Machine Learning Based Error Detection in Transient Susceptibility Tests

Roberto Medico, Niels Lambrecht, Hugo Pues, *Member, IEEE*, Dries Vande Ginste, *Senior Member, IEEE*, Dirk Deschrijver, *Senior Member, IEEE*, Tom Dhaene, *Senior Member, IEEE*, and Domenico Spina, *Member, IEEE*

***Abstract*—Compliance to electromagnetic compatibility (EMC) standards is a fundamental requirement for modern integrated circuits (ICs). In this framework, error detection in transient susceptibility tests is of crucial importance to assess the circuit robustness. However, performing such tests is expensive and requires an ad hoc hardware, whose configuration must be adapted for the different test setups, i.e., the transient waveforms, defined in the EMC standards. This paper describes a novel machine learning based approach for error detection, which only requires the raw output data from a susceptibility test: neither additional information about the architecture of the device under test nor the test configuration is needed. We applied and evaluated anomaly detection techniques (a branch of machine learning methods focused on error detection) for transient susceptibility tests with two pertinent examples (one simulation- and one measurement-based). The proposed techniques detected errors successfully in both unsupervised and supervised scenarios. Moreover, these can give insight on the output behaviors that are more likely to cause errors during the test. As shown by our results, an anomaly detection-based approach is an applicable and viable solution for automatic error detection in transient susceptibility tests.**

***Index Terms*—Anomaly detection (AD), electromagnetic compatibility (EMC), machine learning, transient susceptibility.**

I. Introduction

In recent years, electromagnetic compatibility (EMC) concerns have risen in importance due to the progress in process integration, the increase in signal bandwidth, and the modern integrated circuits (ICs) complexity. Indeed, the amount of radio frequency (RF) emission generated by modern ICs increases due to the factors described above, and the reduction of supply voltage and increased number of interfaces decreases the immunity to RF and impulse interference [1]. Hence, it is of paramount importance for IC designers to evaluate the compliance of their novel devices and systems with EMC standards. Many different types of EMC tests exist, such as the direct transient injection (transient DPI) [2] and the RI 130 [3], [4] tests. Note that typical EMC tests are hardware-based; a suitable test setup is required and measurements on a prototype of the IC must be performed to verify its compliance with the EMC standards. The measurements must be repeated for all the different setups of the specific EMC test considered (i.e., the noise pulse and the wire harness configuration can change) leading to an expensive and time-consuming procedure.

Manuscript received December 4, 2017; revised January 31, 2018 and March 5, 2018; accepted March 28, 2018. Date of publication May 27, 2018; date of current version January 20, 2019. *(Corresponding author: Roberto Medico.)*

R. Medico, N. Lambrecht, D. Vande Ginste, D. Deschrijver, T. Dhaene, and D. Spina are with IDLab, Department of Information Technology, Ghent University-imec, Ghent B-9052, Belgium (e-mail: roberto.medico@ugent.be; niels.lambrecht@ugent.be; dries.vandeginste@ugent.be; dirk.deschrijver@ugent.be; tom.dhaene@ugent.be; domenico.spina@ugent.be).

H. Pues is with the Melexis Technologies N.V., Tessenderlo 3980, Belgium (e-mail: hpu@melexis.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEMC.2018.2821712

In this contribution, we present a novel machine learning based approach for error detection in transient susceptibility tests. In particular, the proposed methodology aims to identify failures and errors occurring during transient susceptibility tests via anomaly detection (AD) techniques. These approaches focus on identifying infrequent deviant events, which do not conform to an expected behavior. In the proposed modeling framework, such infrequent events or anomalies typically correspond to the errors occurring during an EMC test. Different types of AD techniques exist and this paper studies the application of such methods according to the unique characteristics of the problem at hand. For example, supposing that the design of the IC-under-study followed EMC aware strategies, it can be expected that the number of errors during a transient susceptibility test will be very limited. The anomalies will add up to a very small fraction of the available data. In the rest of this contribution, the reader may assume that the terms "anomaly" and "error" have the same meaning, while the former will be preferred when discussing the proposed methodology in a more machine learning-oriented framework, and the latter will be used mainly in the application examples. In the proposed framework, once an initial computational effort is made to build the machine learning model, the model itself can be applied to different EMC test setups for the same device under test (DUT). However, it is not possible to directly use a model trained on a specific DUT for error detection on a different DUT; additional tuning is needed since, in general, the noise response depends on the circuit design.

This paper is organized as follows. Section II gives an overview of the properties of the different AD techniques and describes the framework adopted in this paper to build models for error detection in transient susceptibility tests. The validation of the proposed methodology is carried out in Section III by means of two application examples, one based on simulations, the other on measurements. Finally, conclusions are drawn in Section IV.

II. AD ON TIME-SERIES DATA

AD (also referred to as novelty, outlier, or error detection) is a machine learning technique focused on the identification of events that do not conform to an expected (or *normal*) behavior, such as failures and errors. Several techniques exist to detect such anomalies, and these can be classified according to how they model the normal behavior, which machine learning model they employ, and which types of anomalies they are able to detect [5].

In general, when dealing with data representing a sequence of observations evolving over time (*time-series* data), specific challenges arise for data-driven machine learning techniques, including AD ones. A strong correlation exists for sequential data between successive data points and often anomalies can be considered as such only within a specific time frame (*context* anomalies) or together with their context (*collective* anomalies), instead of merely because of individual observations' values (*point* anomalies). Several techniques have been proposed to deal with time-series data according to the properties of the data at hand [6]. Often the data are transformed in a format more suitable for standard machine learning algorithms, e.g., by obtaining a symbolic representation of the discretized time-series data or by extracting features from subsequences of the raw data. If labeled data are available, i.e., normal and anomalous observations are known *a priori*, the problem falls into the category of *supervised* learning and standard machine learning classifiers (such as neural networks or support vector machines) can be trained on the data. Once such a model is trained, it can be used to detect anomalies in unseen data. The training is often complicated by the strong data imbalance (anomalous observations usually add up to a very small fraction of the total number of observations). A classifier trained on this data needs to take the imbalance into account and suitable evaluation metrics need to be chosen. Several methods exist to deal with such imbalance [7], e.g., by directly rebalancing the data (oversampling the minority class or undersampling the majority class) or by modifying the weights of each class in the classifier cost function. Boosting methods, presented in Section III-B2, have been proven to perform well also for strongly imbalanced datasets [8].

