International Journal of Engineering & Technology
ISSN: 3083-9114
Journal homepage: https://e-journal.scholar-publishing.org/index.php/ijet

AI-Adaptive Fault Prediction on RISC-V Edge Processors Using Lightweight Hardware Counters

Cut Fadhilah[1], Humasak Simajuntak[2], Feri Susilawati[3]

[1]Faculty of Computer and Multimedia, Universitas Islam Kebangsaan Indonesia, Aceh 24251, Indonesia
[2]Departemen of Information System, Institut Teknologi Del, Sumatera Utara, Medan 22381, Indonesia
[3]Department of Informatics Engineering, Aceh Polytechnic, Aceh, Indonesia

Corresponding Author: cutfadhilahzakaria@gmail.com

Abstract
Edge-class RISC-V processors increasingly operate in safety-critical and resource-limited environments where intermittent hardware faults threaten system reliability. This study proposes a co-designed AI-adaptive fault prediction framework that relies exclusively on lightweight hardware performance counters (HPCs) to enable real-time failure awareness with minimal overhead. The research aims to (1) validate the feasibility of on-device adaptive learning, (2) maintain ultra-low-latency inference, and (3) identify the most informative HPC features for early fault sensitivity under realistic class imbalance. Runtime counter traces and labelled fault logs were generated through controlled fault injection and continuous sampling. Temporal features were extracted using sliding-window buffers and normalised before training lightweight ML models that meet TinyML constraints. The framework achieved a fault inference latency of 5 ms, and adaptive learning improved classification accuracy from 85% to 95% across five incremental training rounds (+10 pp). Feature weight analysis showed that Cycles (0.40) and CacheMiss (0.25) provided the strongest fault discrimination, followed by BranchMis predict (0.20) and Stall events (0.15). Counter-signal behaviour exhibited a wider anomaly amplitude of approximately ±1.9 under fault conditions versus ±1.1 in normal states, enabling detection sensitivity despite a 92% normal and 8% fault sample ratio. Evaluation confirms that reliable fault intelligence for RISC-V edge silicon is achievable with a minimal HPC setup, without cloud-heavy retraining. The study concludes that the proposed pipeline supports deployable, adaptive, and ultra-low-overhead fault prediction, improving edge processor dependability while preserving compute, memory, and latency budgets.

Article Info
Received: 13 November 2025
Revised: 12 December 2025
Accepted: 15 December 2025
Available online: 27 December 2025

Keywords
RISC-V Edge Reliability
Lightweight Hardware Counters
Fault Prediction
TinyML Adaptive Learning
Hardware-AI Co-Design

1. Introduction

The rapid expansion of edge computing has pushed processor architectures toward ultra-low-power, open, and scalable instruction sets, among which RISC-V has emerged as a dominant choice due to its modular ISA, customisation flexibility, and strong adoption in embedded intelligence pipelines (Cui,

Li, & Wei, 2023; Radford, 2025; D. Xu et al., 2021). However, as edge processors operate in unreliable field environments including voltage noise, thermal drift, memory ageing, and clock instability, hardware faults and silent data corruption remain key challenges that degrade system reliability (Ehret, Rosario, Gettings, & Kinsy, 2020; Hou et al., 2021; Moghaddasi, Nasab, & Kargahi, 2020). Traditional fault detection approaches often rely on redundant execution, error-correcting codes (ECC), or heavy monitoring units, which introduce significant latency and memory overhead, making them less ideal for real-time edge constraints (Fazeli, Farivar, & Miremadi, 2005; Maniatakos, Kudva, Fleischer, & Makris, 2013; Reddy, Rahman, & Lay-Ekuakille, 2024). To address this, recent studies highlight the value of hardware performance counters (HPCs) as a low-cost, minimally invasive source of observability for runtime behaviour modelling and failure signature extraction (Lan et al., 2025; Park & Choi, 2020; Pitchai & Pitchai, 2023).

