Evaluation of Machine Learning Models for Enhancing System Reliability

Jelena Nedeljković
Department of Electronics
University of Niš, Faculty of Electronic Engineering
Niš, Serbia
jelena.nedeljkovic@elfak.ni.ac.rs & https://orcid.org/0009-0002-2547-8935

Goran Nikolić
Department of Electronics
University of Niš, Faculty of Electronic Engineering
Niš, Serbia
goran.nikolic@elfak.ni.ac.rs & https://orcid.org/0000-0002-3745-7878

Marko Anđelković
System Architectures Department
IHP – Leibniz-Institut für innovative Mikroelektronik
Frankfurt (Oder),Germany
andjelkovic@ihp-microelectronics.com
https://orcid.org/0000-0001-6419-2062

Tatjana Nikolić
Department of Electronics
University of Niš, Faculty of Electronic Engineering
Niš, Serbia
tatjana.nikolic@elfak.ni.ac.rs & https://orcid.org/0000-0003-2649-4478

***Abstract*—Ensuring hardware reliability in embedded applications with limited power and resources is a critical challenge, particularly in systems susceptible to computational errors due to hardware faults. Traditional fault tolerance techniques, such as redundancy, have proven effective but often introduce significant computational and power overhead. This paper proposes a machine learning-based approach for real-time error detection in computational blocks executing fixed functions. Using supervised learning methods, the proposed approach enables predictive modeling of hardware behavior, enabling early detection of faults by comparing the outputs of the functional block and the model. Multiple machine learning models are evaluated to determine the most effective approach for error detection. Experimental results demonstrate that polynomial regression achieves the highest accuracy in approximating nonlinear functions, such as the polynomial (power) function x^n, making them ideal for real-time error detection in embedded applications. Polynomial regression achieved a coefficient of determination (R^2) of 1.000 on both training and test datasets, establishing it as the best model.**

Keywords—system reliability, machine learning models, polynomial regression

I. Introduction

Advancements in scaling technology have led to complex, power-efficient applications, but these systems are more prone to hardware faults. Ensuring hardware reliability is a key challenge, particularly in resource-constrained embedded applications. Addressing hardware reliability at the device level is increasingly focused on architectural and algorithmic solutions [1]. The availability of transistors through technology scaling has made redundancy a widely adopted approach to achieving reliability. Traditionally, fault tolerance in systems has been achieved through hardware redundancy, fault diagnosis techniques, and predefined error-handling mechanisms [2]. For example, N-modular redundancy (NMR) is a strategy in which multiple identical modules (*n*) operate in parallel, and the end result is a majority vote over modules [3]. Embedded applications increasingly rely on inference functions to make decisions and build predictive models from past data. Machine learning (ML) plays a key role in embedded signal analysis by using historical data to create robust models, reducing the need for complex analysis. A particularly effective approach for enhancing hardware reliability in ML algorithms is data-driven reliability. The paper [4] addresses the challenge of efficiently running deep neural networks (DNNs) on resource-constrained devices, highlighting improvements in inference accuracy and latency, while [5] demonstrates the effectiveness of predictive ML models in forecasting faults and minimizing downtime. Challenges of implementing ML on ultra-resource-constrained devices are further explored in [6], [7] and [8] optimizing ML applications for embedded platforms. A growing number of papers apply ML for fault detection across various domains, such as induction motors, machinery, lighthouse light sensors, building systems [9], [10], [11], [12]. They demonstrate ML potential to detect faults and improve system reliability. Another significant application of ML in fault detection is in cloud infrastructure, aerospace and healthcare [13], [14]. Additionally, the need for advanced fault detection in VLSI circuits has led to the development of an AI-augmented framework utilizing machine learning and neural networks, with simulation results demonstrating its effectiveness in improving fault coverage, speeding up detection, and offering scalability compared to conventional techniques. [15].

This paper explores using data-driven training to detect and overcome computational errors in fully fixed-function hardware, ensuring maximum energy efficiency. Instead of relying on conventional redundancy methods that duplicate identical functional blocks, our approach introduces an ML-based block that performs the same function as the original computational block. This method is particularly applicable in cases where the ML model provides a simpler implementation compared to the original block.

The paper is structured as follows: Section I introduces the research, outlining its context and objectives. Section II explores ML-based error detection methods. Section III focuses on the selection of ML models for the proposed approach. Section IV presents the experimental results and their analysis. Finally, Section V summarizes the findings.

