Machine-Learning-Based Error Detection and Design Optimization in Signal Integrity Applications

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

***Abstract*— Evaluating the robustness of integrated circuits (ICs) against noise and disturbances is of crucial importance in signal integrity (SI) applications. In this paper, the addressed challenge is to build a software-based framework allowing for automated detection of failures and fast simulation-based evaluation of designs. In particular, these tasks are here addressed using anomaly detection (AD), a branch of machine learning (ML) techniques focused on identifying erroneous or deviant data. In the proposed framework, the ML model only requires the time-domain waveforms and no additional knowledge about the circuit nor about the errors to be identified. Specifically, a two-step approach to detect anomalous behaviors in output waveforms of digital ICs is proposed, comprising a first phase where the ML models are trained to learn relevant features describing the data and a second one where those features are used to identify anomalies with unsupervised or semisupervised AD techniques. Two relevant application examples validate the performance and flexibility of the proposed method.**

***Index Terms*— Anomaly detection (AD), machine learning (ML), signal integrity (SI).**

I. Introduction

IN RECENT years, signal integrity (SI)-aware design methodologies have gained importance due to the increase in signal bandwidth and the high level of integration and miniaturization of integrated circuits (ICs). Often, time-domain simulations are the preferred method for the assessment of SI performance of modern ICs, allowing to analyze eye diagrams, noise, or jitter.

In this context, the main goal of this paper is to introduce a machine learning (ML)-based framework for automated assessment of errors for SI applications, which can be easily integrated in the design phase. Recently, ML techniques have also been used successfully for related applications, such as error detection in electromagnetic compatibility tests [1], performance assessment of high-speed links [2], and uncertainty quantification for high-speed channel signaling [3]. This paper is an extension of preliminary results in [4], where a novel anomaly detection (AD)-based approach was introduced to detect errors in the output of a simple digital circuit affected by jitter.

The proposed AD-based methodology to detect errors in SI applications follows a two-step approach.

1) First, ML models based on representation learning [5] are used to learn relevant features from the output waveform. This corresponds to projecting the input data onto a new latent representation, referred to as *feature space*.
2) Then, an AD algorithm is used to detect anomalous data in the new feature space.

The rationale behind this approach is to exploit the representational power of ML models that are capable of automatically learning features from high-dimensional data, such as time-series data, to obtain a lower-dimensional and relevant representation, where anomalies are easier to identify and locate. To this end, the solution employs AD, a specific branch of ML techniques focused on the identification of unusual events in data, typically due to errors or failures. Unlike traditional predictive ML techniques, where the goal is either classification or regression to estimate the value of a specific target (which is known at training time), AD techniques involve characterizing the normal behavior of a system without any guidance, i.e., without annotations (labels) of possible failures. In these conditions, AD techniques rely on unsupervised or semisupervised learning [6] to detect *anomalies*. In SI applications, these correspond to the undesired behavior of the signals under study, for example, due to crosstalk effects, jitter, or noise. In this text, *anomaly* refers to significant disturbances in the signal as detected by an AD model, while *error* refers to behaviors satisfying specific error criteria. The approach and results presented in [4] are extended here by generalizing the methodology to both unsupervised and semisupervised AD techniques, together with a detailed analysis of the modeling technique with respect to hyperparameters optimization and validation, as well as an in-depth discussion of the results and limitations of the approach. Moreover, an additional example is introduced to validate the approach and illustrate how this can be directly integrated in the design phase for fully automatic design optimization tasks. Specifically, the challenge addressed is to build a solution that does not require

Manuscript received March 1, 2019; revised April 27, 2019; accepted May 7, 2019. Date of publication May 15, 2019; date of current version September 26, 2019. Recommended for publication by Associate Editor M. G. Telescu upon evaluation of reviewers' comments. *(Corresponding author: Roberto Medico.)*

The authors are with the Department of Information Technology, Internet Technology and Data Science Lab (IDLab), Ghent University—imec, 9052 Ghent, Belgium (e-mail: roberto.medico@ugent.be; domenico.spina@ugent.be; dries.vandeginste@ugent.be; dirk.deschrijver@ugent.be; tom.dhaene@ugent.be).

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

Digital Object Identifier 10.1109/TCPMT.2019.2916902

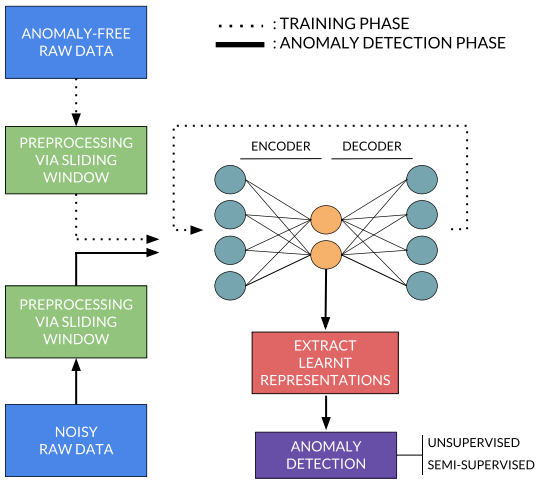

Fig. 1. Overview of the proposed methodology: training phase and AD phase.

a priori knowledge on the circuit design or the types of failures to be expected. Moreover, no recalibration is required for each new simulation. This paper is organized as follows. Section II describes in detail the proposed methodology, which is then validated by two relevant application examples in Section III. Finally, the conclusions are drawn in Section IV. Appendices A and B provide more in-depth details on specific steps of the methodology.

II. Anomaly Detection Framework

A. Notation and Terminology

First, some notations are introduced that will be used throughout this paper, as well as some common terminology. In general, *raw data* (corresponding to the time-domain output waveform of a circuit) are referred to as $\mathbf{T}$, the *processed data* used to train and evaluate the ML model as $\mathbf{X}^{N\times L}$ and the new *feature space* learned by ML models as $\mathbf{Z}^{N\times F}$. Here, N is the number of *subsequences* extracted from the data $\mathbf{T}$ during preprocessing and L is their length, while F is the dimension of the feature space, as described later. Each subsequence is also referred to as a *sample* in the input space. The AD model computes anomaly scores $\mathbf{S} = a(\mathbf{Z})$ for every input sample in $\mathbf{Z}$.

B. Methodology

The proposed methodology is illustrated schematically in Fig. 1. It consists of two main phases: 1) training an ML model to learn features from the data and 2) applying AD techniques to detect errors using the learned features.

As a preprocessing step, the continuous signal $\mathbf{T}$ (raw time-series data) is split into multiple subsequences of a fixed length L using a sliding window approach. Consecutive sequences are extracted by moving a window of size L over $\mathbf{T}$. The stride of the window is typically shorter than L, so to extract overlapping sequences and maximize the amount of training data available. Eventually, a data set $\mathbf{X}^{N\times L}$ is generated in a tabular format, with each row corresponding to a subsequence of length L of the original data.

