads to hundreds of additional correct classifications. From an industrial perspective, this translates to significant savings in the maintenance costs.

In contrast, we observe that when adding an excessive number of synthetic data, e.g., *Balanced x 2*, the model will perform worse than *Balanced*. This illustrates that adding synthetic data to the training set does not necessarily improve the generalization performance on the test set. Indeed, while extending a dataset with new real measurements is always beneficial, there is no a-priori guarantee that adding synthetic data will improve the performance of an ML model.

Let us now focus in Table III on the *per-class* F1-score for the *Base* and the *Mix* models. We consider that the per-class F1-score allows us to quantify the impact of adding synthetic data for each class.

Class-2. Observing the F1-score of Class-2 is particularly important, as in this class we removed 80% of training data points, simulating a scenario of extreme data scarcity. Indeed, data scarcity in Class-2 is reflected in the F1-score of the *Base* model, with an F1-score less than 73%. Adding synthetically generated data significantly improves the F1-score by up to 8%. Comparing the various synthetic data generation strategies, we observe that balancing the dataset so that all classes are equally represented in the training set leads to the best performance.

Similarly, for other classes (*Class-0*, *Class-1*, and *Class-3*), we observe that the *Mix* model outperforms the *Base* model augmenting real data with synthetic data.

We conducted the same analysis in the case of Categorical alarms and concluded that data augmentation improves the F1-score of the *Mix* model by around 3% compared to the *Base* model. For brevity, such results are not included in the paper.

We conclude that balancing the representation of classes in the training dataset, i.e., CVAE in the case of *Balanced*, leads to the best test set performance. Therefore, we compare this scenario to state-of-the-art CTGAN and SMOTE that also re-balance the training dataset, i.e., all classes have the same number of data points.

2) CVAE vs Baselines: We compare *Balanced* in the case of CVAE with SMOTE and CTGAN and make the following observations. First, we observe the impact of adding synthetically generated data to the training set by comparing *Mix* and *Base* in the case of SMOTE. Table II shows that *Base* outperforms *Mix* across all metrics, e.g., *Mix* has a better F1-score by almost 2% compared to *Base*. Moreover, looking into the per-class F1-score, we notice that, especially for the underrepresented Class-2, adding synthetically data via SMOTE actually degrades the F1-score by 5% (from 73% to 68%). The takeaway of these numerical evaluations is that adding *synthetic* data that does not truthfully represent the *real* data may lead to performance degradation in the test set. This proves that having more data in the training dataset (in a scenario when real data are augmented with synthetically generated data) does not necessarily improve the classifier's generalization performance. Finally, comparing CVAE to SMOTE, we observe that CVAE is significantly

TABLE III
PER-CLASS F1-SCORE FOR **BASE** AND **MIX** MODELS FOR CVAE, SMOTE AND CTGAN. FOR CVAE, WE REPORT FOUR CASES OF DATA GENERATION: I) BALANCED, II) BALANCED X2, III) FIXED 150, AND IV) FIXED 400. WE REPORT MEANS AND STANDARD ERRORS OVER 5-FOLD CV.

F1-score	Base	Balanced	Balanced x 2	Fixed 150	Fixed 400	SMOTE	CTGAN
Class-0	96.13 (0.76)	96.32 (0.78)	96.32 (0.96)	95.73 (0.95)	96.20 (0.95)	96.05 (0.92)	95.61 (1.01)
Class-1	93.31 (1.03)	94.12 (0.73)	93.63 (1.21)	93.67 (0.86)	93.79 (1.14)	93.18 (1.09)	93.88 (1.15)
Class-2	72.99 (2.51)	80.95 (2.45)	78.14 (1.82)	78.49 (3.01)	78.69 (3.47)	67.54 (2.79)	75.61 (3.70)
Class-3	90.09 (1.49)	92.64 (0.89)	91.79 (0.82)	92.42 (1.17)	92.85 (1.68)	89.14 (1.25)	90.29 (1.38)

better, as it always improves classifier performance by adding synthetically generated data to the training set.