Identify applicable funding agency here. If none, delete this text box.

ISBN 978-86-6200-031-6

II. Machine Learning-Based Error Detection

Machine learning algorithms enhance resilience by modeling functions based on input-output data relationships. These models detect errors in functional blocks caused by hardware faults. However, the model's impact on system-level reliability depends on its algorithmic structure, particularly during the inference phase. The model itself that implements the inference requires special design attention, similar to a computer accelerator which must now account for computational reliability [16]. ML algorithms allow the construction of high-order analysis models for defined functions of a block or an entire system. In general, such models have compact parameter representations. In this way, the integration of fixed function computational block and ML technologies for real-time error detection is achieved. Figure 1 presents the fundamental classification of fault tolerance techniques [14]. A particular emphasis is placed on the concrete error detection approach, where errors are identified after they have occurred, as opposed to preemptive error detection, which aims to prevent errors before they happen.

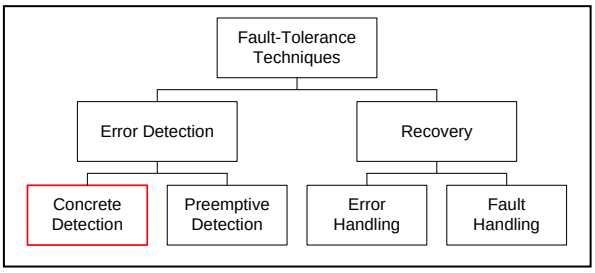

Fig. 1. Fault tolerance techniques

The general block diagram of the proposed ML-based method for increasing system reliability is shown in Figure 2. The functional block represents a part of an embedded system or even an entire system, implemented as a System-on-Chip (SoC) or on an FPGA. This functional block executes a well-defined function, Y = F(X), which may be highly complex, but for each input X, the corresponding output Y is known. The input-output pairs can be represented as integer values or their binary equivalents.

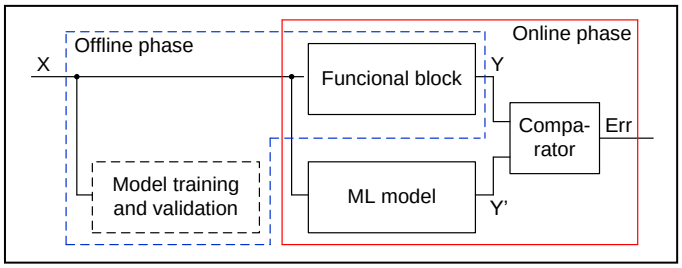

Fig. 2. Error detection system using machine learning

For functions with complex internal dependencies that are difficult to explicitly model, traditional methods such as Look-Up Tables (LUTs) or analytical modeling become inefficient due to excessive memory or computational overhead. In such cases, ML provides an effective alternative by approximating the function without the need for manual rule-based design. The proposed method constructs an ML-based model that replicates the function performed by the functional block. The ML model is designed to generate an output identical to the functional block's output for every given input during system operation. An error detection mechanism is established by introducing a comparator, which continuously compares the outputs of the functional block and the ML model. If a disagreement is detected, an error signal is triggered, indicating potential faults in the system.

The proposed method operates in two distinct phases: 1. Offline phase (training and validation) and 2. Online phase (real-time operation). The offline phase involves the functional block and the model training and validation block, which is depicted with a dashed line in Figure 2. The trained model is then used in the online phase, i.e., during real-time operation. The online phase includes both the functional block and the ML model, which is represented with a solid line in Figure 2.

A. *Offline Phase: Model Training and Validation*

During the offline phase, the ML model is trained using available input-output data from the functional block. Since all input-output pairs are known, they serve as a structured dataset for training. The input is represented as either a bit vector or an integer value, while the corresponding output serves as the target. To ensure that the ML model accurately mimics the functional block, an error function (loss function) is utilized during training. Common loss functions include Mean Squared Error (MSE) – suitable for regression-based outputs. Once trained, the model undergoes validation using a separate dataset that was not included in the training phase. This evaluation ensures generalization and prevents overfitting. Performance metrics such as accuracy (measures correct predictions for discrete outputs) and MSE (assesses the deviation of predicted outputs from actual values) are employed to assess the model's effectiveness.

