Improving Machine Learning Anomaly Detection for Safety-Critical RISC-V Automotive Systems

Elio Vinciguerra[1,*], Maurizio Palesi[1] and Giuseppe Ascia[1]

[1]University of Catania, Catania, Italy

Abstract

Modern automotive systems are evolving into complex cyber-physical platforms, where traditional fixed-policy fault recovery mechanisms prove insufficient against sophisticated faults and cyber-attacks. This work presents an anomaly detection framework for RISC-V-based automotive systems, combining Hardware Performance Counters (HPC) with additional hardware metrics to improve detection accuracy under realistic conditions. The methodology is validated by running FreeRTOS workloads on a full-system RISC-V architecture with controlled fault injection using the CHAOS framework. A comparative analysis of sequence-aware and classical machine learning models demonstrates that integrating temporal data significantly enhances detection, with the GRU-Autoencoder showing the best trade-off between performance and computational efficiency for safety-critical scenarios.

Keywords

Anomaly detection, Artificial Intelligence, gem5, RISC-V, Hardware Performance Counters

1. Introduction

The automotive industry's rapid evolution toward sophisticated cyber-physical systems has exposed critical vulnerabilities in conventional safety architectures. Modern vehicles integrate numerous Electronic Control Units (ECUs) within complex networks, creating attack surfaces that traditional fixed-policy recovery mechanisms cannot adequately address [1]. This paradigm shift necessitates the development of adaptive, learning-based resilience frameworks that can dynamically respond to both accidental failures and deliberate security breaches. Contemporary automotive systems must comply with stringent safety standards, particularly ISO 26262, which establishes comprehensive guidelines for Functional Safety (FS) implementation across Electrical and Electronic (E/E) systems [2, 3]. However, the increasing complexity of these systems, coupled with the adoption of emerging technologies such as RISC-V instruction set architectures [4], presents both opportunities and challenges for safety-critical applications. While RISC-V offers enhanced flexibility and performance monitoring capabilities, it also introduces novel attack vectors and failure modes that require innovative detection methodologies.

Building upon the foundational work presented in [5], which demonstrated the feasibility of anomaly detection in RISC-V-based automotive systems through Hardware Performance Counters (HPC) analysis, this research addresses several critical limitations in the current state-of-the-art. The previous approach, while effective in detecting behavioral anomalies through artificial intelligence techniques applied to HPC data, relied on a limited set of performance metrics and synthetic test environments that may not accurately reflect real-world automotive operational conditions.

This investigation advances the field by introducing a comprehensive anomaly detection framework that incorporates multiple complementary metrics beyond traditional HPC. The proposed methodology is validated through realistic benchmark scenarios, including FreeRTOS execution environments, which more accurately represent the operational characteristics of production automotive systems. Through extensive experimental evaluation, this work demonstrates enhanced detection capabilities and provides practical insights for implementing robust anomaly detection systems in safety-critical automotive applications.

CPSWS'25: CPS Summer School PhD Workshop, September 22, 2025, Alghero, Sardinia, Italy

*Corresponding author.

elio.vinciguerra@phd.unict.it (E. Vinciguerra); maurizio.palesi@unict.it (M. Palesi); giuseppe.ascia@unict.it (G. Ascia)

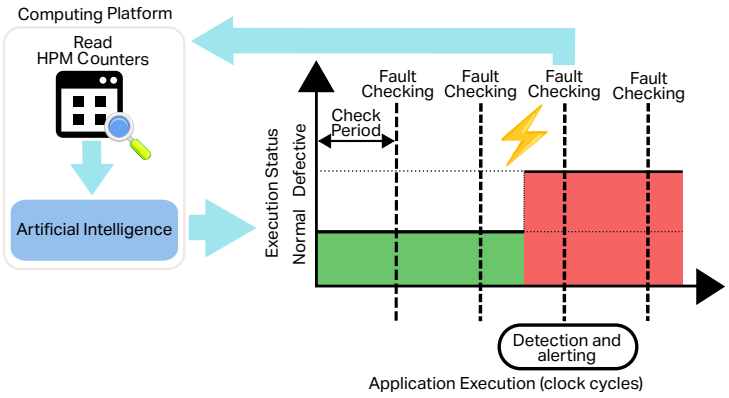

Figure 1: Comprehensive overview of the proposed framework.