1) Training Phase—Unsupervised Feature Learning: The next step is to use $\mathbf{X}^{N\times L}$ as input for a representation learning algorithm that learns a new lower-dimensional feature space $\mathbf{Z}^{N\times F}$. Autoencoders [5] are ML models that can be used for this purpose, since they are able to project (encode) the input data $\mathbf{X}$ onto a lower-dimensional representation $\mathbf{Z}$, that is then used to map the inputs back (decode) onto the original space. An autoencoder is trained to reconstruct the original data via a suitable low-dimensional representation with minimal loss of information (*reconstruction loss*). To do this, the learned feature space needs to conserve and summarize the relevant characteristics of the input data. The encoding operation is here referred to as $e(\cdot)$, while $d(\cdot)$ is the decoding operation

$$\begin{aligned}\mathbf{Z} &= e(\mathbf{X})\\ \hat{\mathbf{X}} &= d(\mathbf{Z})\end{aligned} \tag{1}$$

and $\hat{\mathbf{X}}$ is the reconstruction of $\mathbf{X}$ computed by the autoencoder using $\mathbf{Z}$. In practice, both encoder and decoder are the neural networks that have one or more hidden layers. As shown in Fig. 1 (right), an autoencoder has one input layer, one (or more) hidden layer(s) (also referred to as *bottleneck* layer) responsible for learning the feature space $\mathbf{Z}$, and one output layer. Once trained, new data can be fed to the autoencoder, and the corresponding feature space can be extracted from the bottleneck layer. A nonlinear activation function (typically the sigmoid function) is used in the hidden layer, making the learned features nonlinear transformations of the inputs. A variant of classical autoencoders is used, called contractive autoencoder (CA) [7]: A penalty term is added to the reconstruction loss during training to make the network robust to small changes in the input data. The objective function of the optimization during training becomes

$$\min_{\mathbf{W_e},\mathbf{W_d},\mathbf{b_e},\mathbf{b_d}} \mathcal{L}(\mathbf{X}, d(e(\mathbf{X}))) + \lambda||\mathbf{J}_e(\mathbf{X})||_F^2. \tag{2}$$

Here, $\mathbf{W_e}, \mathbf{W_d}, \mathbf{b_e}, \mathbf{b_d}$ contain the parameters (weights and biases) learned by the autoencoder, $\mathbf{X}$ is the input data, and

$$e(\mathbf{X}) = \mathbf{Z} = \text{sigmoid}(\mathbf{W_e}\cdot\mathbf{X} + \mathbf{b_e}) \tag{3}$$

$$d(e(\mathbf{X})) = \hat{\mathbf{X}} = \text{sigmoid}(\mathbf{W_d}\cdot e(\mathbf{X}) + \mathbf{b_d}) \tag{4}$$

are the encoding and decoding functions, respectively. Note that two terms appear in (2): The first is the reconstruction loss $\mathcal{L}(\cdot)$, typically the mean squared error (MSE), and the second is a penalty term that enforces robustness of the learned representation against small changes in the input data. In particular, the latter is formed by the product of a constant $\lambda \in \mathbb{R}_{>0}$ and the Frobenius norm of the X function

$$||\mathbf{J}_e(\mathbf{x})||_F^2 = \sum_{i,j}\left(\frac{\partial z_j(\mathbf{x})}{\partial x_i}\right)^2 \tag{5}$$

where $z_j(\mathbf{x})$ is the jth learned feature, and x_i is the ith element in the input sample $\mathbf{x} \in \mathbf{X}$ with representation $\mathbf{z} \in \mathbf{Z}$.

As indicated in Fig. 1, in the proposed modeling framework, error-free data (when no disturbances are present) are used as input to train a CA by optimizing the objective in (2).

2) Anomaly Detection Phase—Detecting Errors: After training, new data $\mathbf{X}^{\text{test}}$ are simulated and fed to the network to extract the latent representation $\mathbf{Z}^{\text{test}}$. Then, the values of $\mathbf{Z}^{\text{test}}$ are analyzed to detect errors. As no further training is needed for this part, the data are only passed through the encoder (using the learned weights during training) to extract the features. An AD algorithm can then be used to detect anomalies in $\mathbf{Z}^{\text{test}}$ by looking for abnormal samples in the feature space. Since the model is trained to characterize anomaly-free data $\mathbf{X}$, it can be expected that an anomalous subsequence will be mapped onto a different region in the learned feature space. In general, the output of an AD algorithm is an *anomaly score* $\mathbf{S}$, a positive real number indicating how deviant each observation is. Lower scores indicate that a subsequence does not contain anomalies, whereas scores significantly higher indicate that errors occurred. To obtain a binary indicator 0 or 1 whether a subsequence is normal or anomalous, a suitable threshold τ is defined on the maximum score. In practice, τ is a parameter of the method, and it can be tuned to find its optimal value.

In this paper, two flavors of AD are explored, unsupervised and semisupervised. In the former, a model is trained on the unlabeled data with no additional information (if or how many anomalies occur): typically, the assumption here is that anomalies are few and significantly different from the majority of the data, considered as anomaly free. In this paper, the local outlier factor (LOF) algorithm [8] is used for unsupervised AD. The main assumption of LOF is that anomalies lie in the areas of lower density of the input space compared to normal samples. It is essentially a neighbors-based approach since the density of each input is computed based on the density of the nearest neighbors of each data point. It outputs a score $\in [1, +\text{inf})$, where 1 indicates that no anomalies are present and scores >1 mark potential anomalies. In the semisupervised scenario, the AD model is trained on anomaly-free data (hence, it has knowledge on what normal data should look like) and later used to detect anomalies in the new data. While the unsupervised approach is more flexible and requires no training, the assumptions behind it do not always hold. In this case, semisupervised approaches are preferred, at the cost of the additional computing time required for training. Here, LSAnomaly [9] is used. It is a probabilistic AD method to identify outliers, given a training set of anomaly-free samples. To do this, LSAnomaly embeds a least-squares-based classification into a probabilistic framework. Two hyperparameters need to be set: a regularization parameter (ρ) that controls the sensitivity to anomalies, and σ, the length scale parameter of the kernel used in the model, which controls the smoothness of the decision boundary. In our experiments, the value of ρ was set to 0.1 according to the heuristic proposed in [9]. The value of σ was set using the default initialization using k-NN proposed in [9] for the pulse-amplitude modulation (PAM)-4 signal, while $\sigma = 1$ was chosen for the binary signal. This difference is motivated by the fact that the distance between various subsequences is on average smaller for the binary signal, given that it is less dynamic. Once LSAnomaly is trained, it can be used to obtain anomaly scores on new data. Unlike LOF, the scores are bounded in the range $[0, 1]$ and can be directly interpreted as anomaly probabilities.