B. *Online Phase: Real-Time Operation*

In real-time operation, the trained ML model runs in parallel with the functional block. For a given input, both the functional block and the ML model independently generate outputs. The comparator then checks for mismatches. If the outputs differ, an error is flagged, indicating potential hardware faults, transient errors, or unexpected behavior. The ML-based approach for real-time error detection offers the following advantages: a) Lower computational overhead compared to exhaustive rule-based checking; b) Scalability for complex functions with high-dimensional inputs; c) Adaptability to different functional blocks through retraining.

III. Machine Learning Model Selection

This research focuses on the integration of ML technologies for real-time error detection in a computational block that executes a defined function. The aim is to compare different ML methods for real-time error detection in embedded applications. ML can be classified based on multiple factors, including output type and learning protocols, as illustrated in Figure 3. Machine learning approaches can be classified based on the type of output they generate:

1) Supervised Learning: Uses labeled data to learn the mapping between inputs and outputs, enabling predictions for unseen inputs. It includes:

a) Classification: Predicting categorical outputs.

b) Regression: Predicting numerical outputs.

2) Unsupervised Learning: Identifies patterns and structures in unlabeled data.

3) Semi-Supervised Learning: Combines labeled and unlabeled data for training.

4) Reinforcement Learning: Optimizes actions based on rewards and penalties.

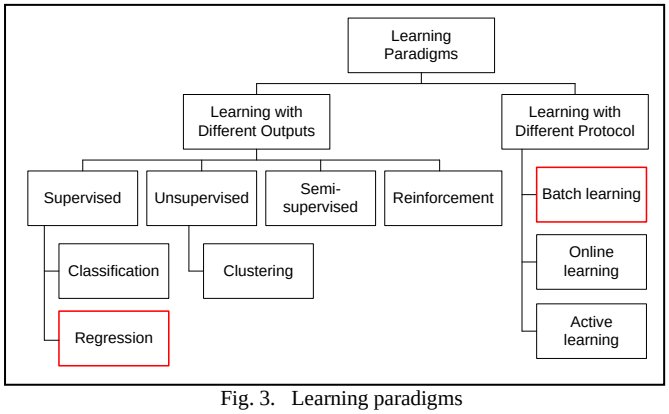

Fig. 3. Learning paradigms

In our research, the model is trained using a dataset with known input and output values, meaning that all the models used fall into the category of supervised learning. Supervised learning is further divided into classification and regression (Figure 3). Since the experiments were conducted using functions that return continuous values (for example, the polynomial function $f(x) = x^n$), the problem is classified as regression. Machine learning approaches can also be classified based on how they process and update data:

1) Batch learning: The model is trained on the entire dataset at once before making predictions. It is suitable for static data and when sufficient computational resources are available.

2) Online learning: The model is iteratively updated with incoming data, making it effective for dynamic environments or when data is not available in advance.

3) Active learning: The model selectively queries experts for labels, focusing on the most informative samples to improve learning efficiency.

All models used in our research fall under Batch learning. To achieve better results, ML techniques have been developed that combine multiple models - ensemble models. These methods reduce errors, variance, and bias, making predictions more robust and accurate. The types of ensemble methods are:

1) Bagging (Bootstrap Aggregation): Multiple models are trained on randomized data with repeated sampling (bootstrap). Each model predicts an output, and then all the outputs are combined to obtain the final result. In this research, the models used from this category are Random Forest Regressor and kNN Regression.

2) Boosting: Models are trained sequentially, with each subsequent model focusing on correcting the errors of the previous one. In the end, all models contribute to the final result, with models that made fewer errors contributing more. In our work, the models used from this category are XGBoost Regressor and Gradient Boosting Regressor.

3) Stacking: This method uses different models and trains a meta-model that takes the predictions from the base models as input. The meta-model is then used for the final prediction. In our research, the models from this category are MLP Regressor and Decision Tree Regression.

4) Voting: This method combines the predictions of several models, using a voting principle. This methodology was not applied in this research.

IV. Experimental Results

To select the most suitable ML model for approximating the defined function, an analysis of 12 different models was conducted. The polynomial function $f(x) = x^n$ was selected due to its simple but nonlinear nature, characterized by even symmetry and a predictable growth pattern. This function allows a clear distinction between models capable of recognizing nonlinear relationships and those limited to linear approximations. Additionally, the polynomial function has wide applications in electronics, including modeling circuit response, characterizing transistors and semiconductor components, and designing filtering systems. Its predictable structure enables an easy assessment of model accuracy, which is crucial for comparing the performance of ML algorithms in the task of approximating nonlinear functions.