2. Proposed Methodology

The framework logic has been presented and analyzed in [5]. At every Check Period (CP), the system state is sent to a computing platform, which uses artificial intelligence models to analyze hardware metrics in order to classify the execution status, as illustrated in Figure 1.

2.1. FreeRTOS Configuration

FreeRTOS is a lightweight, open-source real-time operating system kernel widely adopted in embedded and automotive applications due to its minimal resource footprint and portability across heterogeneous hardware platforms, including RISC-V. Through FreeRTOS, it becomes possible to analyze system behavior in more realistically accurate environments, which is why it has been employed as the operating system executing tasks for the purposes of this work. The selected tasks correspond to the MiBench benchmarks [6] from the Automotive and Industrial Control suite, specifically BasicMath and BitCount. The FreeRTOS application configured in this manner will be simulated in gem5 [7] to obtain the hardware metrics necessary for this work.

2.2. Hardware metrics

In order to classify the execution status as normal or defective, this work employs HPC, following established approaches in the literature [8, 9, 10, 11]. HPC are specialized registers that track various microarchitectural events during program execution, providing low-overhead monitoring capabilities for detecting anomalous behavior patterns. For the purposes of this work, a comprehensive set of RISC-V performance counters is monitored, including basic execution metrics (*mcycle*, *mtime*, *minstret*) and specialized hardware performance events covering instruction execution patterns, memory subsystem behavior, cache performance, and branch prediction accuracy ($mhpmcounter4 - 5$, $mhpmcounter7 - 15$, $mhpmcounter22$, $mhpmcounter27 - 31$). These counters provide detailed insights into arithmetic operations, load/store instructions, cache misses, Translation Lookaside Buffer (TLB) performance, and pipeline behavior. However, the exclusive use of HPC may not be sufficient to fully characterize an execution. To test the effectiveness of machine learning models, a secondary set of metrics is introduced, representing different aspects of the system including cache hierarchy performance, memory controller behavior, branch prediction statistics, instruction throughput, and overall system utilization patterns. The secondary metrics that are most suitable for integration into the system have been extracted through a comprehensive framework that combines four complementary feature selection methods via an ensemble approach. This framework incorporates neural attention weights derived from custom self-attention layers, permutation importance through systematic feature shuffling, mutual information for statistical feature-target dependencies, and cross-dataset stability via bootstrap sampling analysis. The complete mechanism of this system will not be discussed in detail in this work; however, it is

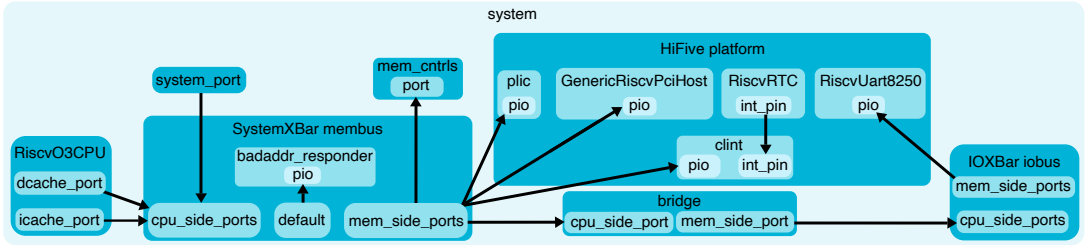

Figure 2: gem5 configuration overview.

inspired by the methodology presented in [12].