When labeled data are not available, e.g., because anomalies are not known *a priori* or it is expensive to manually label them, other techniques falling into the *semisupervised* or *unsupervised* approach can be used. In semisupervised approaches, the model is trained only on data, which is free of anomalies and is then employed to detect anomalies on test data (containing both normal and anomalous observations). In the unsupervised case, instead, no prior knowledge of the training data is given to the model, which tries to separate normal and abnormal observations directly in the given dataset [9]. Note that, while supervised learning approaches can lead to better results, as the model is given more information to learn from, they are often not applicable in practice given the unavailability of labeled data. The two AD methodologies (supervised and unsupervised) presented in this paper are based on the following steps.

1) Raw data segmentation: The raw time-series signal is split into segments.
2) Feature vector representation: Relevant features are extracted from each segment to form the training set.
3) Model building: An AD model (*k*-nearest neighbors (*k*-NN)[9] for the unsupervised case, gradient boosting [10] for the supervised) is built on the training data.
4) Model evaluation: The model performance is assessed.

A detailed description of the proposed modeling framework is given in the remainder of this section.

A. Raw Data Segmentation

The segmentation of raw time-series data is a generic paradigm used to represent time-series data in a format more suitable as input of standard machine learning models. A time-series $\mathbf{T}$ of length Q can be split into several subsequences $\mathbf{S}_i$ of length L with a *sliding window* approach

$$\mathbf{S}_i = \mathbf{T}_{i:i+L} \quad \forall i \in \{1, \ldots, Q-L\}. \tag{1}$$

These segments can be partially overlapping or completely disjoint.

B. Feature Vector Representation

Once the sequences $\mathbf{S}_i$ are obtained, M features can be extracted to describe each segment as a feature vector $\mathbf{F}_i$

$$\mathbf{F}_i = \left[f_0^{(i)}, f_1^{(i)}, \ldots, f_M^{(i)}\right] \quad \forall i \in \{1, \ldots, Q-L\}. \tag{2}$$

These features can be summary statistics, e.g., the maximum, minimum, mean in the considered interval, or features engineered *ad hoc* for the problem at hand. This representation allows one to considerably reduce the dimensionality of the dataset when M is chosen such that $M \ll L$. The new representation can then be used as input training data for a machine learning model. Consequently, identical preprocessing steps need to be applied to the test data in order to detect anomalies.

C. Model Building: Hyperparameters Tuning

Once the type of AD technique is chosen (*supervised*, *semisupervised,* or *unsupervised*), a crucial step for building a machine learning model is hyperparameters tuning. The hyperparameters of a model control how complex the model decision boundary is and how well the model fits to the training data. On one hand, when such complexity becomes too large, the model can start *overfitting* the training data, i.e., the model learns to represent the training data *too well* with an overly complex decision boundary and is not able to generalize to new data. Consequently, the model performance deteriorates on the test data. On the other hand, if the complexity is too low, the model decision boundary could be too simple and unable to properly discriminate between normal and anomalous classes. In this case, the model is *underfitting* the training data and usually this leads to a decrease in performance for both training and test data.

D. Model Evaluation

This step quantifies how well the AD model is performing for the task at hand via a suitable metric, which is the indicator of the model performance and can also be used to tune the hyperparameters, as shown in Section III. Considering the rarity of anomalous events, which leads to a strong data imbalance, standard metrics such as accuracy or the area under the receiver operating characteristic (ROC) curve can be misleading in the context of AD [11], as results close to 100% can be easily obtained by classifying any observation as not anomalous.

Therefore, AD techniques are often evaluated with alternative metrics such as precision, recall, and F_1 on the anomalous class. Precision (P) and recall (R) can be defined in terms of true positives (TP, when a detected anomaly corresponds to an error of the EMC test performed), false positives (FP, when a detected anomaly is not actually an error), and false negatives (FN, when an error is not detected as an anomaly); the F_1 measure is the harmonic mean of precision and recall

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad R = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad F_1 = 2\frac{P \cdot R}{P + R}. \tag{3}$$

Hence, the precision represents the confidence of the model in its error detection, whereas the recall indicates how many errors the model identifies out of all the errors that actually exist. These metrics require the model output to be discrete: for example, 0 for normal instances and 1 for anomalies. However, it is often preferable to have probability scores instead of a binary output to properly represent the likelihood of an event to actually be anomalous. In the latter case, a threshold on the probability needs to be chosen to classify instances and to compute the metrics in (3).

An informative threshold-independent measure to check and compare AD performance is the area under the precision–recall (PR) curve, which is a curve consisting of precision and recall pairs obtained by changing the decision threshold on the predicted probabilities. In scenarios with binary classification on highly imbalanced data, the PR curve can be more informative than other standard measures, like the ROC curve [11]. Indeed, the area under such curve (indicated as PR-AUC in the rest of this paper) summarizes the performance into a single metric robust to data imbalance.

III. Application Examples

This paper describes two application examples of error detection in transient susceptibility tests via AD techniques. In the first example, the transient susceptibility test is performed via simulations, whereas in the second example, measured data are used.

A. IC-Level Transient DPI Test—Unsupervised AD

The IC under test is a digital counter, designed by using the 74HC74 datasheet [12] as a reference. In order to perform a transient DPI test, suitable coupling and decoupling networks are designed at each input/output and power supply pin, as shown in Fig. 1.

Indeed, the purpose of the coupling network is to efficiently couple the transient disturbance signal to the IC, whereas the decoupling network avoids a direct interaction of the disturbance with the voltage supply, and the clock generator. The IC's clock input pin is subject to the noise injection.

