Circuits, Systems, and Signal Processing
https://doi.org/10.1007/s00034-026-03531-4

Identification of Simultaneous Soft Faults in Analog Circuits Using a Hybrid PSO-Machine Learning Approach

M. I. Dieste-Velasco[1]

Received: 1 November 2025 / Revised: 29 January 2026 / Accepted: 30 January 2026

Abstract
Analog circuits are fundamental to a wide range of industrial systems, where their evaluation is essential for ensuring operational reliability and preventing system failures. However, diagnostic methodologies for analog circuits are markedly less developed than those for their digital counterparts, primarily due to the inherent difficulty of detecting soft faults within analog environments. One particularly challenging category of faults involves simultaneous degradations across multiple components that do not result in a hard failure of the circuit. Indeed, there is a notable lack of studies addressing the detection of simultaneous soft faults in analog circuits. This study proposes a method for identifying this type of soft fault occurrence in analog circuits by combining Machine Learning (ML) techniques, specifically Random Forests and Artificial Neural Networks, with an Evolutionary Algorithm (EA) based on Particle Swarm Optimization (PSO). The proposed approach is validated on a second-order Sallen-Key band-pass filter, a circuit in which soft fault classification is particularly challenging. Furthermore, the study highlights the performance improvements achieved through the proposed combined method in detecting and classifying simultaneous soft faults. This study demonstrates that an iterative process combining ML and EA techniques enables accurate fault prediction in electronic circuits. Moreover, the integration of these strategies can enhance the performance of classification problems that are traditionally addressed using either ML or EA in isolation. The effectiveness of the proposed method is evaluated using several statistical metrics, including the Matthews Correlation Coefficient (MCC), F1-score, and others.

Keywords Machine learning · Evolutionary algorithms · RF · ANN · PSO · Simultaneous soft fault diagnosis · Fault classification

✉ M. I. Dieste-Velasco
midieste@ubu.es

[1] Electromechanical Engineering Department, University of Burgos, Avda. de Cantabria s/n, Burgos, Spain

Published online: 09 March 2026

1 Introduction

Analog circuits are commonly found in industrial and avionics systems [21], which makes the identification and classification of faults essential to guarantee their correct performance. Most existing methods used for fault detection in electronic circuits and other engineering components rely on either machine learning (ML) methods or evolutionary algorithms (EAs) independently to achieve fault classification and detection. However, it is uncommon to find approaches in the literature that combine and hybridize both technologies for fault classification. Therefore, this study seeks to contribute to the potential of leveraging a synergy between these two methodologies to enable fault prediction and classification in electronic circuits. While the detection of individual faults in analog circuits has been studied in recent years, the analysis of multiple, simultaneously occurring faults remains significantly less explored, with only a limited number of works addressing this problem. Accordingly, this study focuses on the detection of faults that occur simultaneously in analog circuits and aims to advance the application of hybridized machine learning and evolutionary algorithm techniques for the prediction and classification of such simultaneous faults. First, a review of the state of the art is provided, covering both the machine learning techniques to be used and the evolutionary algorithms that will be combined with them for the prediction and classification of faults in analog circuits. A second-order Sallen-Key filter was selected as the circuit under test (CUT), due to the specific challenges it presents for fault detection. The main challenge lies in the overlap of predictive variables within the frequency domain, which complicates the feature extraction process used for fault prediction. A dedicated feature extraction method is therefore proposed, which will subsequently be applied in conjunction with the selected ML and EA methods.

2 State of the Art

Some of the faults that may occur in analog electronic circuits include hard faults, soft faults, and intermittent faults, among others. Since the prediction of a system's or component's degradation is important for anticipating failures and optimizing maintenance, several methods can be found in the literature for diagnosing the behavior of electronic circuits [1]. In the review by Afacan et al. [1], recent advancements in machine learning (ML) techniques for analog circuits are discussed, highlighting how ML-based methods can be applied for fault diagnosis. Some methods for fault diagnosis in analog circuits can be found in Zhang and Li [32] who defined an output response consisting of a square matrix whose elements may vary depending on the circuit fault and, hence, they diagnosed the faults by comparing the faulty state with the normal behavior of the circuit. Another example can be found in Deng and Chain [8], where an approach for extracting fault features to diagnose soft faults in electronic circuits is outlined. On the other hand, Shi et al. [23] proposed a method that employed density peaks clustering and a dynamic weight probabilistic neural network for analog circuit fault diagnosis, using an operational amplifier active filter as the circuit under test and a method based on a fault dictionary was proposed in [26] for detecting and identifying catastrophic faults on the basis of a diagnostic test. On the other hand, in

Shi et al. [24] a fault diagnosis method for analog circuits was shown by using density peak clustering and a voting probabilistic neural network. In another study, Arabi et al. [2] presented a machine learning-based method for identifying and categorizing parametric faults in analog circuits, utilizing frequency response characteristics of output voltage and supply current for feature extraction. After evaluating multiple classifiers, the quadratic discriminant classifier was selected for its superior accuracy. Recent works have explored fault diagnosis and health prediction using diverse ML techniques. Regarding Artificial Neural Networks (ANNs) and Adaptive Neuro-Fuzzy Inference Systems (ANFISs), a fault diagnosis approach for analog electronic circuits using a Sugeno fuzzy logic classifier, based on statistical analysis of the circuit's frequency response, was proposed by Nasser et al. [19] to detect and identify faulty components. Similarly, Arabi et al. [3] proposed a fault classification approach for analog integrated circuits using a multiclass Adaptive Neuro-Fuzzy Inference System (ANFIS). Tadeusiewicz and Hałgas [25] proposed a method for diagnosing multiple soft faults in analog linear circuits, solving a least squares optimization problem using the Levenberg-Marquardt algorithm and a simultaneous fault detection and isolation method for analog circuits was employed by Moezi and Kargar [17], using Convolutional Neural Networks (CNNs), spectrogram-based feature extraction, and Monte Carlo simulations. Further studies can be found in Binu et al. [5], which reviewed publications from the past few years on fault diagnosis in analog circuits. Some other studies such as of Sheikhan and Sha'bani [22] analyzed the employment of ANN and Particle Swarm Optimization (PSO) in fault detection in analog circuits. Likewise, Zhao et al. [34] proposed an analog circuit fault diagnosis method that integrates ensemble empirical mode decomposition for feature extraction, the maximum information coefficient for feature selection, and particle swarm optimization to optimize Support Vector Machine (SVM) classification. Likewise, soft faults are analyzed in Karthi and Kavitha [11] by introducing an excitation signal to the circuit and acquiring the real and imaginary components of the output frequency response using Monte Carlo analysis, and Zhong et al. [35] used deep belief neural networks to detect intermittent faults in analog circuits. A dual-input model based on a multi-scale self-normalizing convolutional neural network was proposed by Yang et al. [29] to detect faults using circuit response signals and a Fully Convolutional Network (FCN) was employed by Miao et al. [16] as a fault diagnosis model for analog circuits. On the other hand, a fuzzy classifier for diagnosing single and multiple soft faults in analog electronic circuits was proposed by Kumar and Singh [13], frequency-based method to extract voltage features was employed by [9] in order to diagnose single soft faults in electronic circuits using supervised learning with various classifiers k-Nearest Neighbors (KNN), Naive Bayes (NB), Random Forest (RF), Discriminant Analysis Classifier (DAC), Classification Decision Tree (CDT), Support Vector Machines (SVM) and Artificial Neural Networks (ANN), to demonstrate the method's effectiveness, and a fault detection approach for analog circuits using an Extreme Learning Machine (ELM) optimized with the firefly-chaos algorithm was employed by Yu et al. [30]. Zhao et al. [33] introduced a fault diagnosis method based on Deep Belief Networks (DBN), and Zhang et al. [31] proposed a wavelet transform-based feature extraction method combined with a Multiple Kernel Extreme Learning Machine (MKELM) for diagnosing analog circuit faults, where the extracted features were used to train an

MKELM model with its parameters optimized via particle swarm optimization. Likewise, Parai et al. [20] analyzed fault diagnosis in analog circuits by integrating output responses from multiple input signals and applying data fusion with Principal Component Analysis (PCA) and SVM classification and Bilski [4] proposed a hierarchical two-stage classification approach using self-organizing maps to separate easy and difficult fault cases, followed by Random Forest (RF) for complex cases, among many others.

This study is organized as follows: Sect. 3 describes the methodology, including the machine learning algorithms as well as the evolutionary algorithm employed. Section 4 shows the feature extraction technique applied to the circuit under test. Section 5 presents the results obtained as well as a discussion of them, while Sect. 6 provides the conclusions obtained.

3 Methods

The hybridization process is carried out through two distinct approaches, aimed at optimizing the convergence of the methodology employed in this work. First, the Random Forest (RF) machine learning method is used to classify the faults present in the circuit. Subsequently, an evolutionary algorithm (EA) based on Particle Swarm Optimization (PSO) is applied, taking the RF output as the starting point in order to further optimize the results. Additionally, a pattern recognition neural network (Patternnet) is employed to classify the faults, enabling a comparative evaluation with the results obtained using RF. This ANN model is also hybridized with PSO. In this second case, the EA is first used to determine the optimal parameter values. Once the appropriate number of neurons in the hidden layer has been identified, a grid search is conducted to further optimize the configuration based on the EA-derived results. A comparative analysis is also presented between the results obtained from the Random Forest and ANN models, each hybridized with the respective EAs described above.

3.1 Hybridization of Random Forest (RF) and Particle Swarm Osptimization (PSO)

Given a training dataset $\mathcal{T} = \left\{\left(x_j, y_j\right)\right\}_{j=1}^{n}$ where $x_j = (x_{1j} \ldots x_{fj}) \in R^f$ are the input features and $y_j \in \{1 \ldots C\}$ are the class labels, the objective is to develop a learning model $f_\theta \in R^f \rightarrow \{1 \ldots C\}$ that minimizes the classification error when predicting among the C classes.

According to Breiman [7] in Random Forest, via Bootstrap Aggregation (Bagging), the model f_θ is an ensemble of M decision trees $\left\{h_m\left(x_j, \Theta_m\right)\right\}_{m=1}^{M}$ trained using bagging, where each Θ_m is an independent and identically distributed random vector employed in the tree development. Each base learner is trained on a bootstrap sample $\mathcal{T}_m \subset \mathcal{T}$ and then for a given input x_j the ensemble prediction is obtained by majority voting as Eq. (1) shows.

$$f_\theta\left(x_j\right) = majority\_voting\left\{h_m\left(x_j, \Theta_m\right)\right\}_{m=1}^{M} \tag{1}$$

Cross-validation which divides the dataset into k subsets (folds) for training and validation over multiple iterations will be used in this study. The data $\mathcal{T} = \{(x, y)\}$ will be split into k subsets so that $\mathcal{T} = \bigcup_{j=1}^{k} \mathcal{T}_j$, where $\mathcal{T}_j \bigcap \mathcal{T}_{j'} = \varnothing$ for $j \neq j'$, containing each subset approximately the same number of data points (N/k), being N the total number of data points and in each iteration of the cross validation. In each iteration, one subset $\mathcal{T}_j$ is used for validation and the remaining $(k-1)$ subsets for training.