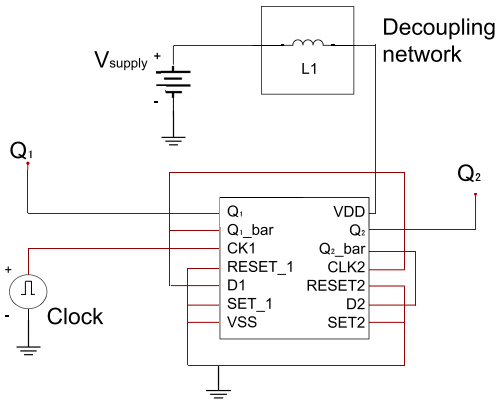

Fig. 2. Example A. The digital counter circuit under study.

III. Numerical Examples

Two application examples are considered to validate this AD-based method. The first example presents a detailed description of the training of a CA to be used for error detection in time-domain signals. The second example illustrates how the AD framework can be used effectively in the design phase to solve SI problems, and how it can be integrated in common design activities, such as optimization. The two examples prove the flexibility of the method, which can be applied to analyze different systems, i.e., a digital circuit in the first example and a channel with distributed elements in the second one. Moreover, it can be employed to evaluate the impact of different undesired effects, i.e., jitter in the first and crosstalk in the second example.

A. Example A: Anomaly Detection Under Jitter Effects

The circuit under study is a digital counter [4], as shown in Fig. 2. It uses a dual D-type flip-flop based on the 74HC74 datasheet [10] and has been analyzed in ADS.[1] It has two output signals (Q_1 and Q_2) with frequency equal to $f/2$ and $f/4$, respectively, where f is the clock frequency. The period of the clock is 200 ns, and its rise/fall time is 5 ns. Furthermore, the clock is affected by Gaussian jitter having a standard deviation of 20 ns. The goal is to apply the unsupervised advocated method to detect problems on the outputs caused by the clock jitter. The digital counter is considered as robust if the variation ΔT of the period of Q_1 and Q_2 with respect to the nominal value (without jitter) satisfies

$$\Delta T_{Q_1} = \pm 40 \text{ ns}, \quad \Delta T_{Q_2} = \pm 80 \text{ ns}.$$

Note that this corresponds to a variation not higher than 10% of the nominal value.

1) Training Phase: To detect if the condition above is satisfied, a CA is first trained to learn relevant features from Q_1 and Q_2 using data where no anomalies are present. To generate this data, a simulation is performed in the range

[1] Advanced Design System, Keysight Technologies, Santa Rosa, CA, USA.

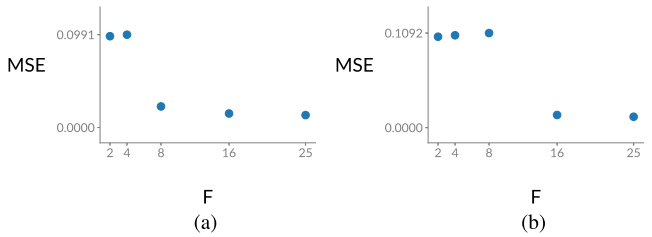

Fig. 3. Example A. The latent dimension F is chosen according to the performance (MSE) on the validation set for (a) Q_1 and (b) Q_2.

$[0, 80]$ μs in the absence of clock jitter. In particular, only the interval $[40, 80]$ μs is retained, i.e., once the IC is operating at steady state. During preprocessing, both Q_1 and Q_2 are normalized to $[0, 1]$. Then, the sliding window mechanism is applied to the data and extracts subsequences of length L, as described in Section II-B. The value of L is set to the period of the signals: 800 for Q_2 and 400 for Q_1, since both signals are sampled with a fixed time-step of 1 ns. This ensures that the subsequences include potential anomalies on the period length. Moreover, the subsequences are extracted with a 99% overlap to obtain more data for training. Next, a CA is trained to reconstruct each subsequence, independently for Q_1 and Q_2, by optimizing the objective in (2) using the Adam optimizer [11], which is an extension of gradient descent. The training is done over several epochs, where one epoch corresponds to a full pass on the training data. In particular, the hyperparameters of the CAs are set as follows: 1 hidden layer with sigmoid activation, F hidden neurons in the bottleneck layer; 250 training epochs; $\lambda = 10^{-4}$ in (2). Following [7], in our experiments, a sigmoid activation was preferred for the hidden layer over other options (e.g., tanh or ReLu) since it produces outputs in the bounded domain $[0, 1]$. Now, choosing the dimension F of the latent space is crucial to successfully train the CA. If F is not sufficiently large, the model may not be able to reconstruct the data well enough. Alternatively, if F is too large, the model will learn a meaningless mapping (e.g., the identity function when $F = L$) and will not be able to generalize to unseen data. To choose a good value of F, part of the training data $\mathbf{X}^{\text{val}}$ is used as validation set. This set is removed from the training data and it is only used after training to evaluate the model performance. Typically, the size of $\mathbf{X}^{\text{val}}$ is chosen as 20% of the available training data. We can then define a set of k candidate values $\{F_1, \ldots, F_k\}$ and pick the one with the best performance on $\mathbf{X}^{\text{val}}$. To ensure that $\mathbf{X}$ and $\mathbf{X}^{\text{val}}$ are significantly different, small random noise was added to the training data, while keeping the validation data unchanged. Fig. 3 shows the best score (measured as MSE) obtained on the validation set for several values of F, for both Q_1 and Q_2. For the former, we can see how the MSE significantly drops when at least $F = 8$ features are used, while this is insufficient for the latter. By choosing $F = 16$, one already obtains a very low MSE on the entire validation set. Note that increasing F beyond 8 and 16, respectively, only slightly improves the performance. In practice, a lower-dimensional feature space is preferred. Once the best F is chosen, the model is trained again on the full training data $\mathbf{X}$.