A proprietary dataset of 1,138 instances, each consisting of an input value and the corresponding output, was developed for training the models. Training was conducted in the Google Colab environment. Each model was evaluated based on its performance metrics on both training and test datasets. The coefficient of determination (R^2) was chosen as the primary evaluation metric, as it effectively indicates how well the model's predictions match the actual values. This metric measures the proportion of variance in the dependent variable explained by the independent variables, with values ranging from 0 to 1. A higher R^2 signifies better model fit, with a value of 1 indicating perfect predictions. Due to its clarity and reliability, R^2 is one of the most commonly used metrics for evaluating regression models. After conducting the experiments, the R^2 results for different models are shown in Table I. Comparing these results helps highlight the strengths and weaknesses of each model in predicting the accuracy of the polynomial function $f(x) = x^n$.

The tested approaches revealed distinct performance differences, with several models, including XGBoost Regressor, Random Forest Regressor, and Gradient Boosting Regressor, showing exceptionally high accuracy in capturing the nonlinear nature of the polynomial function. These ensemble-based models, utilizing multiple decision trees, enhanced predictive performance. Polynomial Regression, in particular, proved well-suited for this task due to its ability to directly incorporate polynomial terms, making it an ideal choice for modeling nonlinear relationships.

Moderate accuracy was observed in models such as Linear Regression, Bayesian Regression, and Multi-layer Perceptron (MLP) Regressor. Linear and Bayesian Regression models, designed for linear relationships, struggled to accurately represent the nonlinear function. Although MLP Regressor is typically capable of capturing nonlinear dependencies, its performance in this case was lower than expected.

TABLE I. R^2 METRIC

Model	R-squared (Training)	R-squared (Test)
XGBoost Regressor	0.99996	0.99997
Linear Regression	0.938	0.935
Random Forest Regressor	0.999	0.999
Gradient Boosting Regressor	0.999	0.999
Support Vector Regression (SVR)	-0.888	-0.118
MLP Regressor	0.869	0.853
Polynomial Regression	1.000	1.000
K-Nearest Neighbors (k-NN) Regression	0.999	0.999
Bayesian Regression	0.938	0.935
Multi-layer Perceptron (MLP)	0.863	0.853
Linear Regression with Log Transformation	-0.823	-1.346
Decision Tree Regression	1.000	0.999

Some models, particularly Support Vector Regression (SVR) and Linear Regression with Log Transformation showed poor accuracy. While SVR is generally considered effective for nonlinear regression tasks, it failed to yield satisfactory results here. Similarly, applying a logarithmic transformation to Linear Regression did not improve performance, as logarithmic scaling is more appropriate for exponential rather than quadratic relationships.

The results highlight the importance of selecting a model based on the function's specific mathematical properties. While linear models proved insufficient for capturing polynomial dependencies, polynomial regression and ensemble learning techniques provided significantly better results. Surprisingly, models like MLP Regressor and SVR, which are expected to handle nonlinearity well, did not perform as effectively in this case, and the same was observed when these models were tested on an exponential function. These results suggest that each function requires thorough investigation and careful approach to model selection, along with additional hyperparameter tuning or alternative architectures, to achieve optimal performance in nonlinear regression tasks.

V. CONCLUSION

The need for reliable hardware in energy-efficient and resource-constrained applications is growing. This paper presents a novel ML-based methodology for real-time fault detection in embedded computing blocks. By training ML models to replicate the behavior of a functional block, faults can be identified through discrepancies between model outputs and hardware outputs. However, one key aspect is that the ML block must be 100% accurate. Therefore, our proposed solution involves using polynomial regression, which achieved a coefficient of determination (R^2) of 1.000 on both the training and test datasets. This approach is only viable if the ML block is simpler and less resource-demanding than the original functional block. Future work will focus on exploring hardware implementations to assess the practical feasibility and performance of the proposed approach.

ACKNOWLEDGMENT

This research has been supported in part by the European Union, within the program HORIZON-WIDERA-2023-ACCESS-02, under Grant Agreement No. 101160293, and in part by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia [Grant Number 451-03-137/2025-03/ 200102].