The hyperparameter optimization is selected so that $\theta \subset \Theta$. In this current study the hyperparameter selected are $\theta = (\theta_1, \theta_2) = (NumLearningCycles, MinLeafSize)$. The dataset $\mathcal{T} = \left\{\left(x_j, y_j\right)\right\}_{j=1}^{n}$ is split into k disjoint folds and the cross validation of the classification error associated with θ is obtained according to Hastie et al. [10] from Eq. (2).

$$CV(\theta) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, \widehat{f}_{\theta}^{(-\mathrm{K}(i))}\left(x_j\right)\right) \tag{2}$$

where, $\hat{f}_{\theta}^{(-k(i))}$ is the classifier trained on the complete dataset excluding the data from the current fold $\mathrm{K}(i)$, using the optimized hyperparameters θ, $L(y_i, \widehat{y})$ is the 0-1 loss function, N is the number of training samples and y_i is the actual value or class value corresponding to the input x_j.

Finally, the selection of the hyperparameters $\theta = (\theta_1, \theta_2) = (NumLearningCycles, MinLeafSize)$ is performed using Bayesian optimization. In this study, the function $CV(\theta)$ as defined in Eq. (2), is minimized over the search domain. Matlab™ 2022b, along with the Statistics and Machine Learning Toolbox of Matlab™ 2022b [27] and the Deep Learning Toolbox of Matlab™ 2022b [28], were used throughout this study. The optimization is carried out using the *Expected Improvement Plus acquisition function*, as implemented by Matlab™. The search will continue until the maximum number of iterations is reached. Once the Random Forest algorithm has been executed, an evolutionary algorithm will be applied to optimize the parameters previously selected by the machine learning method, with the goal of improving hyperparameter search efficiency. Specifically, this study employs Particle Swarm Optimization (PSO). That is, the evolutionary algorithm takes as its starting point the data initially generated by the Random Forest and subsequently performs local optimization in the surrounding parameter space. This strategy enables faster convergence compared to the alternative approach, in which the evolutionary algorithm is used to search for optimal parameters without prior guidance from the Random Forest. In this case *NumLearningCycles and MinLeafSize* are the variables to be optimized, where *NumLearningCycles* refers to the number of decision trees trained in the ensemble. In the case of Random Forests, each tree is built using a bootstrap sample drawn with replacement from the training set. The ensemble then aggregates predictions via majority voting in classification tasks. Increasing the number of trees generally reduces variance and improves generalization, although the benefit plateaus beyond a certain point. On the other hand, the *MinLeafSize* parameter controls the smallest number of samples that a decision tree allows in its final nodes.

Table 1 Particle swarm optimization (PSO) algorithm [12]

$$v_j^{(k)} = w * v_j^{(k-1)} + c_1 * r_1 * \left(pbest_j - x_j^{(k-1)}\right) + c_2 * r_2 * \left(gbest_j - x_j^{(k-1)}\right)$$

$$x_j^{(k)} = x_j^{(k-1)} + v_j^{(k)}$$

It influences how much the tree can subdivide the data, which in turn affects its structural complexity. Setting a low value enables highly detailed splits that may result in models too closely tailored to the training data. In contrast, using a higher value forces the tree to generalize more broadly, which can prevent it from capturing relevant patterns in the data [7, 10, 27].

On the other hand, the evolutionary algorithm employed in this study, Particle Swarm Optimization (PSO), provides an evolutionary method for optimization inspired by the collective motion of a group of particles. It is based on the idea that each particle adjusts its trajectory using information from its own previous experience and that of neighboring particles, in order to reach an optimal solution. This method was initially proposed by Kennedy and Eberhart [12], and has since been applied to a wide range of optimization problems. In the present study, PSO is used in a hybrid configuration with a machine learning method based on Random Forest, with the goal of optimizing two key parameters of the algorithm: MinLeafSize and the number of trees. The optimization procedure is illustrated in Table 1, where, $pbest_j$ represents the best position found by particle j; and $gbest_j$ is the global best position found among all particles; c_1 and c_2 are constants, w is a weighting function, and r_1 and r_2 are random values uniformly distributed in [0,1], $x_j^{(k)}$ and $v_j^{(k)}$ are the position and velocity of the particle j at iteration k, respectively.

The decision trees, derived from the Random Forest modeling, were optimized using Particle Swarm Optimization within the framework of this study, which involves complex and deeply branched structures in which multiple sequential splits are performed based on threshold values of predictor variables (x_1, x_2, ..., x_6) to reach a final class assignment. Each node represents a binary condition that partitions the dataset according to a specific threshold, thereby iteratively segmenting the feature space until a prediction is achieved. A complete traversal of the trees uncovers multiple nodes and decision paths, underscoring both the model's ability to capture intricate data patterns and the intrinsic challenge of comprehensively interpreting its decision logic. The results obtained through the procedure described above are presented in the next section.

3.2 Hybridization of PSO and ANN

In this case, hybridization is performed by first applying the evolutionary algorithm (EA), followed by the machine learning tool considered, a Pattern Recognition Artificial Neural Network (*Patternnet*) [28]. This reverse order, compared to the approach used for Random Forest, is adopted because it has been observed that convergence is

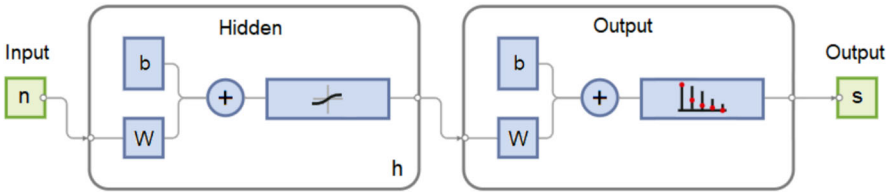

Fig. 1 Patternet neural network employed

faster when the EA is applied first. Grid search, by contrast, is computationally more expensive. It is important to note that the number of parameters chosen for optimization in this case affects the number of hidden neurons in the Patternnet architecture, which is defined as $patternnet(n)$ that is, a single hidden layer with n neurons as shown in Fig. 1.

The optimization could have been extended to other hyperparameters, including the training algorithm, activation functions, or more complex architectures such as $patternnet([n_1, ..., n_m])$, involving multiple hidden layers. Nevertheless, a single-layer configuration is adopted in this study. Figure 1 shows the structure of the ANN employed in this study, where $W_{hidden}(hxn)$ is the weight matrix of the hidden layer, b_{hidden} $(hx1)$ is the bias vector of the hidden layer. Likewise, W_{output} (sxh) is the weight matrix of the output layer and b_{output} is the bias vector of the output layer.

As previously indicated, a PSO algorithm was employed in this study. This method, as described in Table 1, is first employed to optimize the parameters of the neural network, specifically, the number of neurons in the hidden layer, which has been selected as the key hyperparameter in this study. The computation performed by the neural network in the hidden layer is given by Eq. (3), where X is the input vector $(n \times 1)$, a_{hidden} $(hx1)$ is the hidden layer output after the activation function and f_{hidden} is the activation function used, which is hyperbolic tangent sigmoid function as shown in Eq. (4).

$$a_{hidden} = f_{hidden}(W_{hidden} * X + b_{hidden}) \tag{3}$$

$$f_{hidden}(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{4}$$

The computation in the output layer is given by Eqs. (5) and (6). As can be observed in Eq. (5), the activation function of the output layer is of type *softmax*, z_i representing the pre-activation of the $j - th$ output neuron and a_{output} $(sx1)$ is the final output of the network.

$$f_{output}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{s} e^{z_j}} \tag{5}$$

$$a_{output} = f_{output}(W_{output} * a_{hidden} + b_{output}) \tag{6}$$

The algorithm used to optimize the neural network parameters was *trainscg*, which implements a variant of the Scaled Conjugate Gradient (SCG) method [18]. This algorithm updates the weights W_k with the objective of minimizing a differentiable error function $E(W)$, without requiring the explicit computation of the optimal step size, as is the case in the Levenberg-Marquardt method [18]. The weight update rule is given by Eq. (7).

$$W_{k+1} = W_k + \alpha_k * p_k \tag{7}$$

where p_k is the conjugate direction and α_k is a step size computed by approximating the second derivative without explicitly calculating the Hessian matrix, as shown in [18]. The performance metric employed in this current study was the cross entropy [6, 28], which for multiclass classification is defined from Eq. (8).

$$E = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic} log(\widehat{y}_{ic}) \tag{8}$$

where N is the number of samples, C is the number of classes, y_{ic} is equal to 1 if the true class of sample i is class C, and 0 otherwise, and $\widehat{y}_{ic}$ is the predicted probability that sample i belongs to class C, as shown in Bishop [6].

3.3 Metrics for Performance Evaluation

To analyze the different methodologies proposed, confusion matrices will be employed. This approach allows for a graphical representation of the number of correctly predicted classes, as well as the number of misclassifications associated with each fault. The metrics used for each multiple soft fault scenario are shown in Table 2, where TPR stands for True Positive Rate (also known as sensitivity or recall), FNR for False Negative Rate, PPV for Positive Predictive Value (also known as precision), and FDR for False Discovery Rate [14]. Here, TP and TN represent the true positive and true negative results, respectively, while FP and FN denote the false positive and false negative outcomes in each soft fault.

Based on the values obtained from the metrics presented in Table 2, it is possible to evaluate the performance of the method for each class within each classifier. Furthermore, to compare the performance of both classifiers on the training and testing datasets, the global metrics shown in Table 3 are employed. These include global accuracy, global precision, global recall, and global F1-score, and they serve to provide an

Table 2 Classification performance metrics derived from confusion matrix results

TPR	FNR	PPV	FDR
$\frac{TP}{TP+FN}$	$\frac{FN}{TP+FN}$	$\frac{TP}{TP+FP}$	$\frac{FP}{TP+FP}$

Table 3 Global classification performance metrics to compare classifiers

Global accuracy	*Global precision*
$\frac{\sum Correct\ predictions}{N}$	$\frac{\sum_{i=1}^{Num.class} Precision_i}{N}$
Global recall	*Global F1-score*
$\frac{\sum_{i=1}^{Num.class} Recall_i}{N}$	$\frac{1}{N}\sum_{i=1}^{Num.class} 2 * \frac{Precision_i * Recall_i}{Precision_i + Recall_i}$

overall assessment of the classifiers' effectiveness across all classes. Here, N denotes the total number of samples.