2) Unsupervised Anomaly Detection: Once the CAs are trained, new data $\mathbf{X}^{\text{test}}$ (with clock jitter) are fed to the model to retrieve the new F-dimensional representation $\mathbf{Z}^{\text{test}}$. LOF is then used to detect anomalies in this representation. LOF has one main parameter: the number of neighbors n. A too large n may result in multiple undetected anomalies that are similar to each other. Using too few neighbors may cause isolated small clusters of anomalies to be overlooked. In general, it is not possible to know *a priori* the value of n that leads to an optimal detection result. It is suggested in [8] to use a range of values for n, as a heuristic. Specifically, the lower bound of the range (denoted n_{LB}) should be chosen as *"the minimum number of objects a cluster has to contain, so that other objects can be local outliers relative to this cluster"* [8]. In this case, given the regularity of the output waveform and the fact that many subsequences will be overlapping, this value can be assumed to be at least 100, corresponding to the number of (partially) overlapping neighbors of each subsequence. The upper bound (denoted n_{UB}) should be *"the maximum number of close by objects that can potentially be local outliers"* [8]. The choice for n_{UB} is less straightforward, but given the assumption that there are few anomalies in the output, this value is set to 400. Considering that the number of extracted subsequences for Q_1 is 9884, the chosen n_{UB} corresponds to approximately 4% of the total size. For Q_2, the number of subsequences is 4888 (since the window length is doubled), and therefore, the range is halved as $n \in [n_{LB}/2, n_{UB}/2]$. In both cases, the range is scanned with a step size of 100. Once the anomaly scores $S^{(n)}$ are computed for various values of n, these are aggregated using the maximum obtained, as suggested in [8] to obtain the final scores $S^*_{Q_1}$ and $S^*_{Q_2}$

$$S^*_{Q_1} = \arg\max_n S^{(n)}, \text{ with } n \in \{100, 200, 300, 400\}$$

$$S^*_{Q_2} = \arg\max_n S^{(n)}, \text{ with } n \in \{50, 100, 150, 200\}.$$

Fig. 4 shows the AD results on both outputs Q_1 (top) and Q_2 (bottom) in the interval $[40, 80]$ μs for both signals. The waveforms are shown on the top, and the scores $S^*_{Q_1}$ and $S^*_{Q_2}$ at the bottom. Since each score is associated with a subsequence, rather than a single point, the score of a point p is computed as the maximum score between all subsequences containing p. The areas where the error criteria defined in Section III-A are not satisfied are highlighted in red in the bottom plots. For both Q_1 and Q_2, the anomaly score is higher in these highlighted regions. Since the method also identifies other disturbances in the signals, a threshold τ (represented as a dotted horizontal green line) needs to be picked to isolate (ideally) all and only the critical regions. Because the method is fully unsupervised, it is not possible to distinguish *a priori* between a tolerable disturbance or an error, given that specific error criteria are defined. Therefore, for optimal retrieval, a threshold can only be defined *a posteriori*. In this case, the threshold τ was picked to isolate regions containing errors, i.e., for Q_1, Q_2

$$\tau_{Q_1,Q_2} \geq S^j_{Q_1,Q_2} \quad \forall j \in \{\text{regions containing errors}\}.$$

This choice was, indeed, possible for both cases, meaning that the model is able to detect all errors without false alarms.

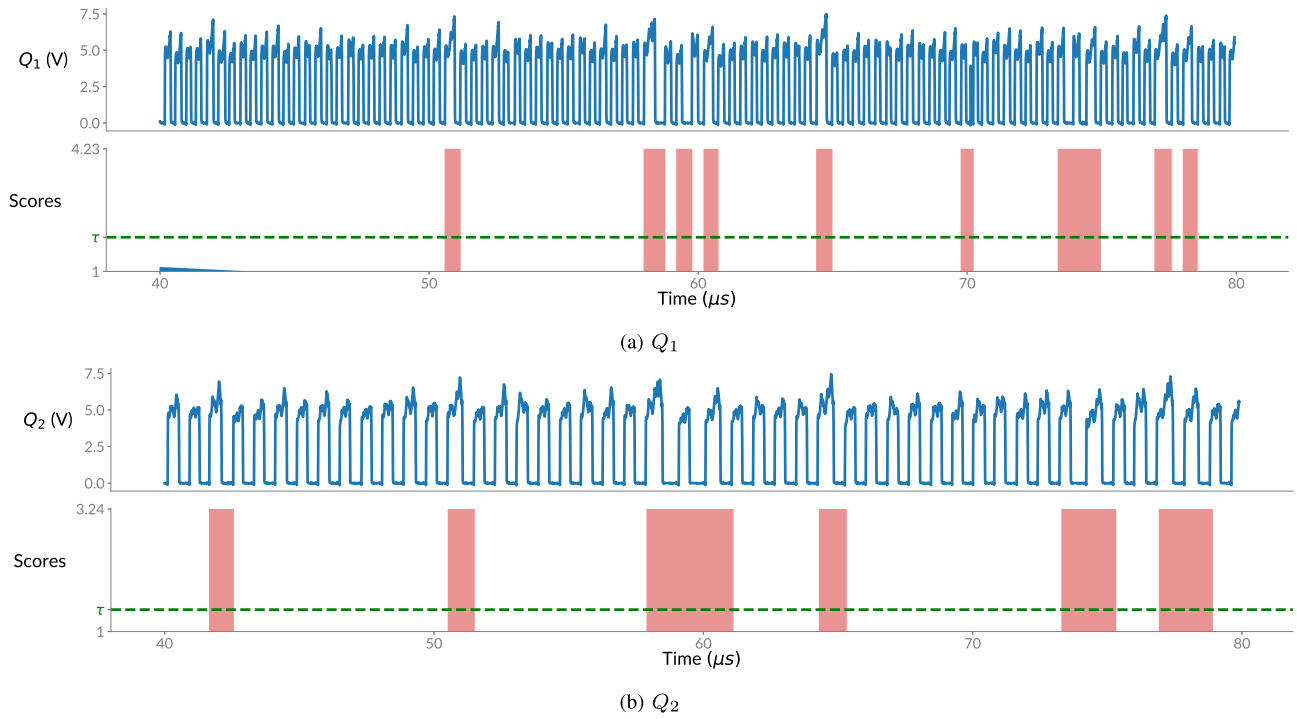

Fig. 4. Example A. Results of the unsupervised AD with LOF, with the anomaly scores computed. For both (a) Q_1 and (b) Q_2, the output waveform is shown above the corresponding anomaly scores. Regions where the defined error criteria are satisfied are highlighted in red. The threshold τ is shown as a dashed green line.

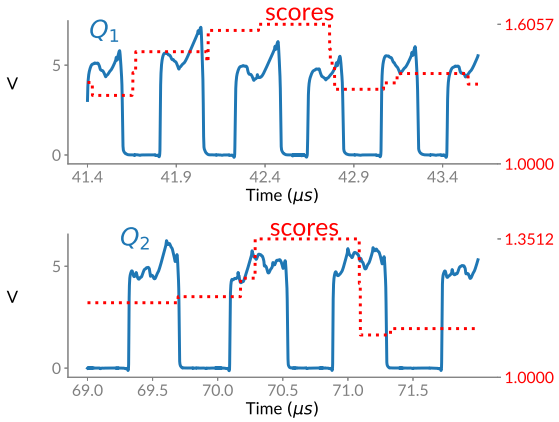

Fig. 5. Example A. The model identifies the disturbances in the input data, without knowledge of the specific *ad hoc* error criteria. For this reason, it also identifies pulses that have an acceptable total length but slightly wrong high- and low-level duration.