2.3. CHAOS

The concrete evaluation of normal or defective execution classification necessitated the use of a fault injector. The Controlled Hardware fAult injectOr System (CHAOS) [13] addresses this need by enabling, through its modular structure, the modeling of faults in CPU registers, cache memory hierarchy, and main memory. CHAOS is a fault injector specifically designed for gem5, and its distinctive feature lies in its modular and open-source nature. The system is organized into three different modules, which offer the ability to inject faults into: CPU registers (CHAOSReg), cache hierarchy (CHAOSCache), and main memory (CHAOSMem). All Instruction Set Architectures (ISAs) and CPU models supported by gem5 are fully compatible with CHAOS. CHAOS provides comprehensive control over fault injection parameters, encompassing fault type and injection methodology, fault occurrence frequency, and temporal distribution patterns. This granular control enables the configuration of highly flexible fault injection campaigns capable of emulating arbitrary fault scenarios and systematically analyzing their consequential impact on system behavior and reliability characteristics.

3. Experiments & Results

This section provides a comprehensive review of experiments designed to determine which machine learning algorithm and hardware metrics set are most suitable for realistically accurate scenarios compared to those presented in [5].

3.1. Experiments Setup

The simulated RISC-V system employs a FullSystem configuration based on the HiFive platform, operating at 1 GHz with a three-level cache hierarchy: 16 KiB L1I, 64 KiB L1D, and 256 KiB L2, backed by 2048 MiB of DDR4 DRAM. An out-of-order (O3) processor in RV64 mode executes the workload with a CP interval of 1,000,000 clock cycles, whose selection rationale is detailed in [5]. Figure 2 illustrates the complete architectural setup adopted for the gem5-based simulation.

CHAOS has been configured to initiate fault injection at a randomly selected clock cycle at the beginning of each simulation, with fault probabilities ranging from 1×10^{-1} to 1×10^{-8} and a random number of bits to alter using a random fault mask.

Furthermore, to diversify FreeRTOS executions and present results based on increasingly realistic scenarios, the parameters of BasicMath and BitCount tasks have been randomized at compile time. Each fault-free simulation operates with a distinct parameter set.

3.2. Machine Learning Techniques

The identification of the most effective machine learning approach for anomaly detection necessitated an analysis across multiple algorithmic paradigms. Eight distinct methodologies were evaluated: Isolation Forest, K-means clustering, K-nearest neighbors (KNN), Local Outlier Factor (LOF), Support

Table 1
Accuracy and F1 Score of various Machine Learning Algorithms Using HPC Only.

	Isolation Forest		K-Means		KNN		LOF		SVM		LSTM-Aut		GRU-Aut		MLP	
	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS
Fault Free [%]	94.99	95.01	94.99	90.58	89.99	95.0	94.99	92.89	95.0	95.0	94.99	\\	94.99	\\	100.0	100.0
CPU Registers [%]	81.37	84.69	83.79	80.79	84.64	85.28	85.38	39.57	85.34	85.71	84.11	\\	83.97	\\	35.43	69.5
L1D Cache [%]	94.48	97.74	97.79	94.58	98.54	98.5	99.13	27.99	98.05	97.7	98.26	\\	98.2	\\	25.54	90.27
L1I Cache [%]	94.69	97.61	97.64	94.36	98.4	98.34	99.05	28.16	98.07	97.58	98.12	\\	98.03	\\	25.78	98.64
L2 Cache [%]	91.6	93.94	93.46	86.37	95.09	95.58	96.42	63.48	94.05	93.66	94.03	\\	93.84	\\	65.2	95.21
Main Memory [%]	30.0	35.51	30.9	32.18	30.49	34.66	32.44	37.58	31.81	35.67	31.41	\\	31.33	\\	30.67	35.46
Overall F1 Score [0-1]	0.8	0.82	0.82	0.79	0.82	0.83	0.84	0.05	0.83	0.82	0.83	\\	0.83	\\	0	0.6

Table 2
Accuracy and F1 Score of Various Machine Learning Algorithms on an Extended Set of Hardware Metrics.