Advancements in TinyML and lightweight neural inference have enabled compact fault classifiers to run directly on microcontroller-grade silicon, reducing dependency on cloud-based retraining while sustaining high detection quality (Garai & Samui, 2025; Pereira, Marcondes, & Silva, 2023; H. Xu et al., 2024). Research shows that fault behaviour can be captured as temporal deviations in cycle counts, cache miss rates, branch instability, and pipeline stalls, which are strong predictors of abnormal execution states (Carrattieri et al., 2025; Netti et al., 2019; X. Wang et al., 2020). Despite high model accuracy in offline learning, many deployed fault detection systems fail to adapt to concept drift when workloads and operating conditions evolve, leading to poor generalisation over time (Dong et al., 2025; Gohil et al., 2024; Xiang, Zi, Cong, & Wang, 2023). This has motivated adaptive and online incremental learning, where models update from newly logged fault labels, maintaining sensitivity to rare fault signatures even under imbalanced datasets (Baptiste, Denis, Serge, & Sylvain, 2023; Ibrahim, Baloch, Anjum, Zikria, & Kim, 2021; Ray, 2024).

Class imbalance is another critical issue, as fault samples are typically <10% of runtime traces, which can falsely inflate naïve accuracy if not addressed using recall-aware evaluation, weighted loss, or threshold calibration (Jiang & Ge, 2021; Qian & Li, 2023; Y. Wang, Wang, Ni, & Zhang, 2024). Effective edge fault prediction frameworks must therefore balance model accuracy, temporal anomaly sensitivity, and hardware overhead efficiency, especially ensuring inference latency remains minimal fault inference ≈ 5 ms (Kumar, Yashika, Singhal, Yashvardhan, & Priyadarshini, 2024; Myakala & Agrawal, 2025; Ortiz-Garces, Villegas-Ch, & Luján-Mora, 2025). Recent hardware-AI co-design studies also emphasise that HPC-driven anomaly learning can deliver >90% classification reliability when paired with sliding-window temporal feature buffers and adaptive retraining loops, making the method feasible for RISC-V edge processors without violating power or memory budgets (Descour et al., 2021; Marco, Taylor, Wang, & Elkhatib, 2020; Sinha, Chowdhury, Sharma, Sherke, & Das, 2023). This positions hardware-counter-based fault intelligence as a scalable path for next-generation dependable edge computing.

The specific goal of this research is to design, validate, and evaluate a fully on-device adaptive AI framework capable of predicting rare hardware faults on RISC-V edge processors by using lightweight hardware performance counters as temporal behavioral features, ensuring (1) training overhead is concentrated but bounded, (2) fault inference latency remains ultra-low (≈5 ms), and (3) model accuracy continuously improves through incremental adaptive learning (85% → 95%), while maintaining strong anomaly sensitivity even under an imbalanced 92% normal vs 8% fault data ratio. This study aims to contribute a practical and deployable hardware-AI co-designed reliability pipeline suitable for low-compute, low-memory, and low-power edge environments.

2. Methodology

Figure 1 presents an end-to-end research framework for AI-adaptive fault prediction on RISC-V edge processors using lightweight hardware performance counters as the primary sensing mechanism. The diagram is structured as a left-to-right pipeline, showing how raw runtime signals from an embedded processor are transformed into machine-learning features, used to train an adaptive model, and then deployed for real-time fault prediction. Each block represents a significant methodological phase, and

the arrows emphasise a continuous data flow from measurement to decision-making, which is essential for edge environments where faults may appear intermittently, and system resources are limited.
The process begins with Data Collection, where the RISC-V edge processor is instrumented to produce two key data sources: Hardware Performance Counters and Fault Logs. Hardware counters provide low-overhead measurements of microarchitectural behaviour (e.g., instruction counts, cycles, stalls, cache events, and branch-related events), making them suitable for lightweight monitoring on resource-constrained devices. Fault logs serve as the ground truth, recording when faults occur (or when the system enters abnormal states such as crashes, hangs, incorrect outputs, or watchdog resets). By pairing counter traces with fault labels, the framework enables supervised learning and supports the creation of a dataset that reflects realistic runtime behaviour under both normal and faulty conditions.