Moreover, the Matthews Correlation Coefficient (MCC) [15] will be employed in this analysis. This coefficient is defined by Eq. (9). Its values range within $\pm$ 1, where a higher positive value indicates better classification performance and the contrary, corresponding the zero value to random classification.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{9}$$

4 Feature Extraction

To determine the multiple soft faults that may occur in the circuit, a second-order band-pass filter is selected as the circuit under test (CUT), as shown in Fig. 2. To define the possible faults in the circuit, a 5% tolerance has been set for the components which were selected based on commercially available parts. The use of a 5% tolerance further increases the difficulty of soft fault identification, as it leads to overlapping class responses, thereby complicating the classification task, because the feature extraction process yields results that are often similar across different fault conditions, making classification difficult.

Table 4 shows the variation ranges of the circuit components in relation to their nominal values, that is, the variation ranges related to soft faults. To obtain the Monte Carlo values, it is assumed that the components vary, each within the ranges shown in Table 4, following a uniform distribution law, where the class names will be 'Nominal', and the combinations of pairs of the classes shown in Table 4, that is 'C1_{high}C2_{high}', 'C1_{high}C2_{low}',…'R4_{high}R5_{low}', 'R4_{low}R5_{high}'}. As can be observed, deviations below the allowed values for the nominal values (low) and above them (high) are considered and a total of 83 possible simultaneous classes will be analyzed. As can be observed, this problem is much more complex than that obtained when only one fault of the circuit component is considered and this classification will be addressed by a reduced number of features that will be extracted from the circuit shown in Fig. 2.

Therefore, the possible simultaneous fault events that will be considered in this study are: {'Nominal','C1hC2h','C1hC2l','C1lC2h','C1lC2l','C1hR1h',

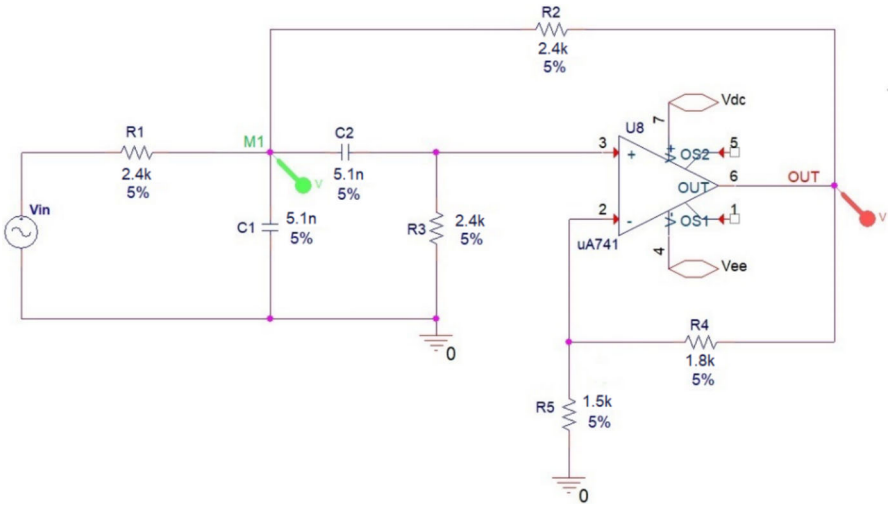

Fig. 2 Electrical diagram of the second order sallen-key band-pass filter employed as the CUT for determining multiple soft faults

Table 4 Variation ranges of the circuit component for multiple soft fault analysis

C1↓	[2.81–3.63] (nF)	R2↑	[3.00–3.48] (kΩ)
C1↑	[6.38–7.40] (nF)	R3↓	[1.32–1.71] (kΩ)
C2↓	[2.81–3.63] (nF)	R3↑	[3.00–3.48] (kΩ)
C2↑	[6.38–7.40] (nF)	R4↓	[0.99–1.28] (kΩ)
R1↓	[1.32–1.71] (kΩ)	R4↑	[2.25–2.61] (kΩ)
R1↑	[3.00–3.48] (kΩ)	R5↓	[0.83–1.07] (kΩ)
R2↓	[1.32–1.71] (kΩ)	R5↑	[1.88–2.18] (kΩ)

’C1hR1l’,’C1lR1h’,’C1lR1l’,’C1hR2h’,’C1hR2l’,’C1lR2h’,’C1lR2l’,’C1hR3h’,’C1hR3l’,’C1lR3h’,’C1lR3l’,’C1hR4h’,’C1hR4l’,’C1lR4h’,’C1lR4l’,’C1hR5h’,’C1hR5l’,’C1lR5h’,’C1lR5l’,’C2hR1h’,’C2hR1l’,’C2lR1h’,’C2lR1l’,’C2hR2h’,’C2hR2l’,’C2lR2h’,’C2lR2l’,’C2hR3h’,’C2hR3l’,’C2lR3h’,’C2lR3l’,’C2hR4h’,’C2hR4l’,’C2lR4h’,’C2lR4l’,’C2hR5h’,’C2hR5l’,’C2lR5h’,’C2lR5l’,’R1hR2h’,’R1hR2l’,’R1lR2h’,’R1lR2l’,’R1hR3h’,’R1hR3l’,’R1lR3h’,’R1lR3l’,’R1hR4h’,’R1hR4l’,’R1lR4h’,’R1lR4l’,’R1hR5h’,’R1hR5l’,’R1lR5h’,’R1lR5l’,’R2hR3h’,’R2hR3l’,’R2lR3h’,’R2lR3l’,’R2hR4h’,’R2hR4l’,’R2lR4h’,’R2lR4l’,’R2hR5h’,’R2hR5l’,’R2lR5h’,’R2lR5l’,’R3hR4h’,’R3hR4l’,’R3lR4h’,’R3lR4l’,’R3hR5h’,’R3hR5l’,’R3lR5h’,’R3lR5l’,’R4hR5l’,’R4lR5h’}, where *l* stands for the lower values and *h* for the higher ones.

A Monte Carlo analysis using Cadence® OrCAD® is carried out a from the values shown in Table 4. From this Monte Carlo analysis, the circuit’s features will be extracted and employed for subsequent classification by using a hybridized method which combines ML and EA. Matlab™ 2022b will be used for developing the hybridized models.

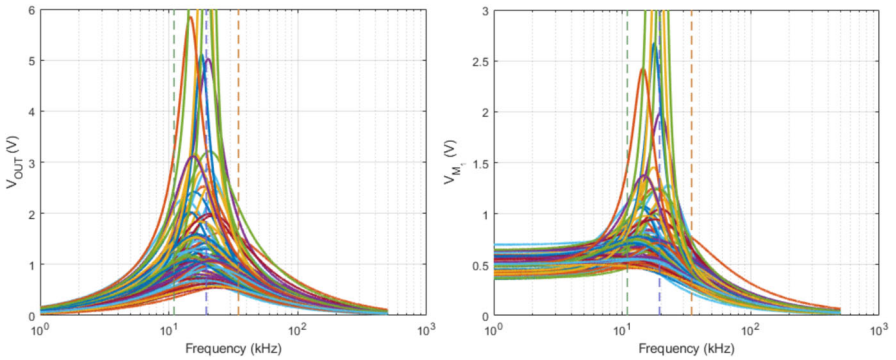

Fig. 3 Circuit responses corresponding to the frequency response of the sallen-key circuit at the selected points OUT and M1, using nominal values and those corresponding to each of the soft faults at their respective central levels

Figure 3 illustrates the extraction of circuit features used to classify the possible soft faults. As can be observed, the proposed feature extraction method is straightforward to implement in practice, as it is based on voltage measurements at two easily accessible points in the circuit: the input and the output, denoted as M1 and OUT, respectively. Voltage values are measured at three frequencies: the nominal frequency and two additional frequencies located at $\pm$ 3 dB relative to the center frequency. Figure 3 shows the values obtained for both the nominal condition and the different soft fault scenarios. These scenarios are defined based on combinations of simultaneous faults, representing the multiple soft fault conditions considered in this study. As shown, the separation between fault conditions is complex, due to a high degree of overlap among the extracted features. Nevertheless, it will be shown that through the combined use of machine learning and evolutionary algorithms, it is possible to achieve a high level of prediction accuracy for the multiple soft faults that may arise in this type of circuit.

Figure 4 presents the results obtained from a Monte Carlo analysis for both the nominal condition and the multiple soft fault scenario given by C1hC2l, at the output (OUT) and input (M1) nodes. The remaining classes are analyzed in a similar manner. According to the procedure described above, the voltage values at each of the three selected frequencies are extracted from these plots. A comparable analysis is conducted for all other multiple soft fault cases.

5 Results and Discussion

This section presents the results of the hybridization between machine learning techniques and evolutionary algorithms, following the methodology described in the previous section, with the objective of detecting and classifying simultaneous faults in electronic circuits.

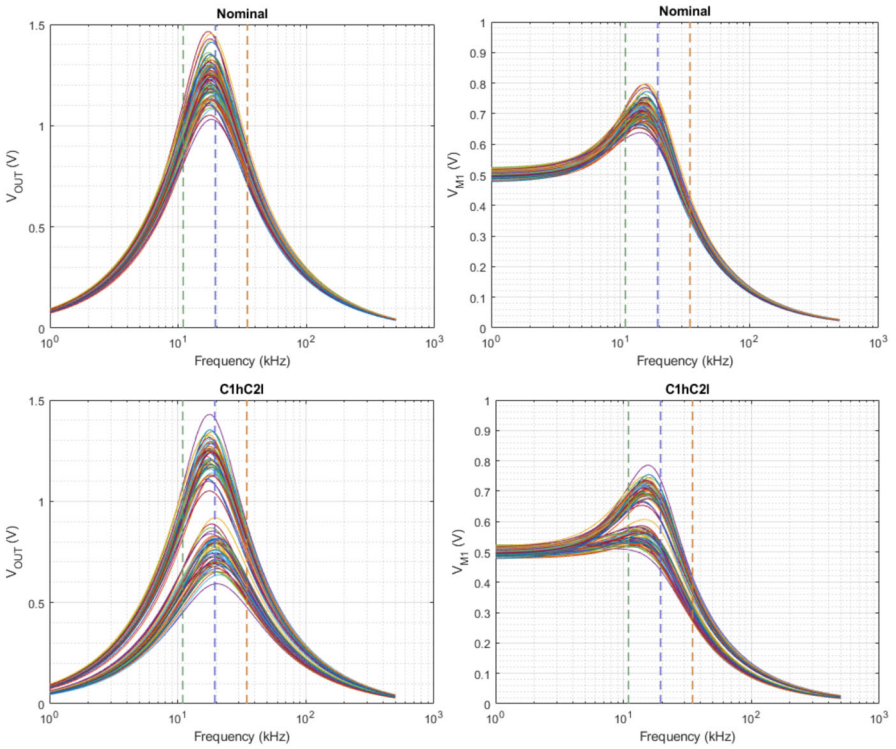

Fig. 4 Results obtained from the monte carlo analysis at the output (V_{OUT}) and at the input (V_{M1}) for the nominal value and for C1hC2l soft fault

5.1 Hybridization of Random Forest and PSO

As was previously mentioned, Random Forest is a supervised algorithm that constructs an ensemble of decision trees trained on datasets generated through random sampling with replacement from the original training set (a technique known as Bagging *which stands for Bootstrap Aggregation*) [10]. In this study, the Random Forest model is implemented using the *"fitcensembl"* function in Matlab™ 2022b. As described in the methodology section, cross-validation and Bayesian optimization are employed in the training process.