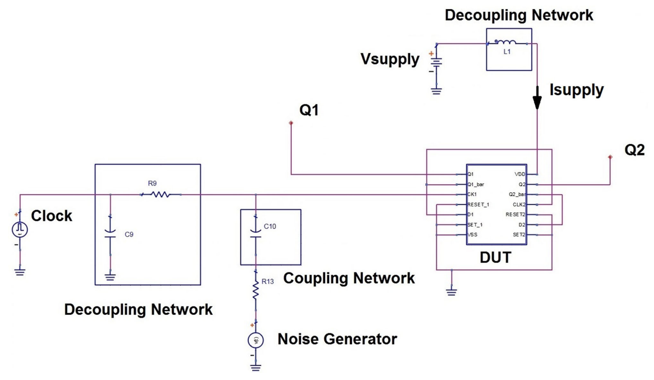

Fig. 1. Example A. Transient DPI test setup for injection at the IC's clock input pin.

In particular, the transient noise generator is assumed as an ideal Thévenin voltage source with a 50-Ω series impedance and the pulses $3a$ and $3b$ described in the standard ISO 7637-2:2011 [13] are injected starting from 100 μs, to allow the IC to operate in steady state before being subjected to the external noise. The time-domain simulations have been performed with the ADS[1] circuit simulator in the range $[0–700]$ μs.

In this example, it was assumed that the DUT passes the transient DPI test if the following conditions hold:

$$\begin{cases} \text{the high-level output voltage must be in the range} \\ [3.84, \max(Q_1, Q_2) + 0.5] \text{ V} \\ \text{the low-level output voltage must be in the range} \\ [\min(Q_1, Q_2) - 0.5, 0.33] \text{ V} \\ \text{the variation of the supply current follows } [14] \\ |\frac{I_{\text{supply}}}{I_{\text{nominal}}}| < 10 \end{cases} \tag{4}$$

where I_{nominal} is the nominal value of the supply current I_{supply} (without noise injection). The values 3.84 V and 0.33 V are chosen based on the technology used for the IC. External CMOS devices must be able to determine whether the output is at a high or a low level; while the values $\max(Q_1, Q_2) + 0.5$ and $\min(Q_1, Q_2) - 0.5$ are case-specific: The maximum output current is obtained in these conditions and excessive current production can damage the circuit.

In this example, the behavior of the DUT is well defined and does not change over time: the supply current is (predominantly) constant and the outputs Q_1 and Q_2 of the digital counter are periodic signals with frequency equal to the half and a quarter of the clock frequency, respectively. Furthermore, we have chosen not to adopt EMC aware strategies during the design phase of the digital counter. So, it is to be expected that the injected noise will have a clear impact on Q_1, Q_2, and I_{supply}. However, it is not possible to predict upfront its effect (*what will be the value of the signals under study when noise is injected in the DUT*); no prior knowledge on the anomalies is available. In this setting, *unsupervised* AD techniques can be useful, since they do not require any additional information besides the raw data. Hence, a suitable machine learning model is applied on

[1]Advanced design system, Keysight Technologies, Santa Rosa, CA, USA.

a dataset (possibly) containing errors where it discriminates between instances belonging to the normal class and potential errors.

1) Raw Data Segmentation: Since the output of the DUT is a 700-μs-long time series, it is first necessary to process this data, as described in Section II-A, in order to represent it in a more suitable format for AD techniques. The first 40 μs of the time series containing the IC transient behavior are not used for training, since the noise is injected once the IC operates at steady state for this susceptibility test, and the noise injection time is known (100 μs). When this would not be the case, existing techniques [15] can be adopted to automatically detect and remove the transient behavior of a circuit from the training data, rather than manually doing so. The use of a sliding window approach to segment the signal is also justified by the fact that the effect of the noise injected on the circuit will affect a group of successive observations (a *collective* anomaly) rather than a single observation (a *point* anomaly). Therefore, the length of the sliding window is an important parameter and it should be chosen according to the interval length of interest to look for anomalies. Since it is not known *a priori* which length is the optimal one in this case, it is convenient to extract partially overlapping segments rather than completely disjoint ones. For this application example, the window length is chosen as the inverse of the clock frequency and the segments are 90$\%$ overlapping.

2) Feature Vector Representation: After the segmentation phase, relevant features are extracted from each segment. This is done in a computational efficient way via the distributed and parallel *tsfresh* library [16]. The following features are extracted from each segment (naming reflects the *tsfresh* standard): *abs_energy, kurtosis, skewness, maximum, minimum, mean, median, mean_abs_change, mean_change,*and*standard_deviation*.

Note that in more complicated settings, standard features do not suffice and more elaborate feature engineering is needed, as shown in Section III-B. In general, since the feature extraction step is blind to the problem at hand, one or more features could be irrelevant and discarded at a later stage to improve computational efficiency. An overview on possible strategies for feature selection (applicable in both supervised and unsupervised approaches) can be found in [17].

3) Model Building: A simple yet powerful unsupervised technique applicable to a dataset of limited size is the *k*-NN approach [9]. This technique makes the assumption that the features extracted from normal observations lie in high-density areas within the feature space, whereas those that are extracted from anomalous segments are far from these dense areas. In this approach, the vectors of extracted features representing each segment are seen as data points in an *M*-dimensional space, where distances between points can be computed (e.g., via Euclidean distance). In this space, it is possible to apply the NN algorithm to assign an *anomaly score* to each segment. Specifically, the score is the mean distance of every data point to its K nearest neighbors, where K is a hyperparameter that can be tuned according to the properties of the data and expected anomalies. If K is too small, e.g., 1 or 2, only isolated anomalies can be detected, since every point that has at least one close neighbor will get a low score. If K is too high, an isolated small cluster of normal observations might be considered anomalous. In this example, since the data are very regular and no different clusters of behaviors are expected, setting $K = 30$ (a sufficiently high value) is a reasonable choice.

4) Model Evaluation: Given that no labels are available, the model predictions are evaluated in those regions that are identified as anomalous, comparing the results obtained with the failure criteria defined in (4).

Fig. 2 shows the results for Q_1, Q_2, and I_{supply}, when 75-V noise pulses are injected into the clock input pin. In this example, the training data consists of 3371 samples, where each sample corresponds to a subsequence extracted with the sliding window approach. For visualization purposes, only a short time interval of data where two anomalies occur is shown. The upper plots show the raw signal waveform, whereas the predicted anomaly scores are shown in the lower plots as overlapping red regions (each region has a length equal to the window length chosen for the segmentation step). The orange bars in the upper plots show the valid range for the outputs as defined in (4). As shown in the plots, higher scores are assigned to regions containing deviant behavior, such as over- or under-shoots. Note that also smaller disturbances are assigned a score higher than the rest, even though the values are still within the valid range. For example, despite that I_{supply} in Fig. 2 is still in the valid range defined in (4), two evident undershoots are present in the data and are successfully identified by the model. This is especially helpful if no specific failure criteria are available or when the user is interested in retrieving any possible disturbance (either critical or not).