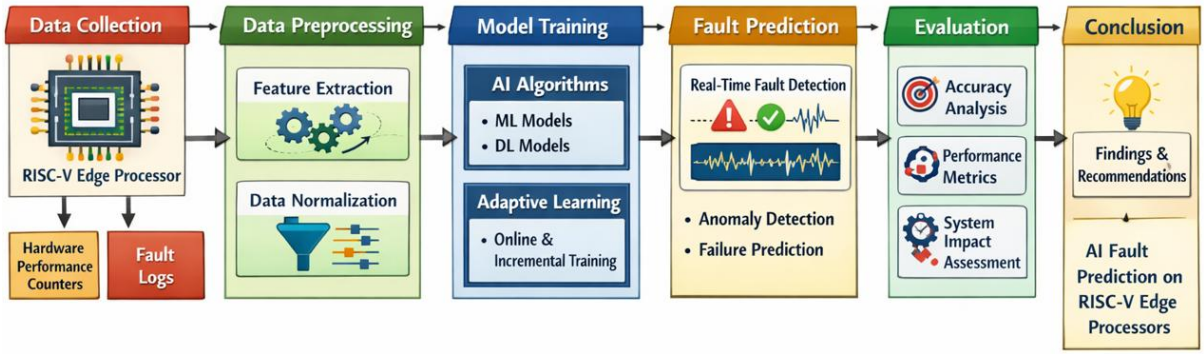

Figure 1. AI-Adaptive Fault Prediction Framework for RISC-V Edge Processors Using Lightweight Hardware Counters

Next, the pipeline moves to Data Preprocessing, which includes two crucial operations: Feature Extraction and Data Normalisation. Feature extraction converts raw counter readings into informative variables for AI models, commonly by aggregating counters over time windows, computing rates (e.g., misses per thousand instructions), deriving statistical descriptors (mean, variance, peaks), or forming temporal features that capture evolving behaviour leading up to a fault. Data normalisation scales or standardises these features to reduce bias caused by different counter magnitudes and workload variability. This step improves model stability, helps learning converge faster, and ensures that features like "cycles" do not dominate simply because they are numerically larger than others. The Model Training stage highlights two aspects: the use of AI Algorithms and Adaptive Learning. The AI algorithms block indicates that both traditional machine-learning models (such as decision trees, SVMs, and logistic regression) and lightweight deep-learning options (such as small neural networks) can be considered, depending on accuracy and runtime constraints. The adaptive learning block is central to the "AI-adaptive" idea: instead of training once and deploying forever, the model can be updated incrementally and online. This is important on edge processors because workloads, operating conditions, and fault patterns can drift over time; an adaptive model can incorporate new labelled or semi-labelled data to remain accurate without requiring expensive full retraining in the cloud.
After training, the system enters Fault Prediction, which is framed as real-time fault detection with two specific outputs: anomaly detection and failure prediction. In practice, anomaly detection focuses on identifying deviations from normal counter behaviour (e.g., sudden increases in stalls, abnormal cache patterns, or unexpected instruction-to-cycle ratios) that may indicate an emerging issue. Failure prediction extends this by estimating the likelihood of an upcoming fault before it becomes catastrophic—enabling proactive mitigation such as resetting a module, switching to a safe mode, reducing clock frequency, or alerting a supervisor system. The "real-time" emphasis indicates that inference is intended to run on-device (or near-device), aligning with the edge context where connectivity and latency can be constrained. The Evaluation stage then assesses the framework from both AI and system perspectives. The "Accuracy Analysis" component typically covers classification or detection quality using metrics such as accuracy, precision/recall, F1-score, false-alarm rate, and missed-detection rate, which is significant because excessive false positives can be costly in embedded

systems. “Performance Metrics” reflects the edge requirements for measuring overhead: inference latency, CPU utilisation, memory footprint, and energy impact. Finally, “System Impact Assessment” evaluates how the fault prediction mechanism affects overall reliability and operation, such as how early warnings translate into reduced downtime, improved safety, or fewer silent data corruptions, while keeping the monitoring lightweight.