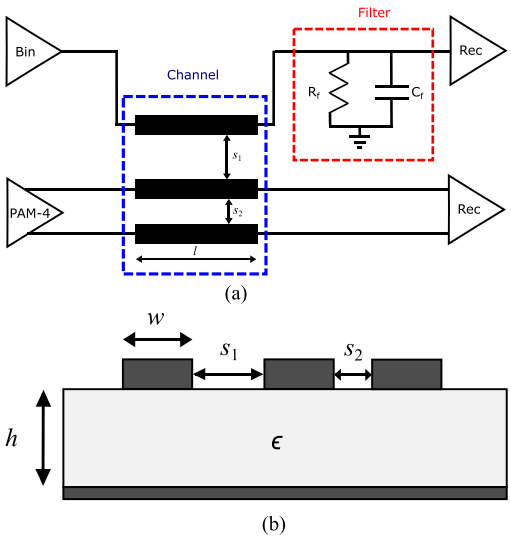

Fig. 6. Example B. (a) High-speed channel. (b) Cross section of the three coupled microstrips.

Another option for the detection is to sort all observations by their anomaly score and flag the K top scoring regions as errors (if, e.g., K is known or can be estimated *a priori*). It is also worth investigating some regions with an elevated score (yet below the threshold), as they, indeed, contain disturbances, even though they do not violate the specified criteria. For example, the region around 42 μs for Q_1 with a high score is shown in detail in Fig. 5 (top). While the total period length does not differ more than 10% from the nominal value, the high level lasts longer than the expected $(T/2)$ for the second and third pulses shown. A similar observation can be made about the region in Q_2 around 70 μs, as shown in detail in Fig. 5 (bottom).

B. Example B: Anomaly Detection-Driven Design Optimization

In this example, the high-speed channel shown in Fig. 6 is considered. A differential PAM-4 signal at 20 Gb/s and a binary (two levels) digital signal at 1 Gb/s are sent into three coupled microstrips. The PAM-4 driver is a behavioral model [12] and generates a pseudorandom bit sequence, while an ideal generator is used to obtain the binary pseudorandom signal. The channel is formed by three coupled microstrips, whose cross section is shown in Fig. 6(b). The conductors have length $l = 5$ cm and width $w = 120$ μm. The spacing between

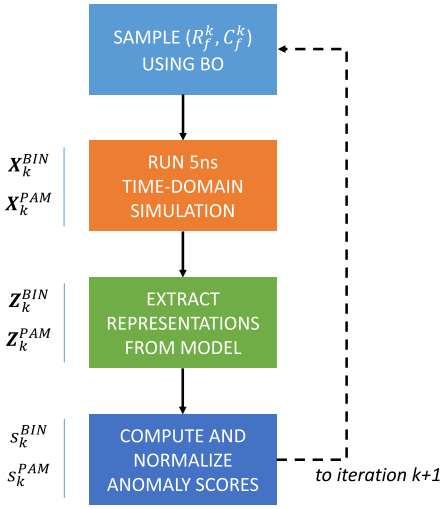

Fig. 7. Example B. The optimization flow using BO: at each iteration k, new (R_f^k, C_f^k) values are sampled and the corresponding anomaly scores are computed and used to guide the sampling in the next iteration. The outputs of each phase are reported on the left side.

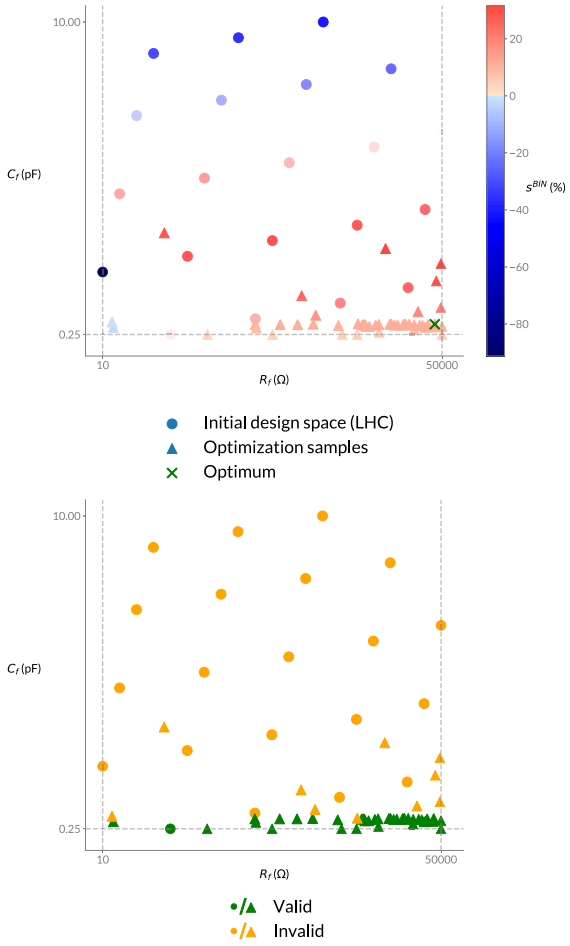

Fig. 8. Example B. Various (R_f, C_f) sampled during initial design (21 samples, shown as circles) and optimization process (50 samples, shown as triangles). The color represents the score on the binary output s^{BIN} (top). The plot on the bottom shows the validity of the same sampled pairs with respect to the constraint on the PAM-4 in (6).

the microstrips is $s_1 = 200\ \mu$m and $s_2 = 100\ \mu$m. The substrate is a Roger RT/duroid 5880 with relative permittivity $\epsilon = 2.2$ and thickness $h = 127\ \mu$m [13]. The scattering parameters of the microstrips are simulated in ADS, and each microstrip is terminated by a resistor $R = 50\ \Omega$ and a capacitor $C = 0.25$ pF connected in parallel, mimicking the input impedance of a receiver [14]. The time-domain simulations are performed in the circuit solver of ADS. In order to reduce the crosstalk effect between the binary and PAM-4 signals, the spacing s_1 between the microstrips where these signals are injected is twice as large as s_2, i.e., the spacing between the two microstrips for the differential PAM-4 channel, as shown in Fig. 6(b). Nonetheless, the interference of the high-speed PAM-4 signals heavily degrades the binary one. Hence, the objective is to design a suitable filter, as indicated in Fig. 6(a), to filter out high-frequency components of the interference of PAM-4 from the received binary signal. However, adding the filter causes a mismatch in the output termination. It is important to verify that, due to the crosstalk, undesired reflections do not couple back into the PAM-4 channel, degrading SI.