Since the scores computed by the chosen AD method are real numbers of arbitrary magnitude, rather than a probability or binary value (0 for normal, or 1 for anomaly), the following strategies can be used to classify each segment: either a threshold τ can be chosen as the minimum score above which a segment will be considered anomalous or, if the number of expected anomalies N is known *a priori*, the segments with the N highest scores can be flagged as anomalous. In the former case, the value of τ can be optimized in the postprocessing step. Fig. 3 shows the number of detected anomalies during a transient DPI test with respect to the value of τ. A suitable threshold can be chosen to detect all (and only) the critical errors or to include other disturbances as well.

The proposed technique was applied to six transient DPI test scenarios depending on the following:

1) the noise injection point: clock, Q_1, Q_2, I_{supply};
2) the noise waveform ($3a$ or $3b$); and
3) the noise amplitude: $[20, 40, 75, 112]$ V;

for a total of 16 raw output signals. By choosing τ, as explained above, the errors defined by the failure criteria (4) have been successfully identified in all cases.

B. SENT-RI 130 Tests—Supervised AD

A test chip provided by Melexis[2] is considered as DUT for a broadband EMC test, namely the RI 130 test [3]. As shown in

[2]Melexis Technologies N.V., Tessenderlo 3980, Belgium.

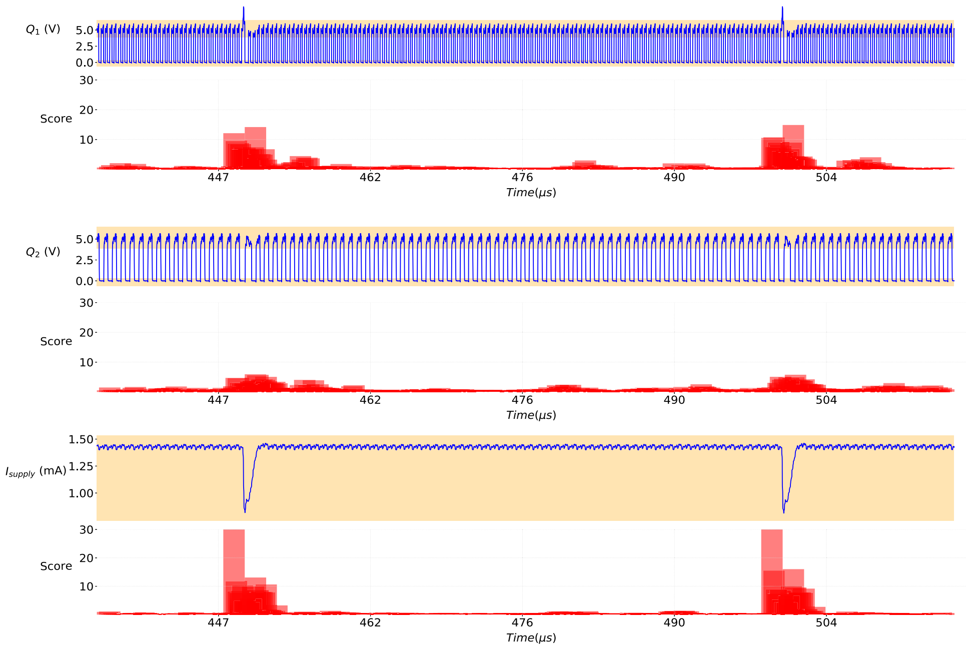

Fig. 2. Example A. Extract of the results of the AD model on Q_1 (top), Q_2 (middle), and I_{supply} (bottom) during a transient DPI test with noise (waveform 3*b*) of 75-V amplitude injected into the clock input pin. For each signal, the raw output waveform is shown on top with the anomaly score shown in red below. Regions of valid behavior according to predefined failure criteria are shown on top of the raw output signal as orange bars.

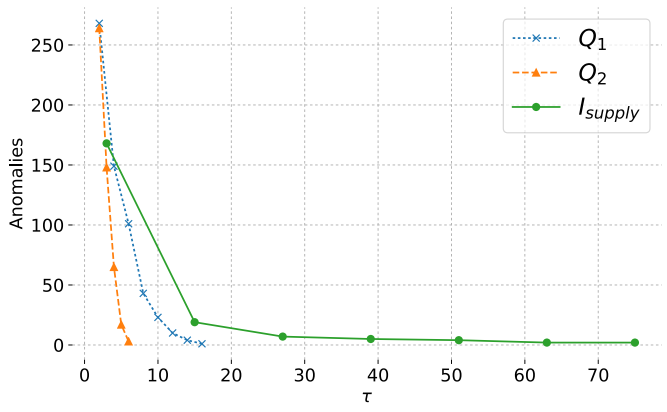

Fig. 3. Example A. How the number of detected anomalies (*y*-axis) change by varying the threshold τ (*x*-axis) for Q_1, Q_2, and I_{supply} for a transient DPI test with noise (waveform 3*b*) of 75-V amplitude injected into the clock input pin.

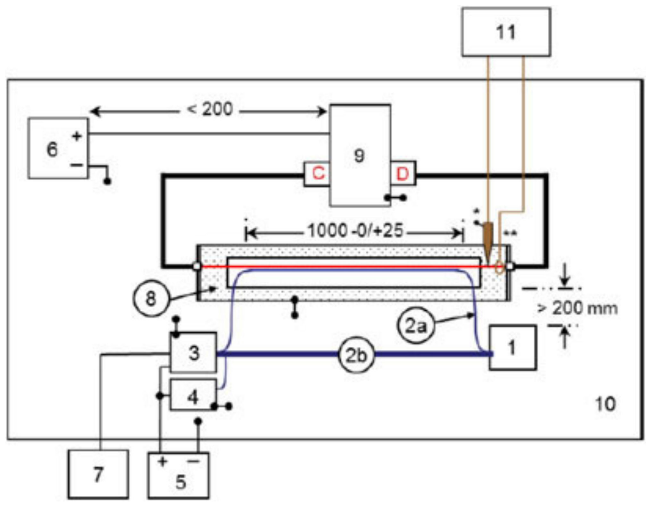

Fig. 4. Example B. Default RI 130 test setup [3], where 1 = DUT, 2a = DUT circuit wire to be tested, 2b = DUT wire harness, 3 = load box, 4 = artificial network, 5 = power supply, 6 = automotive battery, 7 = DUT monitor, 8 = coupling test fixture, 9 = transient generator, 10 = ground plane, and 11 = test point.