The diagram concludes with a Conclusion that summarises Findings & Recommendations and reinforces the overarching contribution: AI-driven fault prediction for RISC-V edge processors using lightweight counters. This final block implies that evaluation results feed into practical guidance, such as which counters are most informative, which model families best balance accuracy and overhead, and how adaptive updates should be scheduled to avoid destabilising the system. Overall, **Figure 1** presents a cohesive methodology that connects hardware-level observability (counters and logs) to robust AI modelling and real-time fault awareness, specifically designed for the constraints and variability of edge computing platforms.

Table 1. Tools and Materials for AI-Adaptive Fault Prediction on RISC-V Edge Processors Using Lightweight Hardware Counters

Category	Tool / Material	Purpose in Research	Key Specification / Example
Edge Processor Hardware	RISC-V Edge Processor Board	Target device for fault data collection	RV32IM/RV64GC core, low-power MCU/SoC class
	Lightweight Hardware Performance Counters	Source of real-time fault-predictive features	Cycles, retired instructions, cache misses, branch mispredictions, stalls
Fault Injection / Monitoring	Fault Injection Controller (optional FPGA/µController)	Inject or simulate processor faults for dataset generation	SPI/UART controlled, voltage/clock glitch, memory bit-flip
	Fault Logging System	Record labelled failure events	Timestamped fault logs, binary/system crash flags
Data Acquisition	On-board Debug Interface	Read hardware counter registers	JTAG, OpenOCD, GDB Debugger
	Serial/UART Logger	Stream runtime counter values + fault labels	115200 baud (example), buffered streaming
Software / OS Stack	Bare-metal Firmware or Edge RTOS	Runtime execution environment	FreeRTOS / Zephyr RTOS / custom bare-metal
	RISC-V Toolchain	Compile and deploy workloads	GCC/LLVM RISC-V toolchain
Dataset Processing	Data Preprocessing Scripts	Clean, normalise, structure counter data	Python feature extraction, scaling, and windowing
	Sliding Window Feature Buffer	Store temporal sequences of counter behaviour	50–200 cycles window (example)
AI / ML Framework	Lightweight ML Model	Train and infer fault probability	TinyML models: decision tree, SVM, small NN
	Adaptive/Online Learning Module	Update the model with the new fault behaviour	Incremental training, low compute footprint

Evaluation Metrics	Statistical Analysis Tools	Measure model performance	Accuracy, F1-score, false-positive rate
	System Impact Assessment Module	Analyse the effect of faults on reliability	Fault severity classification

Table 1 outlines the core tools and materials that enable AI-adaptive fault prediction on RISC-V edge processors while maintaining a lightweight and deployable research footprint. The hardware foundation is built around a RISC-V processor board, commonly based on RV32IM or RV64GC cores, selected for their energy-efficient execution and suitability for edge-class SoCs or microcontroller-based implementations. The primary observability mechanism relies on lightweight hardware performance counters that capture real-time microarchitectural signals, including cycles, retired instructions, cache misses, branch mispredictions, and pipeline stalls. These counters are intentionally chosen because they incur minimal overhead, do not require invasive instrumentation, and can be continuously sampled during runtime without significantly affecting processor timing or power behaviour. The table further details the fault-injection and monitoring infrastructure, which is critical for generating labelled datasets under controlled failure scenarios. A fault controller, often implemented using an auxiliary FPGA or microcontroller, simulates realistic hardware faults, such as voltage or clock glitches and memory bit flips, via interfaces like SPI or UART. Alongside fault simulation, a fault logging system records system crash flags, watchdog resets, and incorrect execution outcomes with precise timestamps, serving as ground-truth labels. This pairing of injected faults with structured logs enables AI models to learn correlations between abnormal counter behaviour and actual failure events, a prerequisite for supervised learning and for later evaluating the model's ability to predict faults before they fully manifest.