Figure 5 shows the evolution of the objective function value as a function of the number of evaluations of the Random Forest model. The graph reveals that, during the initial evaluations, the model exhibited a relatively high error, indicating suboptimal hyperparameter configurations. However, starting from approximately the sixth evaluation, a sharp drop in the objective value is observed, suggesting the discovery of a significantly improved parameter combination. From that point onward, the error remains nearly constant, indicating that no further improvements were achieved. This behavior suggests an early convergence of the model's performance with respect to the evaluated configurations and highlights the sensitivity of Random Forest accuracy

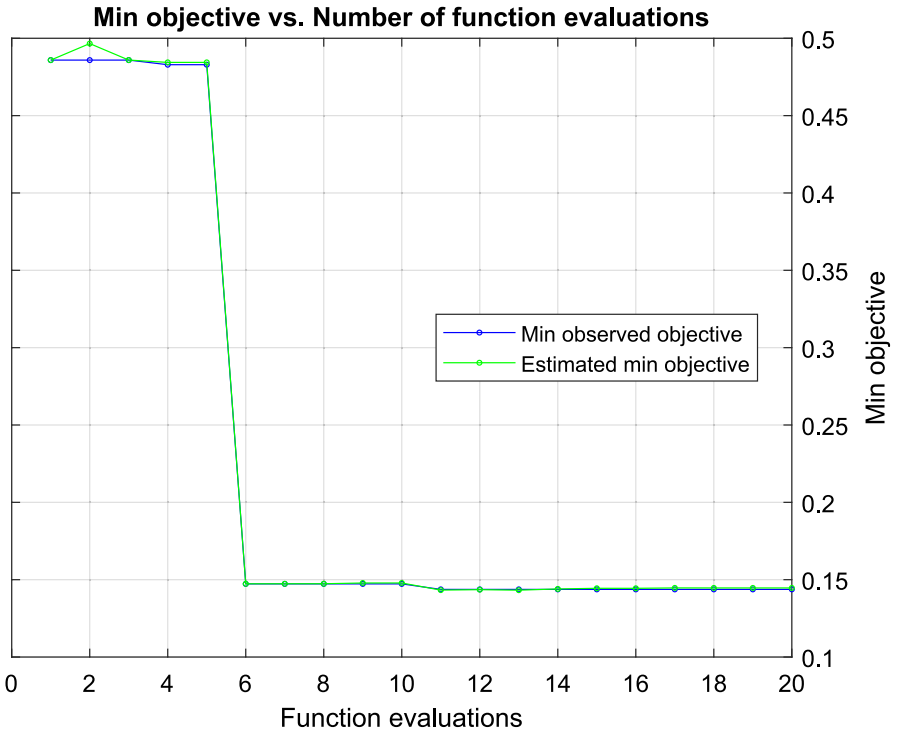

Fig. 5 Convergence of the random forest error during hyperparameter optimization

to its hyperparameters, particularly in the early stages of the search. This observation preliminarily indicates that the evolutionary algorithm, PSO, initialized with the hyperparameter values selected by the Random Forest model is likely to exhibit rapid convergence, given that the machine learning model appears to stabilize around a near-optimal solution. In this study, a 10% variation around the values found by the Random Forest model is considered, with a swarm size of 200 particles, a maximum of 15 iterations, and two variables to optimize: the number of trees and the minimum leaf size. As observed, the convergence of the evolutionary algorithm is remarkably fast, reaching the optimal solution within the first few iterations, as shown in Table 5.

Table 5 shows the progress of the PSO algorithm across several iterations. For each iteration, the total number of objective function evaluations (f-count), the best value achieved so far (Best f(x)), the average function value across all particles (Mean f(x)), and the number of consecutive iterations without improvement (Stall Iterations) are reported.

Based on the above, the initial hyperparameters obtained for the Random Forest model were: MinLeafSize equal to 1 and Number of Trees equal to 499. After executing the PSO algorithm, the optimized values were MinLeafSize equal to 1 and Number of Trees equal to 481. These results led to improved performance on both the training and testing datasets, although the observed variation was relatively small. This outcome

Table 5 Evolution of the objective function during PSO-based optimization

Iteration	f-count	Best f(x)	Mean f(x)	Stall iterations
0	20	0.000140	0.000140	0
1	40	0.000141	0.000141	1
2	60	0.000141	0.000141	2
3	80	0.000142	0.000142	3
⋮	⋮	⋮	⋮	⋮
15	320	0.000142	0.000142	15

Table 6 Comparative results between RF and RF-PSO

	Random Forest [min leaf = 1, num. trees = 499]		Random Forest-PSO [min leaf = 1, num. trees = 481]	
Global metrics	Training	Testing	Training	Testing
Accuracy (%):	98.17%	86.27%	98.24%	86.43%
Precision (%):	98.20%	86.14%	98.28%	86.27%
Recall (%):	98.16%	86.37%	98.24%	86.47%
F1-score (%):	98.17%	85.76%	98.24%	85.90%

is consistent with expectations, as in this current study Random Forest model tends to overfit the training data, and even with a limited number of iterations, the machine learning model achieved high accuracy on the training set. This is illustrated in Table 6, which presents a comparison between the results obtained using Random Forest alone and those obtained using the hybrid RF and PSO approach.

The Random Forest operates as an ensemble model, where the trees perform binary splits based on continuous variables (e.g., x_2, x_3, x_4, etc.) using data-driven thresholds. These rules provide insight into the conditions leading to specific predictions in an individual tree, even though the overall Random Forest model relies on the consensus across all trees in the ensemble. While the tree shown is among the shallower ones in the optimized forest, its structure already exhibits considerable fit to the training data. The global model obtained, composed of 481 trees and trained with a minimum leaf size of 1, indicates a high capacity for fitting. Despite the application of seven fold stratified cross-validation, the large number of trees and highly permissive configuration ($MinLeafSize = 1$) suggest a potential for overfitting.

As shown in Table 6, the accuracy of the Random Forest model decreases when evaluated on an independent dataset, i.e., one not used during training, primarily due to the model's tendency to overfit the training data. As will be discussed later, the use of alternative machine learning tools enables the development of models that yield more

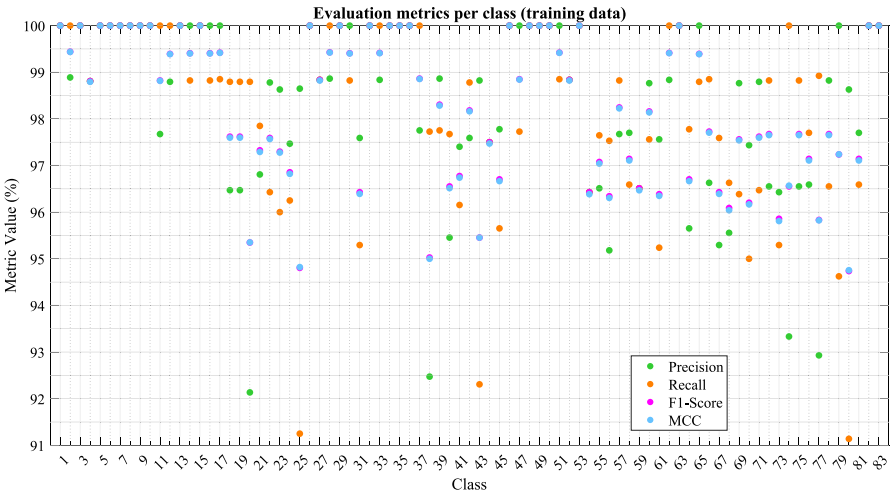

Fig. 6 Class-wise performance on the training set after random forest hyperparameter tuning via PSO

balanced performance between training and testing datasets. In this case, the hybridization of ML with an EA does improve classification accuracy, although the improvement is not substantial. The graph depicted in Fig. 6 confirms that the hybridized model achieves high predictive accuracy on the training data, with most evaluation metrics scoring above 95% across classes. Given the large number of classes, it is not feasible to display all results in a single confusion matrix. Therefore, the results are divided into two groups: classes that were correctly predicted with 100% accuracy, and classes for which some level of misclassification was observed, where the metric values for each fault scenario were calculated using the equations presented in Table 2.

Figures 7 and 8 show the corresponding confusion matrices using the hybrid RF and PSO model. Figure 7 presents the classes that were classified with complete accuracy, while Fig. 8 displays those classes in which at least one instance was misclassified.

Figure 8 shows the classes for which at least one training sample was misclassified, regardless of the number of instances involved. In the case of the Random Forest model, the results obtained on the test set exhibit lower accuracy compared to those on the training set. This suggests that the RF model is not able to generalize effectively, likely due to overfitting during the training process. Table 7 presents the numerical results derived from the confusion chart in Fig. 8, for the misclassified classes in the training dataset.

Figure 9 graphically presents the results obtained using the evaluation metrics applied in this study for the test set. It can be observed that the model's performance on the test set is significantly lower than that achieved on the training set.