Fig. 4, the test bench itself consists of a coupling fixture residing on a large ground plate. In slot A of the coupling test fixture, an aggressor and a single (victim) wire of the wire harness are placed. The other wires lay at least 200 mm away from the coupling test fixture in order to avoid direct field coupling with the aggressor wire. Next, a load simulator, commonly referred to as load box, is connected to the DUT via the wire harness. The transient generator can be operated in four modes, depending on its settings [3]. During the EMC test, the data are transmitted by the DUT via the Single Edge Nibble Transmission for Automotive Applications (SENT) protocol [18] and the information received from the DUT is checked for errors via dedicated hardware. According to the SENT standard [18], a susceptibility test is considered failed if any of the following criteria is met:

$$\begin{cases}\text{More than three erroneous messages out of} \\ \text{any hundred consecutive SENT messages;} \\ \text{More than two successive erroneous SENT messages.}\end{cases} \quad (5)$$

Now, six different RI 130 susceptibility tests have been performed on the same DUT, where the settings used are described in Table I. Specifically, A, B, and C are the available slots in the coupling test fixture in Fig. 4 and *PULSE* and *MODE* represent the transient application mode, as described in [3].

TABLE I
EXAMPLE B. OVERVIEW OF THE PERFORMED RI 130 TESTS

Test	Settings	Messages	Errors	Imbalance
Test1	PULSE A2-2 MODE 3 GND in A	91385	495	0.542 %
Test2	PULSE A2-2 MODE 3 VDD in A	91376	403	0.441 %
Test3	PULSE A2-2 MODE 3 OUT in A, GND in B	91408	244	0.267 %
Test4	PULSE A2-2 MODE 2 GND in A	91379	48	0.053 %
Test5	PULSE A2-2 MODE 3 VDD in A, GND in C	91389	143	0.156 %
Test6	PULSE A2-2 MODE 3 OUT in A, GND in C	91386	134	0.147 %

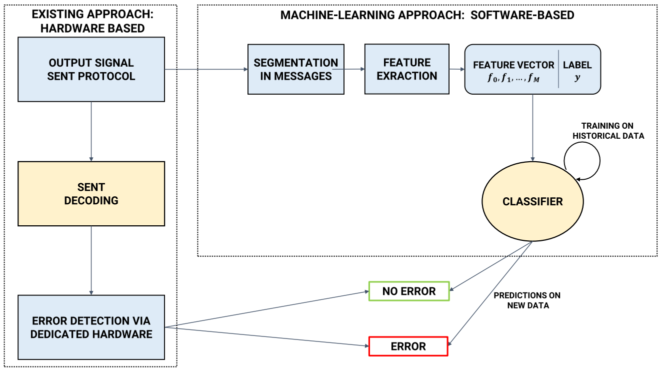
EXISTING APPROACH: HARDWARE BASED
MACHINE-LEARNING APPROACH: SOFTWARE-BASED
OUTPUT SIGNAL SENT PROTOCOL
SEGMENTATION IN MESSAGES
FEATURE EXRACTION
FEATURE VECTOR $f_0, f_1, \ldots, f_M$
LABEL y
TRAINING ON HISTORICAL DATA
SENT DECODING
CLASSIFIER
NO ERROR
PREDICTIONS ON NEW DATA
ERROR DETECTION VIA DEDICATED HARDWARE
ERROR

Fig. 5. Example B. Comparison between the standard hardware-based detection system (left) and the proposed machine learning driven approach (right).

Each test is 100 s long and the sampling rate frequency is 1 MHz, which results in 10^8 measurements per test. Table I also shows the total number of SENT messages and errors per test; the typical length of a message is between 1090 and 1100 data points. The imbalance is always $<1\%$. Fig. 5 illustrates the proposed machine learning approach in comparison with the standard hardware-based one. The AD model operates directly on the raw output SENT signal and the corresponding labels (annotations of the errors), whereas the standard approach employs dedicated hardware to decode the output and check it for errors (this is done via a 4-b checksum test [18]). Once the AD model is trained on historical data, it is able to generate predictions on new (i.e., previously unseen) tests and detect errors.

1) Segmentation and Feature Extraction: The feature extraction step described in Section III-A2 is also applied in this case, but the raw output signal is now split into individual SENT messages, rather than using a sliding window mechanism. Next, the "standard" features listed in Section II-B are extracted together with the following additional features:

1) *first_location_of_maximum*, *last_location_of_maximum*, *absolute_sum_of_changes*, *first_location_of_minimum*, *last_location_of_minimum*, *pct_of_reoccurring_datapoints_to_all*, *binned_entropy*, *sum_of_reoccurring_data_points*, *sum_of_reoccurring_values*, *sum_values*, *number_peaks;*
2) maximum, minimum, and mean length of the intervals where the signal is HIGH and LOW (named with prefix *high_low_stats*);
3) length of the calibration and pause pulse.

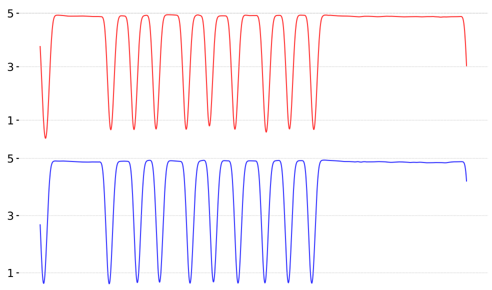

Fig. 6. Example B. Comparison between a SENT message containing an error (top, in red) and one normal (error-free, bottom, in blue) during an RI 130 test.