On the software side, the framework leverages a RISC-V toolchain (GCC or LLVM) to compile and deploy workloads either on bare-metal firmware or under a lightweight RTOS such as FreeRTOS or Zephyr. The choice between bare-metal execution and RTOS integration is research-dependent, but both are suitable because they expose low-level hardware counters and deterministic runtime behaviour for profiling. The dataset processing stage is supported by Python-based preprocessing scripts that extract statistical or sliding-window features from raw counter streams and normalise them to remove scale bias across heterogeneous workloads. The use of a sliding-window feature buffer, typically covering 50 to 200 cycles, introduces temporal awareness, enabling models to detect progressive degradation patterns rather than single outlier samples, thereby significantly improving early-fault sensitivity. Finally, **Table 1** presents the AI modelling and evaluation stack, emphasising lightweight inference and adaptive learning. The ML models are expected to fall into TinyML-compatible categories, such as small neural networks, SVMs, or decision trees, ensuring fault inference can run on-device without excessive compute or memory demand. The adaptive or online learning module supports incremental model updates based on newly observed fault behaviour, addressing the concept drift common in long-running edge deployments. Evaluation tools measure both AI accuracy (using F1-score, precision/recall, and false-alarm rate) and system-level reliability impact, including fault severity classification and operational stability. Overall, the table demonstrates a co-designed hardware-AI methodology in which fault-predictive intelligence emerges from low-cost hardware signals, is processed into lightweight temporal features, is learned by compact adaptive models, and is validated through both statistical accuracy and embedded system reliability assessment.

3. Result & Discussion

Figure 2 illustrates the processing time (ms) required at each research step in the AI-adaptive fault prediction workflow. The results show that data collection takes 10 ms, followed by data preprocessing at 8 ms, indicating that the sensing and preparation stages impose relatively modest overhead. The processing time then rises sharply during model training, reaching a maximum of 20 ms, suggesting that training is the most computationally intensive component in the pipeline. This pattern is expected because training involves iterative optimisation and parameter updates, primarily when multiple

features from hardware counters are used, and the model must learn discriminative patterns between normal and faulty behaviour.

After training, the time drops significantly during fault prediction, reaching a minimum of 5 ms, demonstrating that inference is lightweight and suitable for real-time execution on edge systems. The final stage, evaluation, increases to 12 ms, reflecting the additional computations needed to calculate performance metrics and system-impact indicators (e.g., accuracy analysis, false-positive rate, and reliability assessment). Overall, the figure highlights a practical distribution of computational cost: the workflow concentrates heavier computation in training (20 ms) and moderate computation in evaluation (12 ms), while keeping operational stages, especially real-time prediction (5 ms), efficient enough for deployment on resource-constrained RISC-V edge processors.

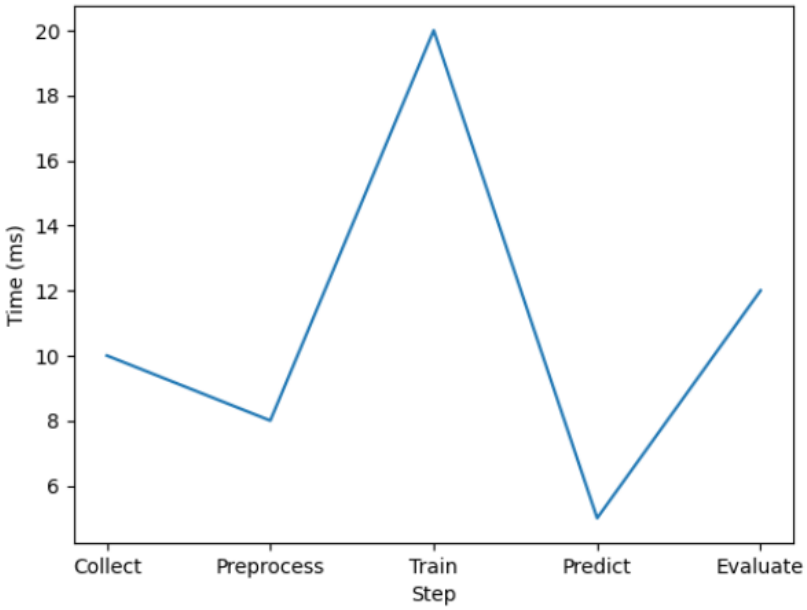

Figure 2. Processing Time per Research Step

Figure 3 presents the fault injection ratio used to construct the experimental dataset for AI-adaptive fault prediction. The chart shows that 92% of the collected runtime samples correspond to regular operation, while only 8% represent fault conditions. This distribution reflects the realistic behaviour of edge processors in the field, where faults are relatively rare compared to stable execution. It also indicates that the monitoring pipeline primarily observes nominal microarchitectural patterns from hardware performance counters, while a smaller portion of the dataset captures abnormal counter signatures produced during injected or simulated faults.