	Isolation Forest		K-Means		KNN		LOF		SVM		LSTM-Aut		GRU-Aut		MLP	
	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS
Fault Free [%]	94.99	95.01	94.99	90.18	89.99	95.0	94.99	92.88	95.0	95.01	95.0	\\	95.01	\\	100.0	100.0
CPU Registers [%]	77.32	80.97	83.21	80.43	84.99	86.14	85.21	39.86	85.38	86.44	83.85	\\	83.95	\\	35.43	39.51
L1D Cache [%]	88.12	92.17	97.38	94.27	98.53	98.85	98.81	32.5	98.09	98.15	98.02	\\	98.15	\\	25.54	29.4
L1I Cache [%]	92.47	96.78	97.25	94.06	98.39	98.71	98.72	31.7	98.09	98.06	97.87	\\	98.01	\\	25.77	29.97
L2 Cache [%]	87.69	90.93	92.6	85.68	95.07	96.44	95.75	64.53	94.25	94.75	93.4	\\	93.78	\\	65.2	67.51
Main Memory [%]	28.35	33.99	57.15	56.56	71.72	85.01	77.65	37.6	59.2	73.14	59.31	\\	71.22	\\	30.67	35.26
Overall F1 Score [0-1]	0.77	0.79	0.87	0.83	0.9	0.93	0.92	0.11	0.88	0.9	0.88	\\	0.9	\\	0	0.01

Vector Machine (SVM), Long Short-Term Memory (LSTM) Autoencoder, Gated Recurrent Units (GRU) Autoencoder, and the Multilayer Perceptron (MLP) architecture detailed in [14, 15, 16, 17]. This selection reflects established practices within the field [14, 15, 16, 17], ensuring methodological alignment with current research standards. Each algorithm was deployed in its one-class variant, a configuration that enables comprehensive pattern recognition across varied operational conditions while facilitating robust learning of nominal system behavior without requiring labeled anomalous examples during the training phase.

The experimental dataset comprised 500 normal execution traces complemented by 14,000 synthetically generated faulty instances. These defective samples were systematically created through randomized fault injection across different temporal points and hardware abstraction layers within the CHAOS framework, ensuring comprehensive fault space coverage. This unsupervised learning approach allows the models to internalize the fundamental characteristics of correct system operation, subsequently enabling detection of any deviation from expected behavior patterns. Consequently, the trained models can identify novel anomalies without prior exposure to specific fault signatures, demonstrating superior generalization capabilities compared to supervised approaches constrained by predefined fault categories.

Table 1 summarizes the results obtained by exclusively leveraging HPC metrics as input features. Each algorithm was evaluated in two distinct configurations: a sequenced (S) variant, where the input consists of a temporal window of 14 consecutive multivariate HPC samples; and a non-sequenced (NS) variant, where the input is limited to a single multivariate sample. Notably, certain models–such as the LSTM-Autoencoder and GRU-Autoencoder–are inherently designed to process sequential data and therefore cannot be applied in the non-sequenced setting. In addition, for each algorithm we analyzed how detection accuracy varies depending on the specific architectural component affected by fault injection. Overall, results indicate that, except for the LOF algorithm in its non-sequenced variant and the MLP in its sequenced variant, most approaches achieve reasonably acceptable accuracy. However, it is worth highlighting that none of the models shows satisfactory performance when faults are injected into main memory, which remains the most challenging scenario.

To enhance detection performance in scenarios where faults are injected into the main memory, an additional set of hardware metrics–introduced in Section 2.2–has been integrated into the analysis. The resulting performance improvements are reported in Table 2. The data demonstrate that incorporating additional hardware metrics beyond HPC alone yields comparable average accuracies across nearly all experimental classes, with the notable exception of the non-sequenced MLP, which exhibits significant performance degradation. However, this mechanism enables improved accuracy for faults injected into

main memory, resulting in satisfactory solutions. Contrary to the results obtained using only HPC (Table 1), the present analysis with an extended set of metrics (Table 2) reveals a marked difference between sequenced and non-sequenced configurations. This divergence can be attributed to the intrinsic characteristics of the algorithms employed. Traditional machine learning algorithms–Isolation Forest, K-Means, KNN, LOF, SVM, and MLP–were originally designed for static data processing and lack inherent mechanisms for handling temporal sequences. When the number of features per sequence is increased, these algorithms exhibit performance degradation in processing sequential inputs, as they are unable to effectively capture temporal dependencies in the data. Conversely, algorithms based on recurrent neural networks–LSTM-Autoencoder and GRU-Autoencoder–are specifically designed for temporal sequence processing and demonstrate greater robustness in handling extended sequential data. Their architecture enables the retention and utilization of previous temporal information, resulting in more stable performance regardless of the data presentation modality. Among the solutions that best adapt to the scope of the present work from an accuracy perspective are the KNN algorithms (both sequenced and non-sequenced variants), LOF (sequenced version), SVM (non-sequenced version), and the GRU-Autoencoder.