Note that these features were chosen due to their potential relevance, considering the specific characteristics of the SENT encoding/decoding protocol.

2) Model Building: For this example, unsupervised techniques struggle to achieve good performance given the data characteristics, namely the very high data imbalance, the complexity of the data, and the subtlety of the errors. Indeed, in this case, it is not possible to visually identify the effect on the injected noise on the transmitted message as for the IC in Section III-A. As an example, Fig. 6 shows a comparison between a message with (top) and without (bottom) errors. The two messages have very similar shapes, and no clear disturbance or deviation (e.g., overshoot, undershoot) can be visually appreciated in either. Thus, leveraging the availability of labeled data, a supervised approach is preferred. Specifically, an *ensemble* method is used to model the data and predict anomalies. *Ensemble* methods are machine learning models where the predictions of several *base models* (also referred to as *weak* learners) are combined to form the final prediction. Ensemble methods have been proven to outperform basic classifiers in a wide variety of tasks and domains [19] and can be classified into *Bagging*, *Boosting*, and *Stacking*.

Gradient boosting [10] is a boosting technique that sequentially minimizes a loss function (e.g., for binary classification tasks, a logistic regression loss) by training at each round r a *weak* learner on the *pseudoresiduals* from the previous round (i.e., the difference between the predictions $D^{r-1}(\mathbf{X})$ of the ensemble on the training data $\mathbf{X}$ up to round $r-1$ and the true labels $\mathbf{y}$) [10]. It can be shown that the residuals $\mathbf{y} - D(\mathbf{X})$ correspond to the negative gradients of the loss function w.r.t. $D(\mathbf{X})$. Therefore, the algorithm is in its essence a *gradient descent* algorithm. In each round, the gradients are multiplied (*boosted*) by weights to let the model focus on those samples who were misclassified at the previous rounds, and therefore, are harder to classify.

Usually, decision trees [20] are used as weak learners. Individual fully grown decision trees are very unstable, in the sense

TABLE II
EXAMPLE B. CROSS-VALIDATION RESULTS FROM THE GRID SEARCH

CBL	LR	MD	MCW	PR-AUC
0.25	0.3	3	5	0.874951
0.50	0.3	3	7	0.875725
...	...	...	...	...
0.25	0.01	10	3	0.902365
0.50	0.01	5	3	**0.902850**

that they tend to overfit the training data. Boosting *weak* (shallow) trees can instead avoid overfitting and drastically improve the performance. All the tests performed in this paper use the parallel and efficient implementation of gradient boosted trees from XGBoost [21]. Finally, an advantage of tree-based ensembles is the *built-in* feature selection: when training the model, the most relevant features are chosen at each split in the nodes of the tree. Therefore, the model automatically assigns an *importance* to the features, based on their relevance: the features whose importance is very low can be discarded to improve the computational efficiency. An analysis on the features importance for this case is presented in Section III-B3.

a) Hyperparameters tuning: As introduced earlier, supervised AD models learn a decision boundary on the training set and predictions can be generated by using such boundary to classify test data. However, using a single RI 130 test out of six as validation set would give biased results, since the model hyperparameters would be tuned to give the best performance only on such data. Therefore, to get a better estimate on the real generalization performance of the classifier, i.e., how the classifier would perform with new data, the model is evaluated in a *leave-one-test-out* cross-validation loop: in turn, each RI 130 test (referred to as *fold* in a cross-validation scenario) is used as validation set and the remaining five folds are used as training set. Note that this implies that six different models are trained on different subsets of the complete data. Each model is then evaluated on the held-out fold, which corresponds to the single RI 130 test not used in the training data. Hence, the amount of messages in the training and validation set used for every model can be directly derived from Table I. Finally, the results of the individual models are then averaged to obtain the final performance score. The proposed cross-validation technique also allows for the comparison between models and thus the optimization of the hyperparameters. The following parameters of gradient boosted trees can be tuned to improve performance:

1) max_depth (*MD*)*:* the maximum depth of the trees at each round;
2) min_child_weight (*MCW*)*:* the minimum weight in each leaf node;
3) $learning\_rate$ (*LR*)*:* the learning rate used for gradient descent;
4) $colsample\_by\_level$ (*CBL*)*:* the fraction of features sampled at each split in the trees.

Note that the hyperparameters control how conservative the model is and its convergence. The combination of shallow trees (max_depth of 3 to 5) with many boosting rounds usually results in models that generalize well.

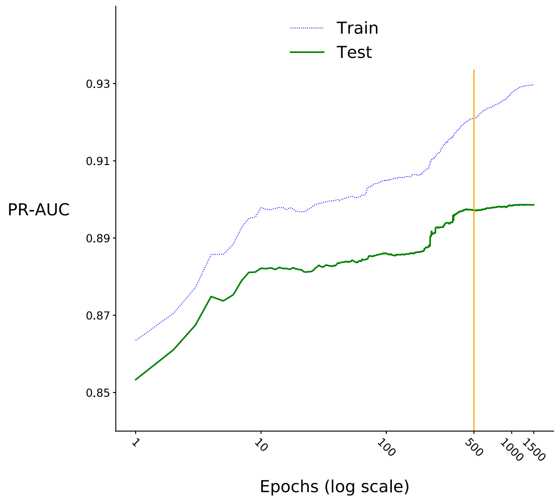

Fig. 7. Example B. Learning process for the model evaluated on Test1. Training and test PR-AUC are plotted versus the epochs (on a logarithmic scale).

TABLE III
EXAMPLE B. PR-AUC VALUE FOR THE BEST MODEL ON EACH TEST

	Test1	Test2	Test3	Test4	Test5	Test6
PR-AUC	0.897	0.871	0.921	0.907	0.910	0.910

Since the output of the model is expressed in terms of normalized probabilities, a threshold $\epsilon \in [0, 1]$ must be defined to discriminate between errors and nonerrors. However, a threshold-independent metric is chosen for the model evaluation and a discussion on the threshold choice is given in Section III-B3. Considering the strong data imbalance shown in Table I, the PR-AUC described in Section II-D is chosen as metric. The optimal values of the hyperparameters are found with a *grid* search. For each parameter, a finite set of possible values is defined and a model is built for each possible combination of values of different parameters (for each fold). The setting with the best final result is selected. The grid search is performed with the following values for each parameter.