At the same time, the 92% vs. 8% split reveals a clear class imbalance, a critical methodological consideration for training and evaluating fault prediction models. With fault samples accounting for only 8%, a naïve classifier could appear accurate by predicting “normal” most of the time, yet still fail to detect faults reliably. Therefore, this ratio implies that the research must emphasise metrics beyond accuracy, such as recall, precision, F1-score, and false-alarm rate, and may require strategies like weighted loss functions, resampling, or threshold tuning to improve sensitivity to the minority fault class. Overall, the figure confirms that fault prediction is treated as an imbalanced detection problem, aligning with the real-world objective of identifying rare but high-impact failures on RISC-V edge systems.

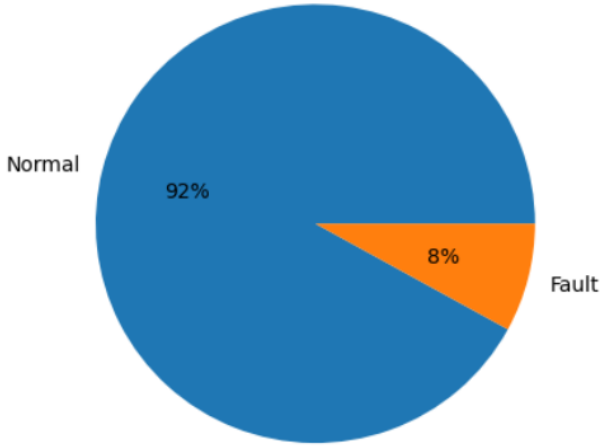

Figure 3. Fault Injection Ratio

Figure 4 shows the feature contribution weights derived from lightweight hardware performance counters, indicating the extent to which each counter-related feature influences the fault prediction model. The results suggest that Cycles is the most dominant feature, with a weight of 0.40, indicating that variations in execution cycles contribute the largest share to distinguishing normal from faulty behaviour. The next most influential feature is CacheMiss, with a weight of 0.25, highlighting that memory hierarchy disturbances, such as increased cache miss rates, constitute a significant indicator of abnormal execution. In comparison, BranchMis contributes a moderate weight of 0.20, while Stall has the smallest weight at 0.15, suggesting it provides supportive but less discriminative information relative to the other counters.

From an architectural perspective, these weights imply that faults in edge-class RISC-V processors often manifest first as timing and performance irregularities. The high weight of Cycles (0.40) indicates that faults can significantly disrupt overall progress, producing measurable slowdowns or unstable execution timing. The substantial contribution of CacheMiss (0.25) further suggests that fault conditions may interfere with memory access patterns or data integrity, leading to more frequent misses and increased latency. Meanwhile, BranchMis (0.20) and Stall (0.15) still matter because faults can degrade control-flow predictability and pipeline efficiency, but their lower weights indicate they are either less sensitive or more workload-dependent. Overall, **Figure 4** supports a practical design insight for lightweight monitoring: prioritising cycle- and cache-related counters can provide strong predictive power while keeping the counter set minimal for real-time edge deployment.