Second, in the case of CTGAN, we notice that the classifier's performance when adding synthetic data, i.e., *Mix* scenario, is either similar or slightly better than when only real data are used, i.e., *Base* scenario. In particular, in Table II, we observe an improvement of 0.24% in Accuracy and 0.72% in F1-score. In terms of the per-class F1-score, in Table III, we observe that for Class-2 (being the least represented class), CTGAN ensures an F1-score improvement by 2.62% when the classifier is trained with mixed (real+synthetic) data compared to when the classifier is trained with real data only. For the other classes, the performance is comparable, e.g., CTGAN ensures an improvement by 0.57% in the case of Class-1. We clearly see that CTGAN outperforms SMOTE, as it preserves or improves the classifier's performance when adding synthetic data to the training set. However, the performance improvement of CTGAN is inferior compared to CVAE, as CVAE improves the F1-score by 8% in the case of Class-2.

VI. CONCLUSION AND FUTURE WORK

Motivated by the high operational costs required for collecting and annotating data, we proposed two Data-Centric approaches for ML-based hardware-failure identification in microwave networks: offline dataset augmentation with CVAE, and online data collection guided by BUAL.

Our main takeaway is that Data-Centric approaches are mandatory for ensuring time and cost-efficient deployment of robust ML models for failure management in microwave networks. Quantitatively, we have demonstrated that BUAL achieves the best-known level of test-set performance with 4.5x fewer training samples, which translates into proportional savings in terms of manual labeling time and waiting time before deployment (three days vs. two weeks). Moreover, we illustrated that synthetically augmenting an offline dataset with CVAE-generated synthetic data improves the F1-score by 2.5% in a scenario characterized by extreme data scarcity (only a few tens of samples for the least-represented class).

We note that for other applications, e.g., when other microwave vendors and operators are considered, our proposed solution might not necessarily work as a '*black-box solution*', and further adjustment might be needed. For instance, in scenarios of data abundance, e.g., over 100k data points, the PCA decomposition and the repeated KMeans clustering in BUAL might not scale very well. Moreover, for synthetic data generation, large datasets would make it feasible to leverage complex CTGANs instead of our CVAE. Nevertheless, we argue that our empirical findings can be practically useful for many problems in ML for communications networks where deployment costs are dominated by data annotation.

Possible future research directions leveraging this dataset include the development of Online Learning approaches by considering the hardware failure monitoring data as a time series. The goal would be to build real-time processing pipelines for monitoring data and promptly flagging any unknown, anomalous behavior. As a result, even in case of unexpected failure events, the time-to-repair will be kept to a minimum.

Lastly, we plan to make public, for the first time to the best of our knowledge, a high-quality dataset for failure identification in microwave networks collected from a real, currently operating microwave network. We hope that our contribution will open up new research directions in the field of ML for microwave failure management, and will serve as a benchmark dataset for future data-driven failure management frameworks.

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, 2019.

[3] F. Musumeci *et al.*, "Supervised and semi-supervised learning for failure identification in microwave networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1934–1945, 2021.

[4] D. Zha, K.-H. Lai, F. Yang, N. Zou, H. Gao, and X. Hu, "Data-centric ai: Techniques and future perspectives," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 5839–5840.

[5] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo, *A Coding Approach to Event Correlation*. Boston, MA: Springer US, 1995, pp. 266–277.

[6] H. Wietgrefe *et al.*, "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.

[7] P. Szilagyi and S. Novaczki, "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.

[8] P. Casas, P. Fiadino, and A. D'Alconzo, "Machine-learning based approaches for anomaly detection and classification in cellular networks," in *8th Traffic Monitoring and Analysis (TMA2016) Workshop*, 04 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 based on tcp loss ratio and round trip time," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3709–3722, 2017.

[10] J. Wu, P. P. C. 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*, 2018, pp. 1–9.

[11] F. Lateano, O. Ayoub, F. Musumeci, and M. Tornatore, "Machine-learning-assisted failure prediction in microwave networks based on equipment alarms," in *2023 19th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2023, pp. 1–7.

[12] O. Ayoub *et al.*, "Explainable artificial intelligence in communication networks: A use case for failure identification in microwave networks," *Computer Networks*, vol. 219, p. 109466, 2022.

[13] O. Ayoub, F. Musumeci, F. Ezzeddine, C. Passera, and M. Tornatore, "On using explainable artificial intelligence for failure identification in microwave networks," in *2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, 2022, pp. 48–55.

[14] 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.

[15] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "Active learning for network traffic classification: A technical study," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 1, pp. 422–439, 2022.