Given the stringent computational constraints inherent to automotive applications, the selection of algorithmically efficient mechanisms becomes paramount. Consequently, the evaluation framework must encompass not merely predictive accuracy but also the computational complexity profiles of candidate algorithms to ensure real-time feasibility in resource-constrained environments. The complexity analysis reveals distinct computational signatures: KNN operates with $\mathcal{O}(n \cdot d)$, complexity, where d represents the dataset cardinality and n the feature dimensionality. LOF demonstrates quadratic scaling at $\mathcal{O}(d^2 \cdot n)$, attributed to the intensive pairwise distance computations required for local density estimation. In contrast, SVM exhibits $\mathcal{O}(n \cdot v)$ complexity, with v denoting the support vector cardinality—a parameter typically orders of magnitude smaller than the full dataset. The GRU-Autoencoder exhibits a complexity of $\mathcal{O}(L \cdot (h^2 + h \cdot i))$, where L represents the sequence length, h denotes the hidden state dimension, and i represents the input dimension. Given that in our case $h \gg i$, this expression simplifies to $\mathcal{O}(L \cdot h^2)$, since each GRU cell requires $\mathcal{O}(h^2)$ operations for the hidden-to-hidden transformations at each time step, and this quadratic term dominates the linear input-to-hidden term $\mathcal{O}(h \cdot i)$.

The GRU-Autoencoder is favored over traditional methods such as KNN, LOF, and SVM for sequential temporal data due to its capability to capture temporal dependencies and evolutionary patterns that these conventional approaches, which rely on static local density estimation or geometric separation principles, are inherently unable to detect. While KNN and LOF operate under the assumption of point independence with computational complexity directly proportional to dataset size, and SVM, despite also assuming point independence, achieves inference complexity dependent on support vectors rather than full dataset size, the GRU-Autoencoder demonstrates superior scalability by operating with complexity dependent on sequence length and embedding dimensionality, thereby achieving enhanced computational efficiency for inference on sequential data where temporal anomalies necessitate a sequence-aware methodology rather than conventional point-wise data analysis.

4. Conclusions

This work has presented an adaptive anomaly detection framework for RISC-V-based automotive systems, addressing the limitations of traditional fixed-policy fault recovery by exploiting both HPC and complementary hardware metrics. Through realistic simulations using FreeRTOS workloads and controlled fault injection, we demonstrated that sequential models—particularly the GRU-Autoencoder—can effectively capture temporal execution patterns, improving detection accuracy even in challenging scenarios like faults injected into main memory. While classical methods such as KNN, LOF, and SVM remain competitive, our results show that sequence-aware approaches offer a better balance between accuracy and computational efficiency for safety-critical applications. As next steps, future work will focus on integrating concrete fault mitigation and recovery strategies into the framework, moving from pure detection to active resilience.

Acknowledgments

This work has been (partially) supported by the Spoke 1 "FutureHPC & BigData" of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 - Next Generation EU (NGEU) and MUR PRIN COLTRANE-V E53D23008060006.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT-4o in order to: Paraphrase and reword, Grammar and spelling check. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] S. Kim, K.-J. Park, C. Lu, A survey on network security for cyber–physical systems: From threats to resilient design, IEEE Communications Surveys & Tutorials 24 (2022) 1534–1573. doi:10.1109/COMST.2022.3187531.

[2] SAE, Automated vehicles: The role of iso 26262, in: The Role of ISO 26262, 2020.

[3] International Organization for Standardization, ISO/FDIS 26262, ISO-26262 Road Vehicles - Functional Safety, International Standard, International Organization for Standardization, 2011.