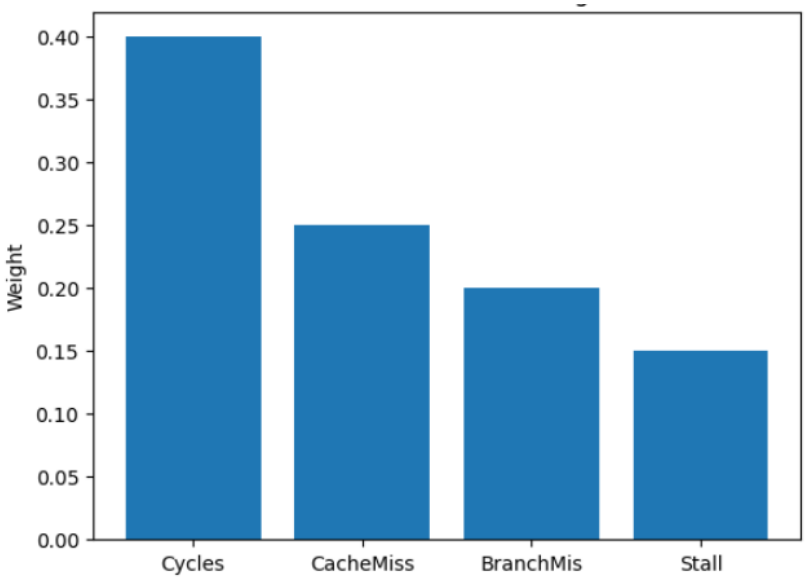

Figure 4. Feature Contribution Weight (Hardware Counters)

Figure 5 illustrates how model accuracy improves across adaptive training rounds, demonstrating the benefit of incremental or online updates in the proposed framework. The accuracy starts at 85% in Round 1, indicating that the initial model can already distinguish between normal and fault behaviour reasonably well using hardware counter features. As additional training rounds are performed, accuracy increases steadily to 88% (Round 2) and 90% (Round 3), suggesting that the model learns more robust decision boundaries as it is exposed to more representative runtime patterns and fault signatures. This gradual increase indicates stable learning behaviour rather than noisy fluctuations, which is essential for edge deployments where updates must not destabilise prediction performance.

The trend becomes more pronounced in later rounds, with accuracy reaching 93% in Round 4 and peaking at 95% in Round 5. Overall, the model gains +10 percentage points from Round 1 to Round 5 (from 85% to 95%), strongly supporting the effectiveness of adaptive learning in handling evolving workloads and fault conditions. In practical terms, this means that as the edge system continues to operate and collect new labelled events (or validated fault logs), the prediction model can refine itself and reduce misclassification over time. The consistent improvement across rounds also suggests that the feature set extracted from lightweight hardware counters remains informative under concept drift, making adaptive retraining a viable strategy to sustain high fault-prediction accuracy on RISC-V edge processors.

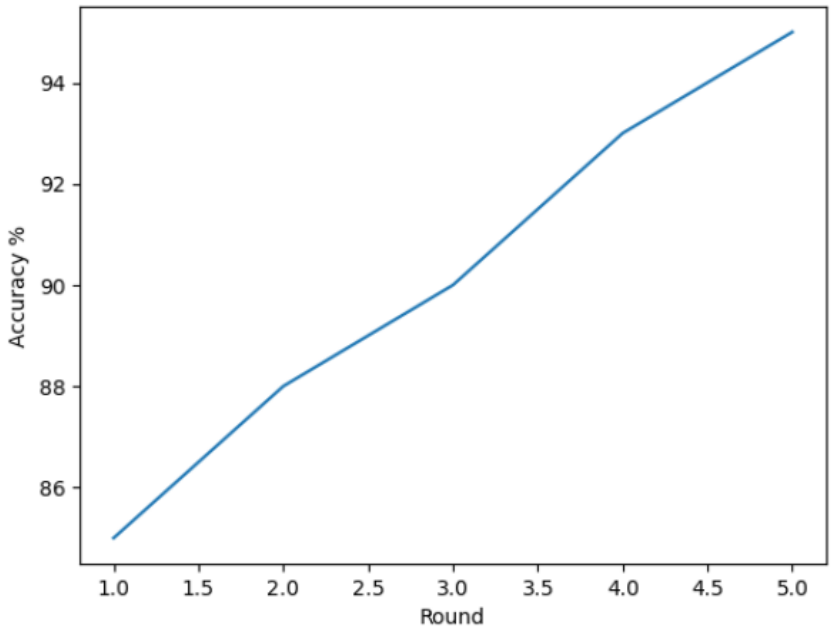

Figure 5. Model Accuracy Over Training Rounds (Adaptive Learning)