1) CBL $\in \{0.25, 0.5, 1\}$.
2) LR $\in \{0.01, 0.1, 0.3\}$.
3) MD $\in \{3, 5, 10\}$.
4) MCW $\in \{3,\ 5,\ 7,\ 10\}$.

With these sets, the number of possible combinations is 108. Another important parameter to be set is the number of boosting rounds (or *epochs*), as discussed in the following.

3) Evaluation: Fig. 7 shows the learning process for the model evaluated on Test 1 (and trained on all the other RI 130 tests, i.e., Tests 2–5) during 1500 epochs. The x-axis represents the number of boosting rounds (epochs) on a logarithmic scale and the y-axis the PR-AUC. As the plot shows, most of the learning happens in the initial boosting rounds when both train and test performances are increasing. Around 500 epochs (indicated by the orange vertical line), the test performance reaches a plateau while the train performance keeps on increasing; the learning process is stopped to avoid overfitting. A similar behavior can be observed for the other models as well; hence, the

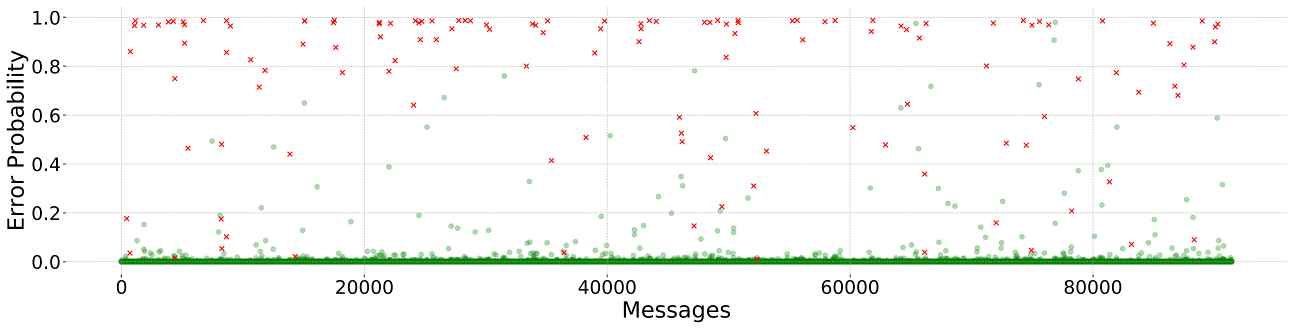

Fig. 8. Example B. Probabilities predicted by the AD model for Test 6. Actual errors are marked with a red cross, whereas messages not containing errors are marked as green dots.

number of epochs is fixed at 500. The results of the grid-search process described above are summarized in Table II. Note that the PR-AUC value shown is the cross-validated performance across all tests. For brevity, the table is sorted by the PR-AUC score (in ascending order) and only shows the two worst and two best combinations out of all the possible ones. The closer the PR-AUC score is to unity, the higher the model capability in correctly detecting errors.

Table III shows a breakdown of the results for each test file using the best parameters. As shown in the results, despite the strong data imbalance, the model is able to detect most errors with high precision. Fig. 8 shows the predicted probabilities for Test 6, where messages without errors are marked as green dots, and errors as red crosses. The x-axis represents the number of the message, whereas the y-axis represents the probability of such message being an anomaly, as predicted by the model. As the plot shows, the model correctly assigns probabilities close to 0 to most of the normal messages (*true negatives*), and probabilities close to 1 to most of the actual errors (*true positives*). There are some exceptions: either true errors whose probability is very low (*false negatives*), or normal messages whose assigned probability is high (*false positives*). By setting the threshold τ for the class decision (1 if $p > \tau$ else 0) of the probabilities to the standard value of 0.5, precision and recall are 87.4% and 77.6%, respectively. However, depending on the desired outcome of the detection model (whether more importance is given to avoiding false positives, and therefore, to high precision, or to retrieving all errors, i.e., high recall), a different threshold can be chosen. Clearly there is a tradeoff between precision and recall, and the model can be tuned to favor one or the other according to case-by-case needs.

Finally, it is possible to extract insights on how the model generates its predictions by looking at the importance of each feature. This reflects how relevant/useful a feature is for the model in identifying a message as an error. Hence, the proposed model is able not only to detect the errors, but also to give insight on the signal behaviors that are more *likely* to cause an error in the DUT. This knowledge can help experts to evaluate the effect of different noise types on the DUT and individuate possible design problems. Fig. 9 shows the top ten features according to their importance, computed by retraining the model on the entire data (six tests). The *score* for each feature

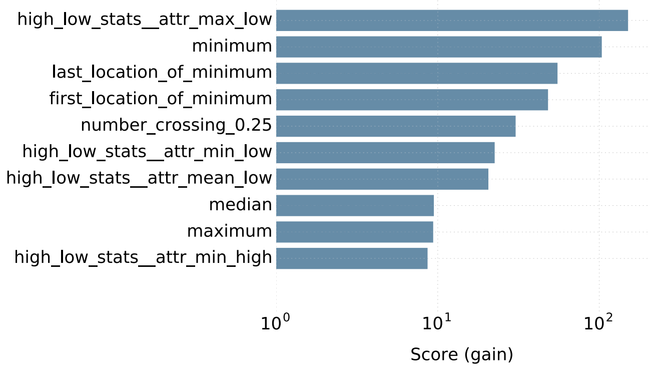

Fig. 9. Example B. Features Importance (gain score) extracted from the gradient boosting classifier trained on the full dataset consisting of six RI 130 tests. Note that the x-axis is represented in a logarithmic scale.

is computed as the average *gain* (i.e., improvement in accuracy) caused by such feature when it is used in branches of trees. As expected, only a few features are relevant for the problem (indeed, the feature extraction step is *blind* to the problem; therefore, potentially irrelevant features are also extracted). However, both standard (e.g., *minimum*, *last_location_of_minimum*) and ad hoc (e.g., *high_low_stats__attr_max_low*, *high_low_stats__attr_min_low*) extracted features appear between the top ten and contribute the most to the model predictions.