5.2 Hybridization of PSO and Artificial Neural Network

In this section, the proposed methodology is applied in the reverse order of that applied to the Random Forest model. Specifically, the evolutionary algorithm (EA) is first used

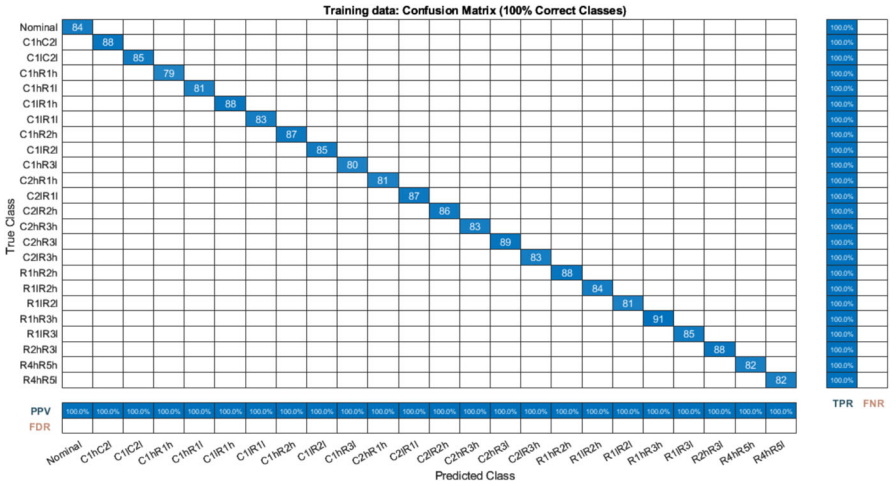

Fig. 7 Confusion matrix for the training data (Random forest and PSO). Correctly classified classes

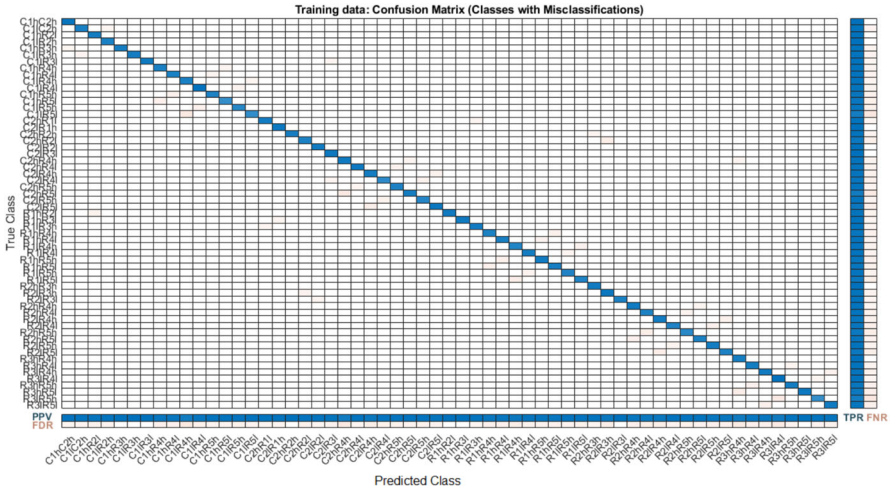

Fig. 8 Confusion matrix for the training data (Random forest and PSO). Incorrectly classified classes

to optimize the hyperparameters of a pattern recognition neural network. Subsequently, a grid search is performed to fine-tune the values obtained through the EA. A comparative analysis is then presented between the results provided by both classifiers. As mentioned in the Methods section, a Patternet-type neural network is employed in this study, as illustrated in Fig. 1. The activation function used in the hidden layer was the hyperbolic tangent sigmoid, while a softmax function was applied at the output layer. Once the base architecture of the neural network was defined, the PSO algorithm was used to optimize the number of neurons in the hidden layer of the Patternnet ANN. Although PSO can be applied to simultaneously tune multiple hyperparameters, such

Table 7 Numerical results of the confusion matrix for the training data (Random Forest and PSO)

Class	C1hC2h	C1lC2h	C1hR2l	C1lR2h	C1hR3h	C1lR3h	C1lR3l	C1hR4h	C1hR4l
TPR	100.00%	98.81%	100.00%	100.00%	98.82%	98.82%	98.85%	98.80%	98.80%
FNR	0.00%	1.19%	0.00%	0.00%	1.18%	1.18%	1.15%	1.20%	1.20%
PPV	98.89%	98.81%	97.67%	98.80%	100.00%	100.00%	100.00%	96.47%	96.47%
FDR	1.11%	1.19%	2.33%	1.20%	0.00%	0.00%	0.00%	3.53%	3.53%
Class	C1lR4h	C1lR4l	C1hR5h	C1hR5l	C1lR5h	C1lR5l	C2hR1l	C2lR1h	C2hR2h
TPR	98.80%	97.85%	96.43%	96.00%	96.25%	91.25%	98.84%	100.00%	98.82%
FNR	1.20%	2.15%	3.57%	4.00%	3.75%	8.75%	1.16%	0.00%	1.18%
PPV	92.13%	96.81%	98.78%	98.63%	97.47%	98.65%	98.84%	98.86%	100.00%
FDR	7.87%	3.19%	1.22%	1.37%	2.53%	1.35%	1.16%	1.14%	0.00%
Class	C2hR2l	C2lR2l	C2lR3l	C2hR4h	C2hR4l	C2lR4h	C2lR4l	C2hR5h	C2hR5l
TPR	95.29%	100.00%	100.00%	97.73%	97.75%	97.67%	96.15%	98.78%	92.31%
FNR	4.71%	0.00%	0.00%	2.27%	2.25%	2.33%	3.85%	1.22%	7.69%
PPV	97.59%	98.84%	97.75%	92.47%	98.86%	95.45%	97.40%	97.59%	98.82%

Table 7 (continued)

Class	C2hR2l	C2lR2l	C2lR3l	C2hR4h	C2hR4l	C2lR4h	C2lR4l	C2hR5h	C2hR5l
FDR	2.41%	1.16%	2.25%	7.53%	1.14%	4.55%	2.60%	2.41%	1.18%
Class	C2lR5h	C2lR5l	R1hR2l	R1hR3l	R1lR3h	R1hR4h	R1hR4l	R1lR4h	R1lR4l
TPR	97.50%	95.65%	97.73%	98.85%	98.84%	96.43%	97.65%	97.53%	98.82%
FNR	2.50%	4.35%	2.27%	1.15%	1.16%	3.57%	2.35%	2.47%	1.18%
PPV	97.50%	97.78%	100.00%	100.00%	98.84%	96.43%	96.51%	95.18%	97.67%
FDR	2.50%	2.22%	0.00%	0.00%	1.16%	3.57%	3.49%	4.82%	2.33%
Class	R1hR5h	R1hR5l	R1lR5h	R1lR5	R2hR3h	R2lR3h	R2lR3l	R2hR4h	R2hR4l
TPR	96.59%	96.51%	97.56%	95.24%	100.00%	97.78%	98.80%	98.85%	97.59%
FNR	3.41%	3.49%	2.44%	4.76%	0.00%	2.22%	1.20%	1.15%	2.41%
PPV	97.70%	96.51%	98.77%	97.56%	98.84%	95.65%	100.00%	96.63%	95.29%
FDR	2.30%	3.49%	1.23%	2.44%	1.16%	4.35%	0.00%	3.37%	4.71%
Class	R2lR4h	R2lR4l	R2hR5h	R2hR5l	R2lR5h	R2lR5l	R3hR4h	R3hR4l	R3lR4h
TPR	96.63%	96.39%	95.00%	96.47%	98.82%	95.29%	100.00%	98.82%	97.70%

Table 7 (continued)

Class	R2lR4h	R2lR4l	R2hR5h	R2hR5l	R2lR5h	R2lR5l	R3hR4h	R3hR4l	R3lR4h
FNR	3.37%	3.61%	5.00%	3.53%	1.18%	4.71%	0.00%	1.18%	2.30%
PPV	95.56%	98.77%	97.44%	98.80%	96.55%	96.43%	93.33%	96.55%	96.59%
FDR	4.44%	1.23%	2.56%	1.20%	3.45%	3.57%	6.67%	3.45%	3.41%

Class	R3lR4l	R3hR5h	R3hR5l	R3lR5h	R3lR5l
TPR	98.92%	96.5%	94.62%	91.14%	96.59%
FNR	1.08%	3.45%	5.38%	8.86%	3.41%
PPV	92.93%	98.82%	100.00%	98.63%	97.70%
FDR	7.07%	1.18%	0.00%	1.37%	2.30%

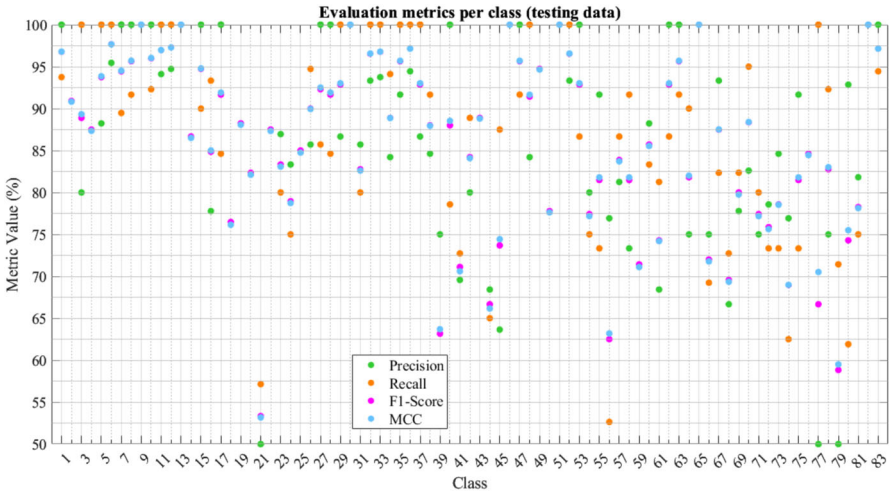

Fig. 9 Class-wise performance on the test set after random forest hyperparameter tuning via PSO

as the number of hidden layers, the search in this study was limited to the number of neurons in a single hidden layer. This parameter was selected due to its significant impact on overfitting. Moreover, this choice substantially reduces the dimensionality of the search space, enabling the evolutionary algorithm to perform a more efficient and stable exploration. For this reason, only one parameter was selected for optimization: the number of neurons in the hidden layer of a Patternnet-type neural network. The lower and upper bounds were set to 8 and 20, respectively, allowing for a reasonably constrained search space and avoiding architectures that are either overly complex or too simplistic. This choice is justified by the relatively large amount of available data, comprising 83 classes with 100 instances per class. The dataset was split into 70% for training, 15% for validation, and 15% for testing, thereby enabling an independent evaluation of model performance. It is important to note that the same test set was used for the Random Forest model, ensuring that both hybrid models (ML & EA) are comparable, as they were trained and validated using equivalent datasets. Regarding the PSO optimizer configuration, a maximum of 5 iterations was selected to promote a fast and efficient exploration of the solution space. Once the optimal number of neurons was identified, this value was used as the starting point for a grid search in which the number of neurons was varied by $\pm$ 10%, as described in the methodology section.

After the iterative process presented in Table 8, the PSO algorithm identified a Patternnet neural network with 20 neurons in the hidden layer, achieving an accuracy of 93.56% on the training dataset and 92.45% on the test dataset. This result will serve as the baseline for a subsequent grid search procedure, in which 10 ANNs will be evaluated per configuration, to determine the most suitable architecture. The configuration that achieved the highest accuracy on the training dataset after applying the initial PSO result and conducting the grid search was the one that used 19 neurons in the hidden layer and reached a training accuracy of 94.26%, outperforming the

Table 8 Optimization of the hidden layer size in a pattern recognition neural network using PSO

Iteration	f-count	Best f(x)	Mean f(x)	Stall iterations
0	5	0.06426	0.06577	0
1	10	0.06218	0.06438	0
2	15	0.06175	0.06349	0
3	20	0.06119	0.06309	0
4	25	0.06119	0.06316	1
5	30	0.06119	0.06296	2

results obtained by the PSO both on the training and test datasets. Figure 10 shows the evolution of the cross-entropy loss for the selected artificial neural network (ANN) after hyperparameter tuning using PSO, followed by refinement through grid search.