[16] Q. Xu and R. Zheng, "When data acquisition meets data analytics: A distributed active learning framework for optimal budgeted mobile crowdsensing," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[17] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[18] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," in *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

[19] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *International conference on machine learning*. PMLR, 2017, pp. 1183–1192.

[20] V.-L. Nguyen, M. H. Shaker, and E. Hüllermeier, "How to measure uncertainty in uncertainty sampling for active learning," *Machine Learning*, vol. 111, no. 1, pp. 89–122, 2022.

[21] E. Ayanoglu, K. Davaslioglu, and Y. E. Sagduyu, "Machine learning in nextg networks via generative adversarial networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 2, pp. 480–501, 2022.

[22] H. Navidan, P. F. Moshiri, M. Nabati, R. Shahbazian, S. A. Ghorashi, V. Shah-Mansouri, and D. Windridge, "Generative adversarial networks (gans) in networking: A comprehensive survey & evaluation," *Computer Networks*, vol. 194, p. 108149, 2021.

[23] Y. Yang, Y. Li, W. Zhang, F. Qin, P. Zhu, and C.-X. Wang, "Generative-adversarial-network-based wireless channel modeling: Challenges and opportunities," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 22–27, 2019.

[24] I. William H Clark, S. Hauser, W. C. Headley, and A. J. Michaels, "Training data augmentation for deep learning radio frequency systems," *The Journal of Defense Modeling and Simulation*, vol. 18, no. 3, pp. 217–237, 2021.

[25] K. Davaslioglu and Y. E. Sagduyu, "Generative adversarial learning for spectrum sensing," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[26] B. Tang, Y. Tu, Z. Zhang, and Y. Lin, "Digital signal modulation classification with data augmentation using generative adversarial nets in cognitive radio networks," *IEEE Access*, vol. 6, pp. 15 713–15 722, 2018.

[27] M. Massaoudi, S. S. Refaat, and H. Abu-Rub, "Intrusion detection method based on smote transformation for smart grid cybersecurity," in *2022 3rd International Conference on Smart Grid and Renewable Energy (SGRE)*, 2022, pp. 1–6.

[28] M. Sun, H. Qian, K. Zhu, D. Guan, and R. Wang, "Ensemble learning and smote based fault diagnosis system in self-organizing cellular networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[29] S. A. Abdulkareem, C. H. Foh, F. Carrez, and K. Moessner, "Smote-stack for network intrusion detection in an iot environment," in *2022 IEEE Symposium on Computers and Communications (ISCC)*, 2022, pp. 1–6.

[30] A. Tesfahun and D. L. Bhaskari, "Intrusion detection using random forests classifier with smote and feature reduction," in *2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies*, 2013, pp. 127–132.

[31] M. Razghandi, H. Zhou, M. Erol-Kantarci, and D. Turgut, "Variational autoencoder generative adversarial network for synthetic data generation in smart home," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 4781–4786.

[32] Y. Qu, H. Ma, Y. Jiang, L. Wang, and J. Yu, "A network data reinforcement method based on the multiclass variational autoencoder," *Security and Communication Networks*, vol. 2022, pp. 1–10, 07 2022.

[33] L. Z. Khan, J. Pedro, N. Costa, L. De Marinis, A. Napoli, and N. Sambo, "Data augmentation to improve performance of neural networks for failure management in optical networks," *Journal of Optical Communications and Networking*, vol. 15, no. 1, pp. 57–67, 2023.

[34] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in neural information processing systems*, vol. 32, 2019.

[35] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[36] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in neural information processing systems*, vol. 32, 2019.

[37] D. Elreedy and A. F. Atiya, "A comprehensive analysis of synthetic minority oversampling technique (smote) for handling class imbalance," *Information Sciences*, vol. 505, pp. 32–64, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519306838

[38] A. Malinin, L. Prokhorenkova, and A. Ustimenko, "Uncertainty in gradient boosting via ensembles," in *International Conference on Learning Representations*, 2021.

[39] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[40] G. Citovsky *et al.*, "Batch active learning at scale," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[41] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal, "Deep batch active learning by diverse, uncertain gradient lower bounds," in *International Conference on Learning Representations*, 2020.

[42] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[43] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013.

[44] A. Pagnoni, K. Liu, and S. Li, "Conditional variational autoencoder for neural machine translation," *arXiv preprint arXiv:1812.04405*, 2018.

[45] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[46] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794.