1) Training Phase: As explained in Section II, autoencoders are used to learn the lower-dimensional representation $\mathbf{Z^{BIN}}$ and $\mathbf{Z^{PAM}}$ from the output of the digital $\mathbf{X^{BIN}}$ and PAM-4 signal $\mathbf{X^{PAM}}$, respectively, which are preprocessed as discussed in Section III-A1. Since the PAM-4 signal is differential, we have concatenated its (flipped) negative part to the positive one, in order to obtain single time-series data. Next, these new representations are then fed to an AD algorithm to identify abnormal subsequences.

2) Anomaly Detection and Design Optimization: Once the latent features are learned, AD methods can be used to identify deviant subsequences in the output waveforms. In contrast to the previous example, we can no longer assume that only a few anomalies will be present in the data. Therefore, in this case, we apply the CA models to the training data to retrieve the learned representations $\mathbf{Z^{BIN}_{train}}$ and $\mathbf{Z^{PAM}_{train}}$, and later we use these to train an LSAnomaly model for semisupervised AD. In this case, the obtained anomaly scores are used to guide the optimization in a second modeling step. The goal is to find the optimal design of the filter components (R_f and C_f) to improve the signal quality. Specifically, the desired filter should improve the output of the binary signal without degradation of the PAM-4 output. Several optimization algorithms can be used in this framework because the anomaly scores define the cost function to be minimized. Among others, we chose Bayesian optimization (BO) [15]. Specifically, the implementation used for this paper is GPFlowOpt [16]. A detailed description of BO and its implementation in this paper can be found in Appendix A. As illustrated in Fig. 7, at each iteration k, new values of R_f^k and C_f^k are selected using BO; these are then used to run a [0–5]-ns simulation and generate the outputs $\mathbf{X_k^{BIN}}$ and $\mathbf{X_k^{PAM}}$. Next, the trained autoencoders are used to extract $\mathbf{Z_k^{BIN}}$ and $\mathbf{Z_k^{PAM}}$, and finally, the anomaly scores are computed using LSAnomaly. These scores s^{BIN} and s^{PAM} are also normalized with respect to the scores obtained when no filter is used (as detailed in appendix B), and hence, they can be interpreted as improvement/degradation on the signal quality introduced by the filter.

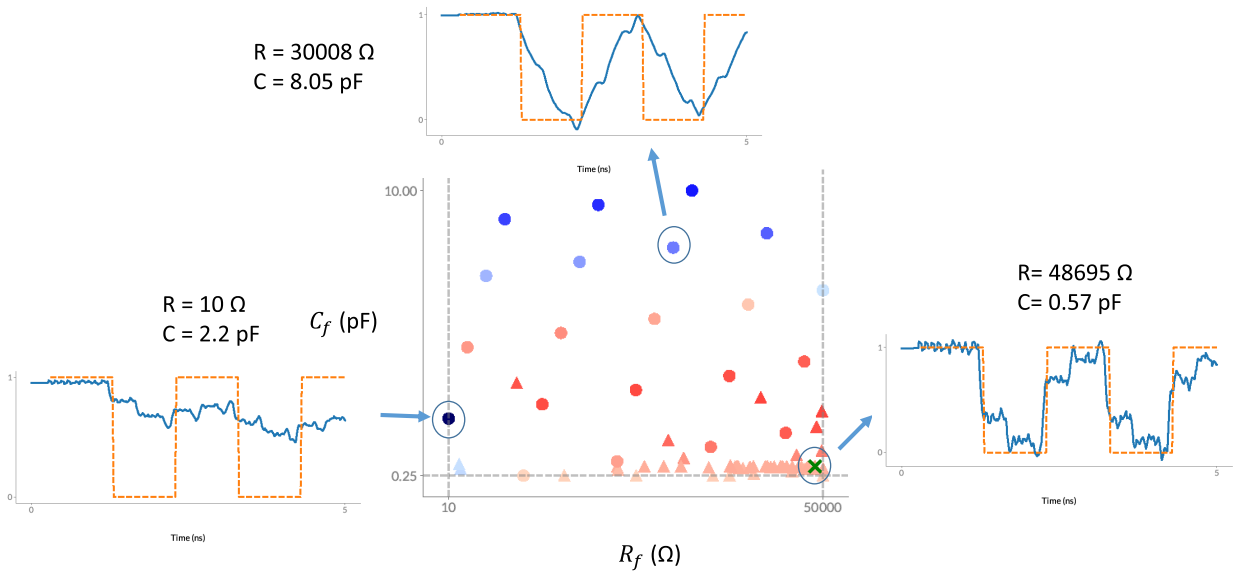

Fig. 9. Example B. Illustration of some output waveforms obtained during the optimization. The orange dashed curve represents the ideal output, and the blue line indicates the actual output.

Formally, the objective of the optimization can then be formulated as

$$\max_{R_f, C_f} \; s^{\text{BIN}} \quad \text{s.t. } s^{\text{PAM}} \geq 0 \tag{6}$$

i.e., the goal is to maximize the improvement on the binary output, under the constraint that the PAM-4 signal does not degrade (i.e., if $s^{\text{PAM}} < 0$). Note that, in practice, the introduction of the filter is not expected to improve the PAM-4 filter, so optimal feasible regions will have $s^{\text{PAM}} \approx 0$. The following settings were chosen for the optimization.

1) $R_f \in [10, 50\,000]$ Ω, $C_f \in [0.25, 10]$ pF.
2) Initial design: 21 (R_f, C_f) samples on Latin hyper cube (LHC).
3) 50 optimization iterations.

The initial LHC design is used to identify promising regions of the design space. At the end of the 50 iterations, the best performing feasible (R_f, C_f) value is chosen as optimum. It is important to remark that only few periods of the binary signal are needed to evaluate the impact of the chosen (R_f, C_f) configuration in terms of anomaly score (given that the PAM-4 is 20 times faster). Hence, the time-domain simulations can be performed over a small range, namely, [0–5] ns, thereby increasing the efficiency of the proposed approach.

Fig. 8 (top) illustrates the results of the optimization process: The 21 samples for the initial design space are shown as circles, and the values sampled during the 50 optimization iterations are shown as triangles. The color of each sample indicates the performance of the filter on the binary output. Compared to the situation without filter, red means improvement, while blue indicates a degradation of the signal quality. The optimum ($R_f^* = 48695$ Ω, $C_f^* = 0.57$ pF) is marked with a green cross. In Fig. 8 (bottom), each sample is instead colored according to its validity: green if the constraint on the PAM-4 is satisfied, else orange. As illustrated, the BO strategy quickly finds an optimal feasible subregion of the space, where the sampling will further focus on.