The graph depicted in Fig. 11 shows that the hybridized model (PSO and ANN) achieves high predictive accuracy on the training data. However, the performance of the model is inferior to that obtained using the Random Forest combined with PSO. As in the previous case, given the large number of classes, it is not feasible

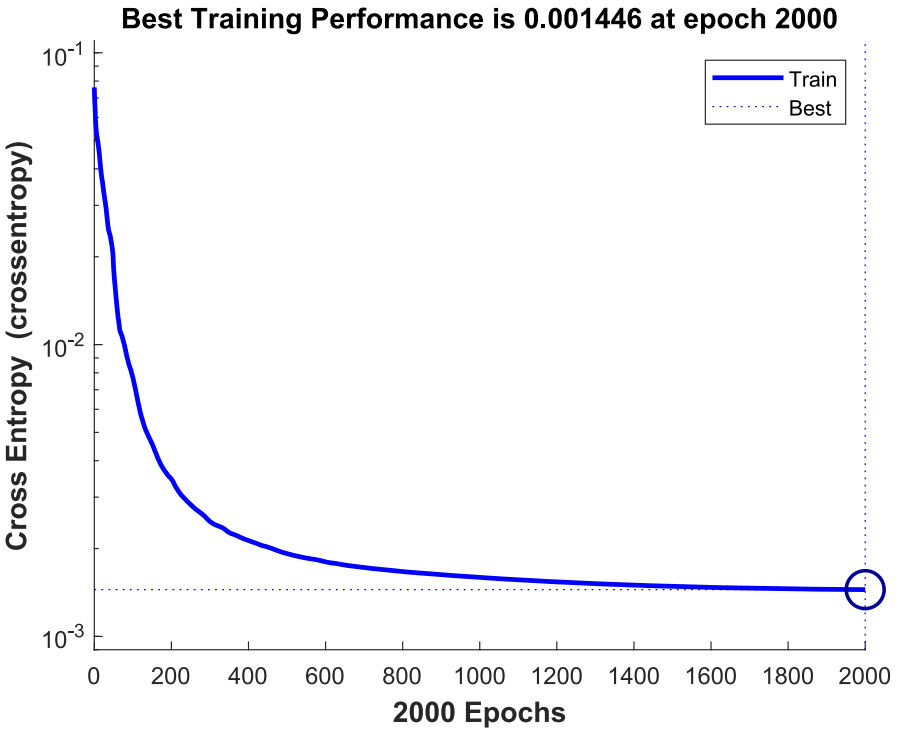

Fig. 10 a Cross-entropy evolution for the selected neural network model

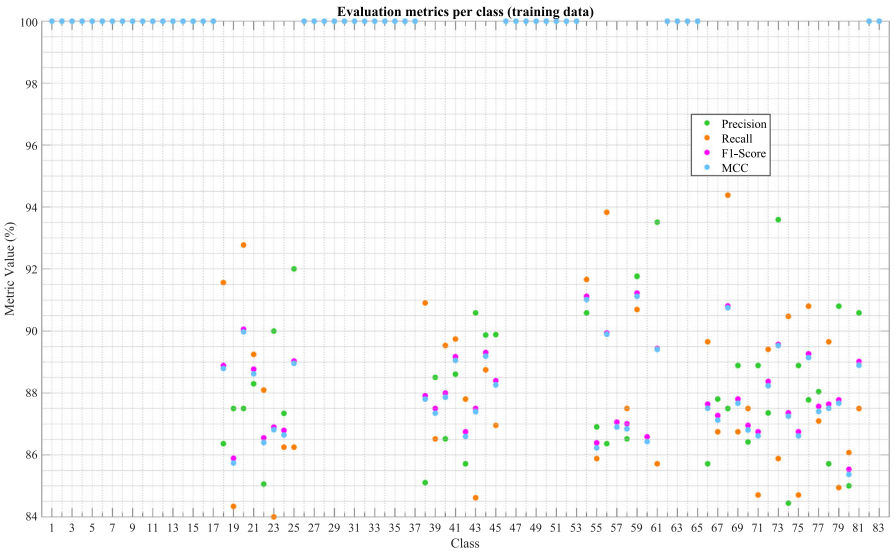

Fig. 11 Class-wise performance on the training set after hybridization of PSO and ANN

to display all results in a single confusion matrix. Therefore, the results are divided into two categories: classes that were correctly predicted with 100% accuracy, and classes for which some degree of misclassification occurred. Figure 12 and Fig. 13 show the confusion matrices generated using the hybrid Patternnet and PSO model, separated accordingly. Figure 12 displays the classes that were perfectly classified

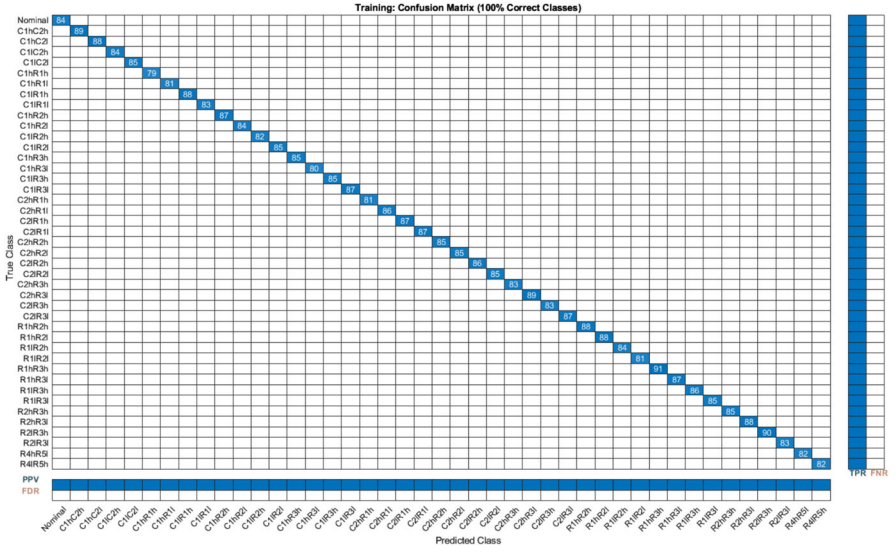

Fig. 12 Confusion matrix for the training data (ANN and PSO). Correctly classified classes

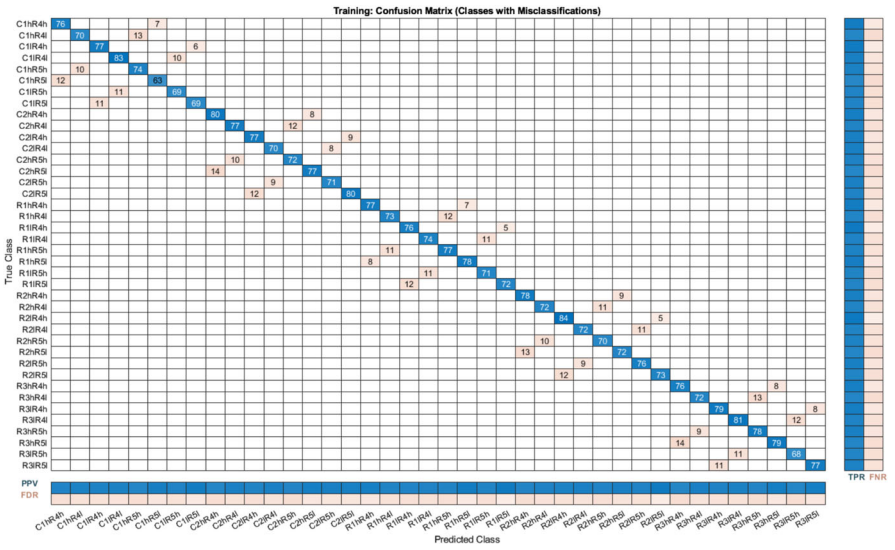

Fig. 13 Confusion matrix for the training data (ANN and PSO). Incorrectly classified classes

(Classes classified with 100% accuracy, i.e., with both PPV and TPR equal to 100%), while Fig. 13 presents those for which at least one instance was misclassified.

Figure 13 shows the classes for which at least one training sample was misclassified, regardless of the number of instances involved, and the corresponding numerical results are presented in Table 9.

Figure 14 graphically presents the results obtained using the evaluation metrics applied in this study for the test set. It can be observed that the model's performance on the test set is lower than that achieved on the training set. It is worth noting that the hybrid model based on the ANN demonstrates more balanced behavior compared to the RF-based model, as it is able to maintain the performance achieved on the training dataset when evaluated on previously unseen data (test dataset).

5.3 Comparative Analysis of the Results Obtained

This section presents a comparative analysis of the two hybrid models evaluated in this study.

Figures 15 and 16 show the per-class evaluation metrics obtained on the training and testing sets, respectively, for the PSO-ANN and RF-PSO classifiers. Each subplot displays the class-wise values of Precision, Recall, F1-Score, and Matthews Correlation Coefficient (MCC), allowing for a detailed comparative performance analysis between the two models.

Figure 15 presents a comparison of the per-class evaluation metrics obtained on the training set for the PSO-ANN and RF-PSO classifiers. Four subplots are shown, corresponding to Precision, Recall, F1-Score, and the Matthews Correlation Coefficient (MCC). Overall, on the training set, the RF-PSO model (dashed line) exhibits

Table 9 Numerical results of the confusion matrix for the training data (ANN and PSO)

Class	C1hR4h	C1hR4l	C1lR4h	C1lR4l	C1hR5h	C1hR5l	C1lR5h	C1lR5l	C2hR4h
TPR	91.57%	84.34%	92.77%	89.25%	88.10%	84.00%	86.25%	86.25%	90.91%
FNR	8.43%	15.66%	7.23%	10.75%	11.90%	16.00%	13.75%	13.75%	9.09%
PPV	86.36%	87.50%	87.50%	88.30%	85.06%	90.00%	87.34%	92.00%	85.11%
FDR	13.64%	12.50%	12.50%	11.70%	14.94%	10.00%	12.66%	8.00%	14.89%
Class	C2hR4l	C2lR4h	C2lR4l	C2hR5h	C2hR5l	C2lR5h	C2lR5l	R1hR4h	R1hR4l
TPR	86.52%	89.53%	89.74%	87.80%	84.62%	88.75%	86.96%	91.67%	85.88%
FNR	13.48%	10.47%	10.26%	12.20%	15.38%	11.25%	13.04%	8.33%	14.12%
PPV	88.51%	86.52%	88.61%	85.71%	90.59%	89.87%	89.89%	90.59%	86.90%
FDR	11.49%	13.48%	11.39%	14.29%	9.41%	10.13%	10.11%	9.41%	13.10%
Class	R1lR4h	R1lR4l	R1hR5h	R1hR5l	R1lR5h	R1lR5l	R2hR4h	R2hR4l	R2lR4h
TPR	93.83%	87.06%	87.50%	90.70%	86.59%	85.71%	89.66%	86.75%	94.38%
FNR	6.17%	12.94%	12.50%	9.30%	13.41%	14.29%	10.34%	13.25%	5.62%
PPV	86.36%	87.06%	86.52%	91.76%	86.59%	93.51%	85.71%	87.80%	87.50%
FDR	13.64%	12.94%	13.48%	8.24%	13.41%	6.49%	14.29%	12.20%	12.50%
Class	R2lR4l	R2hR5h	R2hR5l	R2lR5h	R2lR5l	R3hR4h	R3hR4l	R3lR4h	R3lR4l
TPR	86.75%	87.50%	84.71%	89.41%	85.88%	90.48%	84.71%	90.80%	87.10%
FNR	13.25%	12.50%	15.29%	10.59%	14.12%	9.52%	15.29%	9.20%	12.90%
PPV	88.89%	86.42%	88.89%	87.36%	93.59%	84.44%	88.89%	87.78%	88.04%