Out of the six tests, three failed and three passed according to the criteria defined in (5). The proposed software-based approach, calibrated to have precision equal to 0.947 by choosing a threshold of 0.75, successfully classified five out of six tests as passed or failed. One of the failed tests was not classified as such because of the missed detection of some errors (indeed, the recall with this threshold is 67%). However, the advantage of a high confidence is that the model is very reliable in its error detection. Specifically, the probability P that a test actually fails when is classified as failed by the model in case exactly N errors [where $N \geq 4$ according to the first criteria in (5)] out of hundred consecutive messages are detected can be computed as

$$P_N = \sum_{K\in\{4,\ldots,N\}} \binom{N}{K} \text{precision}^K \cdot (1 - \text{precision})^{N-K} \quad (6)$$

TABLE IV
EXAMPLE B. COMPUTATIONAL TIMES OF THE PROPOSED AD TECHNIQUE

Phase	**Model building**	**Evaluation**
Training	$\sim$ 108s	-
Feature Extraction	$\sim$ 450s	$\sim$ 93s
Detections on new data	-	$\sim 5\mu$s

where N is the maximum number of detected errors out of hundred consecutive messages. In the worst-case scenario, when a test fails because only four out of hundred consecutive messages are detected as error, by (6) we have $P_4 \approx 80.4\%$. Note that the corresponding probability for $N = 5$ and $N = 6$ rises up to ≈96.2% and ≈98%, respectively. Similar considerations hold for the other failure criteria in (5).

Finally, Table IV gives an estimation of the computational time needed for the error detection via the adopted AD technique. In particular, the computational times shown correspond to each phase of the proposed AD technique: feature extraction, training, and prediction on a new test. Despite the large amount of data, the training phase only takes a few minutes, whereas detecting anomalies on a full new dataset (once the model is trained) is almost real time. Note that, during evaluation, the feature extraction step is able to process a complete RI 130 test (around 91 000 messages, or ∼2 GB of raw text data) in less than two minutes, leveraging the parallelization introduced by *tsfresh*. All tests were performed using Python 2.7 on a machine with the following hardware: 12 CPU(s) Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40 GHz with 64 GB RAM. These results show that the proposed method is accurate and computationally efficient, making this approach competitive with existing, expensive hardware-based detection techniques.

IV. CONCLUSION

In this paper, we investigated the applicability and performance of two AD techniques for automatic error detection in transient susceptibility tests with two relevant application examples, one simulation- and the other measurement-based. In both applications, the techniques, one based on *k*-NN (in an *unsupervised* scenario, where no labels are known *a priori*) and the other on gradient boosting (in a *supervised* scenario, where labels are available on historical data) obtained successful results, detecting most of the errors with high confidence. Moreover, the proposed techniques are computationally efficient: on modern hardware, the processing and training phase takes a few minutes, whereas detections on new data are basically real time. These findings demonstrate the potential of the application of AD techniques for automatic error detection in transient susceptibility tests. Since the proposed approach is completely software based, future work will address the challenge of automating the diagnosis of compliance with EMC standards directly during the design phase (e.g., by simulating a susceptibility test on a software model of the IC), before the realization and testing of prototypes, leading to a considerable reduction of production time and cost.

REFERENCES

[1] M. Ramdani, E. Sicard, S. B. Dhia, and J. Catrysse, "Towards an EMC roadmap for integrated circuits," in *Proc. Asia-Pac. Symp. Electromagn. Compat./19th Int. Zurich Symp. Electromagn. Compat.*, May 2008, pp. 8–11.

[2] *Integrated Circuits, Measurement of Impulse Immunity, Part 3: Non-synchronous Transient Injection Method, 1st ed.* , IEC Standard 62215-3, Jul. 2013.

[3] *Electromagnetic Compatibility Specification for Electrical/Electronic Components and Subsystems*. Ford Standard FMC1278, Oct. 2016. [Online]. Available: www.fordemc.com/docs/download/FMC1278.pdf

[4] N. Lambrecht, H. Pues, D. De Zutter, and D. Vande Ginste, "Modeling of contact bounce in a transient electromagnetic compatibility test for the analysis and optimization of nonlinear devices," *IEEE Trans. Electromagn. Compat.*, vol. 59, no. 2, pp. 541–544, Apr. 2017.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys*, vol. 41, no. 3, Jul. 2009, Art. no. 15.

[6] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2250–2267, Sep. 2014.

[7] G. M. Weiss, "Mining with rarity: A unifying framework," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 7–19, Jun. 2004.

[8] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, Dec. 2007.

[9] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS One*, vol. 11, no. 4, Apr. 2016, Art. no. e0152173.

[10] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, pp. 1189–1232, Oct. 2001.

[11] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PloS One*, vol. 10, no. 3, Mar. 2015, Art. no. e0118432.

[12] "Dual D-type flip-flop with set and reset; Positive edge-trigger," Nexperia, Nijmegen, The Netherlands, 74HC74 and 74HCT74 data sheet, Rev. 5, 2015.

[13] *Road Vehicles Electrical Disturbances from Conduction and Coupling; Part 2: Electrical Transient Conduction Along Supply Lines Only*, ISO Standard 7637-2, Rev. 3rd ed., 2011.

[14] S. B. Dhia, M. Ramdani, and E. Sicard, *Electromagnetic Compatibility of Integrated Circuits: Techniques for Low Emission and Susceptibility*. New York, NY, USA: Springer, Jun. 2006.

[15] R. R. Rhinehart, *Nonlinear Regression Modeling for Engineering Applications: Modeling, Model Validation, and Enabling Design of Experiments*. Hoboken, NJ, USA: Wiley, 2016.

[16] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh–A Python package)," *Neurocomputing*, Elsevier, 2018.

[17] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[18] *"SENT—Single edge nibble transmission for automotive applications*," SAE Int., Warrendale, PA, USA, Tech. Rep. J2716, 2016.

[19] T. G. Dietterich, "Ensemble methods in machine learning," *Multiple Classifier Systems*, vol. 1857. Berlin, Germany: Springer, Jun. 2000, pp. 1–15.

[20] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Boca Raton, FL, USA: CRC Press, Jan. 1984.

[21] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

Authors' photograph and biography not available at the time of publication.