Fig. 9 shows some of the sampled values in the parameter space and the corresponding 5-ns output waveforms obtained using that setting for the filter. In these plots, the ideal output and actual output are shown with the orange dashed lines and blue solid lines, respectively. It is found that the optimization leads to output waveforms that are smoother and less noisy, as the corresponding anomaly score decreases. Furthermore, it is clear that a bad filter design might lead to worse results than not using any filter at all. The optimum corresponds to an improvement of the anomaly score of the binary signal with respect to the case without filter of around $s^{\text{BIN}} = 11.4\%$, with no degradation on the PAM-4 ($s^{\text{PAM}} = 0$). Even though the exact value of this improvement is somewhat arbitrary (indeed, it depends on the definition of anomaly score used), it highlights how the AD-driven approach achieved the main goal of designing a filter to obtain a significantly better binary output without degrading the PAM-4 output.

Since the computed improvements depend on the way the scores are computed, a comparison with the eye diagrams simulation is performed. This gives a more concrete and immediate appreciation of the actual improvement given by the filter. In Fig. 10, a comparison of the eye diagrams for both binary and differential PAM-4 signals obtained with R_f^* and C_f^* versus the ones computed with no filter is shown. The diagrams are obtained from a transient simulation in the range [0–1] μs, corresponding to 1000 and 20 000 symbols for the binary and PAM-4 signals, respectively. As illustrated, the filter clearly improves the eye of the binary output, without the degradation of the differential PAM-4 signal. In particular, the eye width (EW) and eye height (EH) for the binary signal

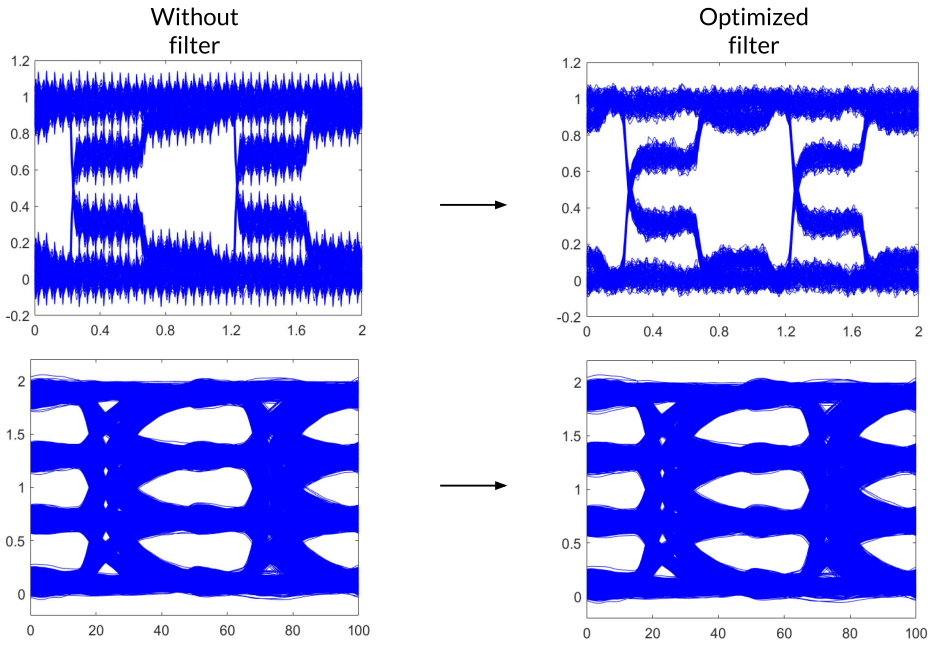

Fig. 10. Example B. The optimization process leads to a significant improvement in the eye opening for the binary signal, without degrading the differential PAM-4 eye.

TABLE I
EH AND EW OF THE BINARY SIGNAL WITH AND WITHOUT THE FILTER

	No Filter	Optimized Filter
EH	0.260 V	0.357 V
EW	0.989 ns	0.971 ns

TABLE II
COMPUTATIONAL TIMES OF THE PROPOSED APPROACH EXPRESSED AS AVERAGE TIME ± STANDARD DEVIATION ACROSS MULTIPLE RUNS

	Binary	PAM-4
Training AE	48 s ± 98.3 ms	47 s ± 1.17 s
Training LS	154 ms ± 6.41 ms	1.57 s ± 42.5 ms
Detection: AE + LS	13.1 ms ± 98.4 μs	52.8 ms ± 1.45 ms

obtained with and without the optimized filter are indicated in Table I.

While the EW is almost identical in both cases (difference of 18 ps), the difference in terms of EH leads to a substantial improvement of the eye opening, thanks to the designed filter. It is important to note that this time-consuming eye diagram computation is only performed to illustrate the improvement introduced by the approach and it is not required by the proposed AD method. As a reference, on our computer,[2] 10 s are required on average to perform a single time-domain simulation in the range [0–5] ns, as needed by the proposed AD-based optimization, while about 57 min are needed for a simulation up to 1 μs, required to compute the eye diagram. However, a direct performance comparison with the existing methods like the eye diagram is not the main goal of this example, as these methods could also be optimized for faster execution. Instead, here, we showcase how the AD-driven approach can be directly integrated in modeling pipelines, and efficiently achieve competitive results in different SI tasks and with different circuits. Finally, the computational time of the proposed method is summarized in Table II, which reports times for both training and AD phase.

[2]Intel Core i5-6300HQ CPU at 2.30 GHz, 2304 Mhz, 4 Core(s), four Logical Processor(s), 16-GB RAM.

IV. CONCLUSION

This paper described a novel ML-based framework to automatically identify the errors in output waveforms for SI applications. The solution is a fully automated and software-based framework that only requires time-domain output waveforms and can be easily integrated in the design phase of modern ICs. Specifically, the proposed framework uses autoencoders to learn relevant features from the outputs of a digital circuit, where anomalies are then detected via AD methods. The model needs to be trained on error-free data only once and can then be used to detect errors on every output generated from the same circuit. Two relevant examples were used to validate the approach: a digital counter affected by clock jitter and a high-speed channel affected by crosstalk. Results from both cases showed how the model was able to successfully identify errors and disturbances in the new outputs. Exploiting this diagnostic power, it was also shown how our solution can be integrated in the design phase and used to guide the optimization of a circuit in a fully automated way, proving that AD techniques can be a valuable asset for SI-aware design strategies. Finally, since the proposed approach requires error-free data, future work could focus on building a completely unsupervised method that jointly learns robust features and performs AD on data containing anomalies, by leveraging the rarity of these events.