[4] A. Waterman, Y. Lee, R. Avizienis, H. Cook, D. Patterson, K. Asanovic, The risc-v instruction set, in: 2013 IEEE Hot Chips 25 Symposium (HCS), 2013, pp. 1–1. doi:10.1109/HOTCHIPS.2013.7478332.

[5] E. Vinciguerra, E. Russo, M. Palesi, G. Ascia, Data-driven simulation based fault detection in automotive risc-v applications, in: 2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2024, pp. 509–516. doi:10.1109/MCSoC64144.2024.00089.

[6] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, R. B. Brown, Mibench: A free, commercially representative embedded benchmark suite, in: IEEE 4th Annual Workshop on Workload Characterization, Austin, TX, 2001.

[7] J. L.-P. et al., The gem5 simulator: Version 20.0+, 2020. arXiv:2007.03152.

[8] P. M. Anand, P. V. S. Charan, S. K. Shukla, Hiper - early detection of a ransomware attack using hardware performance counters, Digital Threats 4 (2023). URL: https://doi.org/10.1145/3608484. doi:10.1145/3608484.

[9] M. El Bouazzati, P. Tanguy, G. Gogniat, R. Tessier, Diwall: A lightweight host intrusion detection system against jamming and packet injection attacks, ACM Trans. Embed. Comput. Syst. (2025). URL: https://doi.org/10.1145/3711833. doi:10.1145/3711833, just Accepted.

[10] A. Palumbo, R. Salvador, Leveraging gem5 for hardware trojan research: Simulation for machine-learning-based detection, in: Proceedings of the 22nd ACM International Conference on Computing Frontiers: Workshops and Special Sessions, CF '25 Companion, Association for Computing Machinery, New York, NY, USA, 2025, p. 9–16. URL: https://doi.org/10.1145/3706594.3728869. doi:10.1145/3706594.3728869.

[11] A.-T. Le, T.-T. Hoang, B.-A. Dao, A. Tsukamoto, K. Suzaki, C.-K. Pham, A real-time cache side-channel attack detection system on risc-v out-of-order processor, IEEE Access 9 (2021) 164597–164612. doi:10.1109/ACCESS.2021.3134256.

[12] K. K. Abrokwa, Q. Jiang, Z. Ma, Y. Zhang, Explainable performance anomaly detection and rectification in sixth generation open radio access network based on contrastive and deep reinforcement learning, Available at SSRN (2024). URL: https://ssrn.com/abstract=5171031. doi:10.2139/ssrn.5171031.

[13] E. Vinciguerra, E. Russo, M. Palesi, Chaos, https://github.com/eliovinciguerra/CHAOS.git, 2024. URL: https://github.com/eliovinciguerra/CHAOS.git.

[14] K. Zhan, C. Wang, X. Zheng, et al., Seq2seq-based gru autoencoder for anomaly detection and

failure identification in coal mining hydraulic support systems, Scientific Reports 15 (2025) 542. doi:10.1038/s41598-024-84130-8.

[15] B. Aksar, E. Sencan, B. Schwaller, O. Aaziz, V. J. Leung, J. Brandt, B. Kulis, M. Egele, A. K. Coskun, Runtime performance anomaly diagnosis in production hpc systems using active learning, IEEE Transactions on Parallel and Distributed Systems 35 (2024) 693–706. doi:10.1109/TPDS.2024.3365462.

[16] E. Altulaihan, M. A. Almaiah, A. Aljughaiman, Anomaly detection ids for detecting dos attacks in iot networks based on machine learning algorithms, Sensors 24 (2024). URL: https://www.mdpi.com/1424-8220/24/2/713. doi:10.3390/s24020713.

[17] G. Princz, M. Shaloo, S. Erol, Anomaly detection in binary time series data: An unsupervised machine learning approach for condition monitoring, Procedia Computer Science 232 (2024) 1065–1078. URL: https://www.sciencedirect.com/science/article/pii/S1877050924001054. doi:https://doi.org/10.1016/j.procs.2024.01.105, 5th International Conference on Industry 4.0 and Smart Manufacturing (ISM 2023).