REFERENCES

[1] R. Aalund, V. Philip Paglioni, "Enhancing Reliability in Embedded Systems Hardware: A Literature Survey," in *IEEE Access*, vol. 13, pp. 17285-17302, 2025, doi:10.1109/ACCESS.2025.3534138.

[2] M. A. Mukwevho and T. Celik, "Toward a Smart Cloud: A Review of Fault-Tolerance Methods in Cloud Systems," in IEEE Transactions on Services Computing, vol. 14, no. 2, pp. 589-605, 2021, doi:10.1109/TSC.2018.2816644.

[3] J. Baek, J. Baek, J. Yoo, H. Baek, "An N-Modular Redundancy Framework Incorporating Response-Time Analysis on Multiprocessor Platforms" *Symmetry*, 11(8):960, 2019, doi:10.3390/sym11080960.

[4] B. Taylor, V. S. Marco, W. Wolff, Y. Elkhatib, Z. Wang, "Adaptive deep learning model selection on embedded systems", 19th ACM SIGPLAN/SIGBED Int. Conf. on Lang., Compilers, and Tools for Embedded Systems (LCTES), 2018, USA, 31-43, doi:10.1145/3211332.3211336

[5] M. Haroon, Z. A. Siddiqui., M. Husain,, A. Ali, T. Ahmad, "A Proactive Approach to Fault Tolerance Using Predictive Machine Learning Models in Distributed Systems", *Intern. Journ. of Experim. Research and Review*, 44, 208-220, 2024, doi:10.52756/ijerr.2024.v44spl.018

[6] S. S. Saha, S. S. Sandha, M. Srivastava, "Machine Learning for Microcontroller-Class Hardware: A Review," in *IEEE Sensors Journal*, 22(22), pp. 21362-21390, 2022, doi: 10.1109/JSEN.2022.3210773

[7] Haigh, K.Z., Mackay, A.M., Cook, M.R., Lin, L.G., "Machine Learning for Embedded Systems: A Case Study", 2015, https://api.semanticscholar.org/CorpusID:16033444

[8] Branco S, Ferreira AG, Cabral J. Machine Learning in Resource-Scarce Embedded Systems, FPGAs, and End-Devices: A Survey. Electronics. 2019; 8(11):1289. https://doi.org/10.3390/electronics8111289

[9] M. Z. Ali, M. N. S. K. Shabbir, X. Liang, Y. Zhang and T. Hu, "Machine Learning-Based Fault Diagnosis for Single- and Multi-Faults in Induction Motors Using Measured Stator Currents and Vibration Signals," *IEEE Transactions on Industry Applications*, vol. 55, no. 3, pp. 2378-2391, 2019, doi: 10.1109/TIA.2019.2895797

[10] D. Neupane, M. Reda Bouadjenek, R. Dazeley, S. Aryal, "Data-driven machinery fault diagnosis: A comprehensive review," *Neurocomputing*, Vol. 627, 2025, 129588, doi:10.1016/j.neucom.2025.129588

[11] M. Kampouridis, N. Vastardis, G. Rayment. "Using machine learning for fault detection in lighthouse light sensors." *arXiv preprint arXiv:2409.05495* (2024)

[12] W. Nelson, C. Dieckert. 2024. "Machine Learning-Based Automated Fault Detection and Diagnostics in Building Systems" *Energies*, 17, no. 2: 529, doi:10.3390/en17020529

[13] C. Kalaskar, T. Somasundaram, "Fault Tolerance of Cloud Infrastructure with Machine Learning", *Cybernetics and Information Technologies*, Volume 23, No 4: 26-50, 2023, DOI:10.2478/cait-2023-0034

[14] N T. R. Chillapalli, S. Murganoor, "AI and Machine Learning Solutions for Real-Time Fault-Tolerant System Design in Critical Applications", *Cybernetics and Information Technologies*, Vol. 28, No. 1 (2025), DOI:10.52783/anvi.v28.2239

[15] A.K.Ojha, „AI-Augmented Fault Detection and Diagnosis in VLSI Circuits: A Step toward Intelligent Chip Design“ in Journal of Artificial Intelligence, Machine Learning and Neural Network, Vol. 04, No. 06 (2024), DOI: 10.55529/jaimlnn.46.39.46

[16] Z. Wang, K. H. Lee and N. Verma, "Overcoming Computational Errors in Sensing Platforms Through Embedded Machine-Learning Kernels," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 8, pp. 1459-1470, 2015, doi: 10.1109/TVLSI.2014.23421