Table 9 (continued)

Class	R2lR4l	R2hR5h	R2hR5l	R2lR5h	R2lR5l	R3hR4h	R3hR4l	R3lR4h	R3lR4l
FDR	11.11%	13.58%	11.11%	12.64%	6.41%	15.56%	11.11%	12.22%	11.96%

Class	R3hR5h	R3hR5l	R3lR5h	R3lR5l
TPR	89.66%	84.95%	86.08%	87.50%
FNR	10.34%	15.05%	13.92%	12.50%
PPV	85.71%	90.80%	85.00%	90.59%
FDR	14.29%	9.20%	15.00%	9.41%

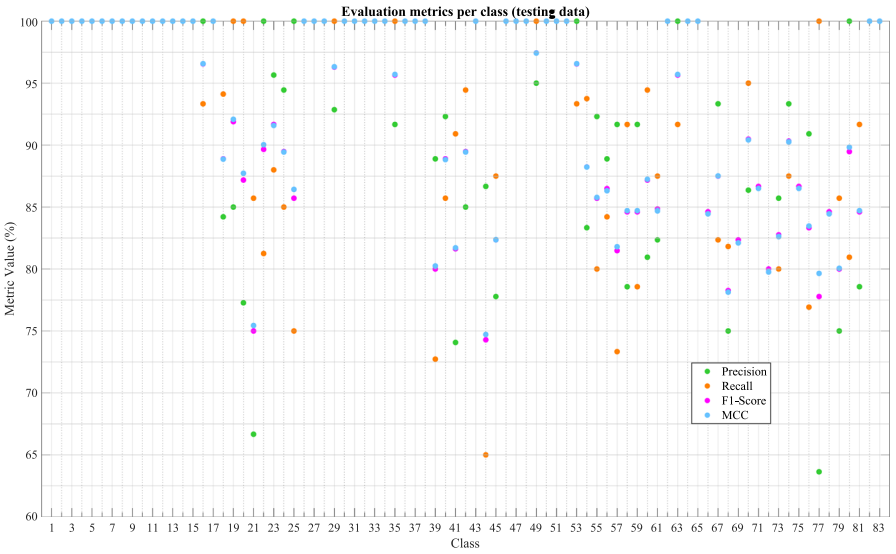

Fig. 14 Class-wise performance on the test set after hybridization of PSO and ANN

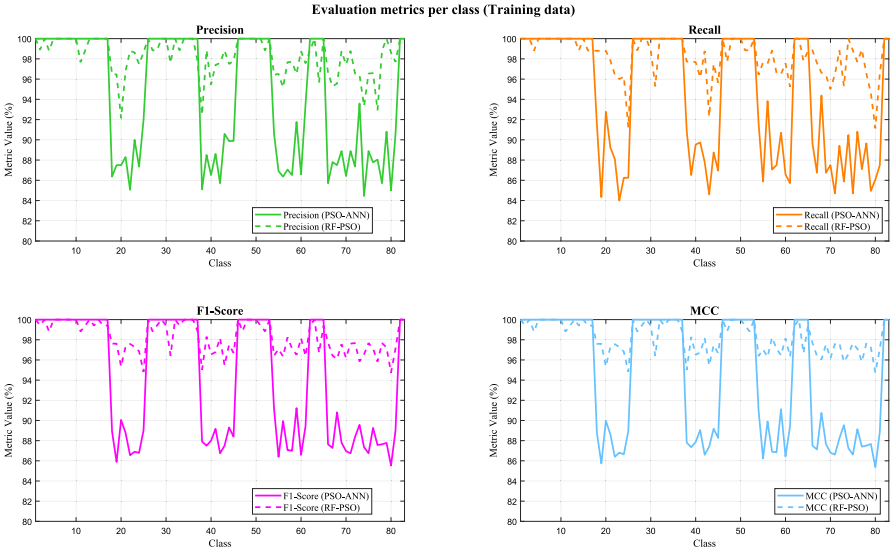

Fig. 15 Class-wise metrics for PSO-ANN and RF-PSO on the training data

a more robust and stable performance across all classes compared to the PSO-ANN model (solid line). In the Precision subplot, RF-PSO consistently maintains high values, whereas PSO-ANN shows notable drops in certain classes. This trend is more pronounced in the Recall subplot, where PSO-ANN displays significantly lower and more variable values than RF-PSO, indicating a reduced ability to correctly identify

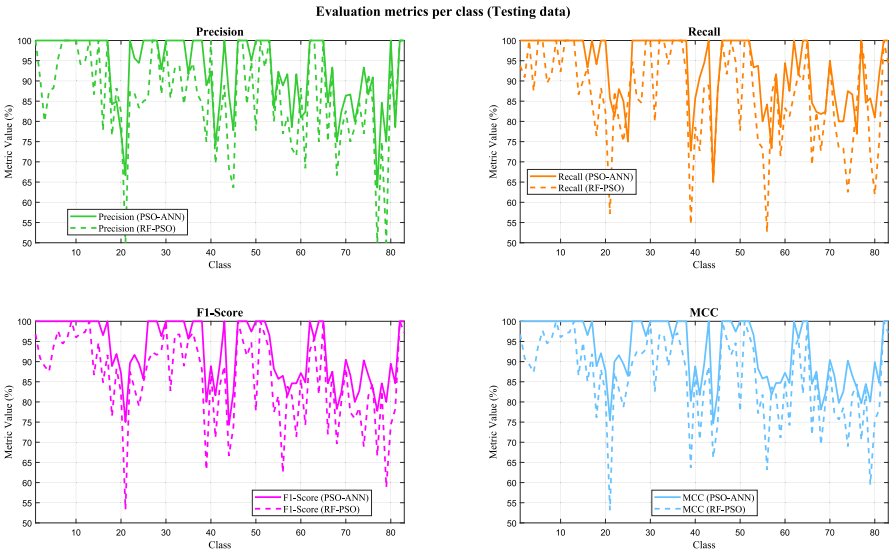

Fig. 16 Class-wise metrics for PSO-ANN and RF-PSO on the testing data

positive instances across all classes. As a result, the F1-Score of PSO-ANN is also negatively affected, particularly in classes where either Precision or Recall is deficient. In contrast, RF-PSO maintains high and stable F1-Scores throughout. Finally, the MCC metric further reinforces this difference: RF-PSO achieves high values for most classes, while PSO-ANN shows considerable drops in several cases. Taken together, these results indicate that RF-PSO delivers a more consistent and reliable performance during training, especially in terms of recall and overall correlation between predictions and actual classes.

However, regarding the test dataset, the results shown in Fig. 16 reveal the opposite behavior. As observed in the Precision subplot, both models achieve generally high levels, but PSO-ANN demonstrates greater consistency across many classes, with fewer significant drops compared to RF-PSO, which shows sharper declines in specific classes. In terms of Recall, RF-PSO tends to yield higher values across most classes, though with greater variability, while PSO-ANN maintains slightly lower but more stable recall values. Regarding the combined metric F1-Score, both models show similar overall performance; however, PSO-ANN performs slightly better in several classes.

A critical aspect to highlight is the detection of Class 1 (Nominal), which is especially relevant in the context of the problem addressed. Hence, any differences in correctly classifying this class should be decisive when selecting the most appropriate model. In this regard, the results show that PSO-ANN is able to correctly detect Class 1 in both training and test sets, maintaining high values in Precision, Recall, and F1-Score. In contrast, RF-PSO suffers a significant performance drop for Class 1, in the test dataset, indicating a reduced capacity to correctly identify nominal instances.

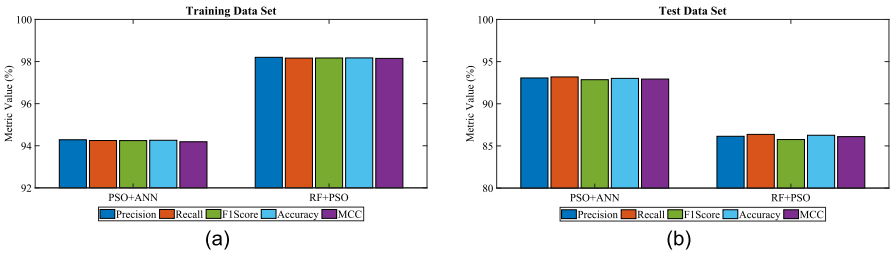

Fig. 17 Global performance metrics for the two evaluated hybridized models **a** Results obtained on the training dataset and **b** Results obtained on the test dataset

In light of these results, it can be concluded that PSO-ANN not only generalizes better but also satisfies the most functionally critical criterion: the accurate detection of Class 1. This capability, consistently maintained across both training and testing phases, constitutes a decisive argument in favor of preferring PSO-ANN over RF-PSO. Detecting multiple soft faults without confusing them with the nominal class is essential in the targeted application, and PSO-ANN proves superior in this respect.

Figure 17 presents a comparative analysis of global performance metrics for the PSO-ANN and RF-PSO models using both the training and test datasets. In the training set, Fig. 17 (a), the RF-PSO model clearly outperforms PSO-ANN across all evaluation metrics, Precision, Recall, F1 Score, Accuracy, and MCC, achieving values consistently above 98%. However, in the test set, Fig. 17 (b), a significant performance drop is observed for RF-PSO, with all metrics falling below those of PSO-ANN. This contrast suggests a potential overfitting issue in the RF-PSO model, which, despite its superior training performance, does not generalize as well to unseen data. In contrast, PSO-ANN demonstrates more balanced behavior, achieving slightly lower training scores but maintaining better generalization on the test data.

Moreover, as derived from the per-class metric evaluation plots, PSO-ANN achieves a higher number of perfectly classified classes (100% accuracy) compared to RF-PSO. The ability to predict a class without error not only prevents confusion with other classes but also enhances model interpretability and reliability, particularly in scenarios sensitive to misclassification. From a practical standpoint, this result further reinforces the suitability of the PSO-ANN model, as a greater proportion of perfectly predicted classes can translate into lower operational risk and fewer critical errors.