APPENDIX A
BAYESIAN OPTIMIZATION

BO is an ML-based approach that aims at finding the global optimum of an objective function g over a bounded

domain $\mathcal{X}$

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} g(\mathbf{x}) \tag{7}$$

where $\mathbf{x}$ is a vector of real-valued parameters and $\mathcal{X}$ is the design space. BO uses a surrogate model of g that is easier and cheaper to evaluate and an *acquisition function*, which is used to select the next candidates to sample [17].

The main idea behind BO is to balance exploration (regions with high uncertainty) and exploitation (region with high scores) in the design space using the uncertainty of the surrogate model. This balance is guaranteed by the acquisition function used [15], [17]. Since the objective (6) has a constraint, in this paper, we jointly learn feasible and optimal regions of the design space using the acquisition functions probability of feasibility (PoF) [18] for the constraint and expected improvement (EI) [19] for the objective, as in [20].

In our case, g is a function that, given R_f and C_f, returns the anomaly scores s^{BIN} and s^{PAM} on the output of the simulation using such filter values. In this paper, we used the *GPFlowOpt* implementation [16] of BO algorithms, as it allows an easy integration with our software pipeline (Python and Tensorflow [21]). The surrogate model(s) used are Gaussian processes [22], and an LHC design was used as the initial design space.

Appendix B
Computing and Normalizing Anomaly Scores

In this appendix, we show in detail how the scores $\mathbf{S}_k^{\text{PAM}}$, $\mathbf{S}_k^{\text{BIN}}$ of all subsequences, outcome of the AD algorithm, are aggregated and normalized to compute the scalar scores s_k^{BIN} and s_k^{PAM} for binary and PAM-4 signals, used in the optimization at each iteration k.

First, the average scores $\overline{s}$ are computed for binary and PAM-4

$$\overline{s}_k^{\text{BIN}} = \frac{1}{M}\sum_{i=1}^{M} S_{i,k}^{\text{PAM}}$$

$$\overline{s}_k^{\text{PAM}} = \frac{1}{N}\sum_{i=1}^{N} S_{i,k}^{\text{PAM}}$$

where M and N are the respective number of subsequences extracted.

These raw scores $\overline{s}_k^{\text{BIN}}$ and $\overline{s}_k^{\text{PAM}}$ obtained are then normalized with respect to the improvement or degradation compared to the raw scores $\overline{s}_{nf}^{\text{BIN}}$, $\overline{s}_{nf}^{\text{PAM}}$ obtained when no filter is used. The normalization is done as follows:

$$s_k^{\text{BIN}} = 100 * \frac{\overline{s}_{nf}^{\text{BIN}} - \overline{s}_k^{\text{BIN}}}{\overline{s}_{nf}^{\text{BIN}}} \tag{8}$$

$$s_k^{\text{PAM}} = 100 * \frac{\overline{s}_{nf}^{\text{PAM}} - \overline{s}_k^{\text{PAM}}}{\overline{s}_{nf}^{\text{PAM}}}. \tag{9}$$

If $s_k^{\text{BIN}} \geq 0$, the filter R_f, C_f introduced an improvement (of the anomaly score) in the binary output, else a degradation.

References

[1] R. Medico *et al.*, "Machine learning based error detection in transient susceptibility tests," *IEEE Trans. Electromagn. Compat.*, vol. 61, no. 2, pp. 352–360, Apr. 2019.

[2] R. Trinchero, P. Manfredi, I. S. Stievano, and F. G. Canavero, "Machine learning for the performance assessment of high-speed links," *IEEE Trans. Electromagn. Compat.*, vol. 60, no. 6, pp. 1627–1634, Dec. 2018.

[3] H. M. Torun, J. A. Hejase, J. Tang, W. D. Beckert, and M. Swaminathan, "Bayesian active learning for uncertainty quantification of high speed channel signaling," in *Proc. IEEE 27th Conf. Elect. Perform. Electron. Packag. Syst. (EPEPS)*, Oct. 2018, pp. 311–313.

[4] R. Medico, D. Spina, D. VandeGinste, D. Deschrijver, and T. Dhaene, "Autoencoding density-based anomaly detection for signal integrity applications," in *Proc. IEEE 27th Conf. Elect. Perform. Electron. Packag. Syst. (EPEPS)*, Oct. 2018, pp. 47–49.

[5] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[6] 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.

[7] S. Rifai, P. Vincent, X. Müller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. 28th Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 2011, pp. 833–840.

[8] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf.*, Dallas, TX, USA, May 2000, pp. 93–104.

[9] J. A. Quinn and M. Sugiyama, "A least-squares approach to anomaly detection in static and sequential data," *Pattern Recognit. Lett.*, vol. 40, pp. 36–40, Apr. 2014.

[10] *Dual D-Type Flip-Flop With Set and Reset; Positive Edge-Trigger*, document 74HC74 and 74HCT74 Data Sheet, Rev. 5, Nexperia, 2015.

[11] D. P. Kingma and J. Ba. (Dec. 2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[12] D. Luo, M. Kalantari, and P. Yue, "Study on the effects of distortions and common-mode noise in high-speed PAM-4 systems," *Electron. Lett.*, vol. 54, no. 8, pp. 484–486, Apr. 2018.

[13] *RT/Duroid 5870/5880 High Frequency Laminates*, Rogers, Toronto, ON, Canada, 2018.

[14] C. Morgan and A. Healey, "A comparison of 25 Gbps NRZ & PAM-4 modulation used in reference, legacy, and premium backplane channels," in *Proc. Int. Symp. Microelectron.*, Jan. 2012, vol. 2012, no. 1, pp. 283–294.

[15] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.

[16] N. Knudde, J. van der Herten, T. Dhaene, and I. Couckuyt, "GPflowOpt: A Bayesian optimization library using tensorflow," in *Proc. Workshop Bayesian Optim. Neural Inf. Process. Syst.*, Nov. 2017, pp. 1–5.

[17] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 2951–2959.

[18] M. Schonlau, "Computer experiments and global optimization," Ph.D. dissertation, Dept. Statist., Univ. Waterloo, Waterloo, ON, Canada, 1997.

[19] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *J. Global Optim.*, vol. 13, no. 4, pp. 455–492, 1998.

[20] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints," in *Proc. ICML*, Jun. 2014, pp. 937–945.

[21] M. Abadi *et al.* (Jan. 2015). "TensorFlow: Large-scale machine learning on heterogeneous distributed systems." [Online]. Available: http://tensorflow.org/

[22] C. E. Rasmussen, "Gaussian processes in machine learning," in *Proc. Adv. Lectures Mach. Learn.* Berlin, Germany: Springer, Feb. 2003, pp. 63–71.