6 Conclusions

The present study has demonstrated the viability and effectiveness of hybridizing machine learning (ML) models with evolutionary algorithms (EA) for the detection and classification of simultaneous soft faults in analog circuits. By integrating Particle Swarm Optimization (PSO) with two ML techniques, Random Forest (RF) and a Patternnet-based Artificial Neural Network (ANN), two distinct hybrid approaches were evaluated under controlled conditions using a common dataset and a consistent validation methodology.

The RF-PSO model exhibited superior performance on the training dataset, achieving high and stable global values across all evaluation metrics, particularly Precision (98.28%), Recall (98.24%), F1-Score (98.24%), Accuracy (98.24%), and Matthews Correlation Coefficient (MCC) (98.22%). This indicates that Random Forest, when optimized with PSO, can effectively model the underlying structure of the data. However, the model also showed signs of overfitting, as evidenced by a notable drop in performance when evaluated on the independent test set, with a significant decrease in these overall metrics to 86.27%, 86.47%, 85.90%, 86.43%, and 86.26%, respectively. This suggests that the hybridized model suffers from overfitting.

In contrast, the PSO-ANN model displayed more balanced behavior between training and testing phases. While its training performance was slightly inferior to that of RF-PSO, the ANN maintained a higher generalization capacity and improved stability across most classes in the test set. This was particularly evident in key metrics such as MCC and Precision, where the ANN consistently delivered smoother and more reliable results, even in the presence of class imbalance and fault overlap. The observed values in the training set were 94.28%, 94.25%, 94.24%, 94.26%, and 94.19% for Precision, Recall, F1-Score, Accuracy, and MCC, and 93.06%, 93.18%, 92.85%, 93.01%, and 92.93% in the test set, respectively.

Furthermore, PSO-ANN was able to achieve perfect predictions (100% accuracy) for a higher number of classes than RF-PSO, indicating greater class-level precision and a reduced likelihood of critical misclassification. The hybridized model RF-PSO was able to correctly detect 24 classes in the training set and 7 in the test dataset, while the hybridized model ANN-PSO detected 43 classes in the training dataset and 39 in the test set.

Moreover, it was found that the PSO-ANN model achieved superior performance in detecting Class 1 (Nominal), which plays an important role in the target application. Correctly identifying this class prevents the misclassification of actual faults as nominal states, which is essential to prevent faulty circuits from being classified as correct. The ANN-PSO-based model not only achieved high detection rates for Class 1 in both the training and test sets but also maintained consistent metric values across the entire class distribution.

In conclusion, while RF-PSO proves to be a powerful model in terms of in-sample performance, the PSO-optimized ANN model offers a more generalizable and robust solution for fault classification in analog circuits. Its stability across classes, superior treatment of the nominal condition, and a higher number of perfectly classified classes position it as the preferred approach for the dataset analyzed in this study.

Funding Open access funding provided by FEDER European Funds and the Junta de Castilla y León under the Research and Innovation Strategy for Smart Specialization (RIS3) of Castilla y León 2021-2027. Open access funding provided by the CRUE-CSIC agreement with Springer Nature, the FEDER European Funds, and the Junta de Castilla y León under the Research and Innovation Strategy for Smart Specialization (RIS3) of Castilla y León 2021–2027.

Data Availability Data will be provided upon reasonable request.

Declarations

Conflict of interest The author declares no potential conflict of interest.

References

1. E. Afacan, N. Lourenço, R. Martins, G. Dündar, Review: machine learning techniques in analog/RF integrated circuit design, synthesis, layout, and test. Integration **77**, 113–130 (2021). https://doi.org/10.1016/j.vlsi.2020.11.006
2. A. Arabi, M. Ayad, N. Bourouba, M. Benziane, I. Griche, S.S.M. Ghoneim, E. Ali, M. Elsisi, R.N.R. Ghaly, An efficient method for faults diagnosis in analog circuits based on machine learning classifiers. Alexandria Eng. J. **77**, 109–125 (2023). https://doi.org/10.1016/j.aej.2023.06.090
3. A. Arabi, N. Bourouba, A. Belaout, M. Ayad, An accurate classifier based on adaptive neuro-fuzzy and features selection techniques for fault classification in analog circuits. Integration **64**, 50–59 (2019). https://doi.org/10.1016/j.vlsi.2018.08.001
4. P. Bilski, Hierarchical diagnostics of analog systems based on the ambiguity groups detection. Measurement **119**, 1–10 (2018). https://doi.org/10.1016/j.measurement.2018.01.029
5. D. Binu, B.S. Kariyappa, A survey on fault diagnosis of analog circuits: taxonomy and state of the art. AEU–Int. J. Electron. Commun. **73**, 68–83 (2017). https://doi.org/10.1016/j.aeue.2017.01.002
6. C.M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, NY, 2006)
7. L. Breiman, Bagging predictors. Mach. Learn. **45**, 5–32 (2001)
8. Y. Deng, G. Chai, Soft fault feature extraction in nonlinear analog circuit fault diagnosis. Circuits Syst Signal Process **35**, 4220–4248 (2016). https://doi.org/10.1007/s00034-016-0265-z
9. M.I. Dieste-Velasco, Soft fault diagnosis in analog electronic circuits using supervised machine learning. Integration **104**, 102482 (2025). https://doi.org/10.1016/j.vlsi.2025.102482
10. T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning* (Springer, New York, New York, NY, 2009)
11. S.P. Karthi, K. Kavitha, Detecting and classifying parametric faults in analog circuits using an optimized attention neural networks. Circuits Syst. Signal Process **43**, 5401–5437 (2024). https://doi.org/10.1007/s00034-024-02722-1
12. J. Kennedy,R. Eberhart, Particle swarm optimization in: *Proceedings of ICNN'95 - International Conference on Neural Networks.* IEEE, pp 1942–1948 (2010)
13. A. Kumar, A.P. Singh, Fuzzy classifier for fault diagnosis in analog electronic circuits. ISA Trans. **52**, 816–824 (2013). https://doi.org/10.1016/j.isatra.2013.06.006
14. A.J. Larner, *The 2x2 Matrix* (Springer International Publishing, Cham, 2021)
15. B.W. Matthews, Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim. Biophys. Acta Protein Struct. Mol. Enzymol. **405**, 442–451 (1975). https://doi.org/10.1016/0005-2795(75)90109-9
16. Y. Miao, Y. Zhang, F. Chen, Z. Wang, Analog Circuit Incipient Fault Detection Based on Attention Mechanism and Fully Convolutional Network. IEEE Access (2024). https://doi.org/10.1109/ACCESS.2024.3403908
17. A. Moezi, S.M. Kargar, Simultaneous fault localization and detection of analog circuits using deep learning approach. Comput. Electr. Eng. (2021). https://doi.org/10.1016/j.compeleceng.2021.107162
18. M.F. Møller, A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw. **6**, 525–533 (1993). https://doi.org/10.1016/S0893-6080(05)80056-5

19. A.R. Nasser, A.T. Azar, A.J. Humaidi, A.K. Al-Mhdawi, I.K. Ibraheem, Intelligent fault detection and identification approach for analog electronic circuits based on fuzzy logic classifier. Electronics **10**, 2888 (2021). https://doi.org/10.3390/electronics10232888
20. M. Parai, S. Srimani, K. Ghosh, H. Rahaman, Multi-source data fusion technique for parametric fault diagnosis in analog circuits. Integration **84**, 92–101 (2022). https://doi.org/10.1016/j.vlsi.2022.01.005
21. J. Qu, X. Fang, Y. Chai, Q. Tang, J. Liu, An intermittent fault diagnosis method of analog circuits based on variational modal decomposition and adaptive dynamic density peak clustering. Soft. Comput. **26**, 8603–8615 (2022). https://doi.org/10.1007/s00500-022-07226-1
22. M. Sheikhan, A.A. Sha'bani, PSO-optimized modular neural network trained by OWO-HWO algorithm for fault location in analog circuits. Neural Comput. Appl. **23**, 519–530 (2013). https://doi.org/10.1007/s00521-012-0947-9
23. J. Shi, Y. Deng, Z. Wang, Analog circuit fault diagnosis based on density peaks clustering and dynamic weight probabilistic neural network. Neurocomputing **407**, 354–365 (2020). https://doi.org/10.1016/j.neucom.2020.04.113
24. J. Shi, Y. Deng, Z. Wang, X. Guo, An adaptive new state recognition method based on density peak clustering and voting probabilistic neural network. Appl. Soft Comput. J. **97**, 106835 (2020). https://doi.org/10.1016/j.asoc.2020.106835
25. M. Tadeusiewicz, S. Hałgas, A method for multiple soft fault diagnosis of linear analog circuits. Meas. J. Int. Meas. Confed. **131**, 714–722 (2019). https://doi.org/10.1016/j.measurement.2018.09.001
26. M. Tadeusiewicz, A. Kuczyński, S. Hałgas, Catastrophic fault diagnosis of a certain class of nonlinear analog circuits. Circuits Syst. Signal Process. **34**, 353–375 (2015). https://doi.org/10.1007/s00034-014-9857-7
27. The MathWorks Inc. Statistics and Machine Learning Toolbox User's Guide (R2022b) (2022)
28. The MathWorks Inc. Deep learning Toolbox^{TM} User's Guide (R2022b) (2022)
29. J. Yang, T. Gao, S. Jiang, A dual-input fault diagnosis model based on SE-MSCNN for analog circuits. Appl. Intell. **53**, 7154–7168 (2023). https://doi.org/10.1007/s10489-022-03665-3
30. W. Yu, Y. Sui, J. Wang, The faults diagnostic analysis for analog circuit based on FA-TM-ELM. J. Electron. Test. **32**, 459–465 (2016). https://doi.org/10.1007/s10836-016-5597-x
31. C. Zhang, Y. He, T. Yang, B. Zhang, J. Wu, An analog circuit fault diagnosis approach based on improved wavelet transform and MKELM. Circuits Syst. Signal Process. **41**, 1255–1286 (2022). https://doi.org/10.1007/s00034-021-01842-2
32. T. Zhang, T. Li, Analog circuit soft fault diagnosis utilizing matrix perturbation analysis. Analog Integr. Circuits Signal Process. **100**, 181–192 (2019). https://doi.org/10.1007/s10470-019-01433-x
33. G. Zhao, X. Liu, B. Zhang, Y. Liu, G. Niu, C. Hu, A novel approach for analog circuit fault diagnosis based on deep belief network. Measurement **121**, 170–178 (2018). https://doi.org/10.1016/j.measurement.2018.02.044
34. S. Zhao, X. Liang, L. Wang, H. Zhang, G. Li, J. Chen, A fault diagnosis method for analog circuits based on EEMD-PSO-SVM. Heliyon **10**, e38064 (2024). https://doi.org/10.1016/j.heliyon.2024.e38064
35. T. Zhong, J. Qu, X. Fang, H. Li, Z. Wang, The intermittent fault diagnosis of analog circuits based on EEMD-DBN. Neurocomputing **436**, 74–91 (2021). https://doi.org/10.1016/j.neucom.2021.01.001