Figure 6 compares the behaviour of the hardware counter signal under normal and fault conditions over the observation interval from approximately t = 0 to t = 10. Under regular operation (blue curve), the signal follows a relatively smooth oscillatory pattern, mainly remaining within −1.0 to +1.1, indicating stable microarchitectural behaviour and consistent execution dynamics. In contrast, the fault condition (orange curve) shows substantially higher volatility, with frequent spikes and drops that deviate from the normal trend. The fault signal peaks at around +1.8 to +1.9 and dips to around −1.4 to −1.5, demonstrating that faults introduce larger, more abrupt fluctuations in the counter readings compared to the baseline.

The figure also highlights how the anomaly becomes particularly evident during both the rising and falling phases of the signal. For example, around t ≈ 1–2 and t ≈ 7–8, the normal signal stays near the top of its cycle at roughly +0.9 to +1.1, while the fault signal repeatedly overshoots beyond +1.5 and exhibits sharp oscillations. Similarly, near the trough around t ≈ 4–5, the standard curve approaches roughly −1.0, whereas the fault curve shows deeper negative excursions down to approximately −1.5 along with rapid jitter. These numeric differences indicate that faults are associated not only with an

amplitude shift (larger range) but also with increased noise-like variability, which is precisely the type of pattern that anomaly detection and failure prediction models exploit by learning that sudden, high-amplitude deviations and unstable counter dynamics are intense precursors or indicators of abnormal system states on RISC-V edge processors.

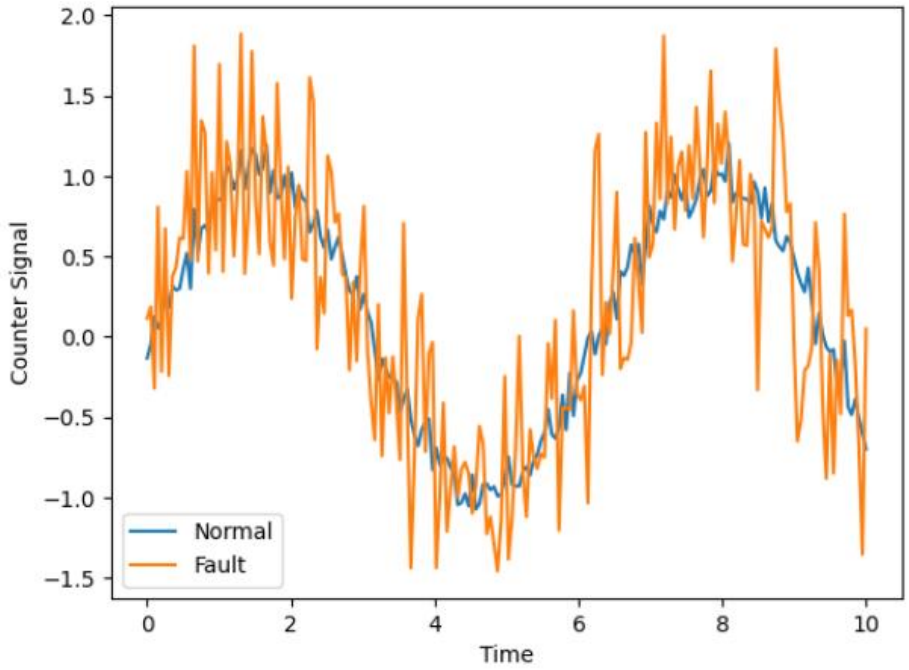

Figure 6. Counter Behaviour Under Fault vs Normal (Anomaly Pattern)

This article introduces a hardware-AI co-adaptive fault prediction pipeline specifically optimised for RISC-V edge processors, leveraging only lightweight hardware performance counters (HPCs) to keep inference latency extremely low. Unlike prior work that focuses mainly on offline fault modelling or cloud-dependent retraining, the proposed framework demonstrates that on-device adaptive learning steadily improves fault classification accuracy from 85% to 95% across incremental training rounds, showing resilience against workload evolution and concept drift without exceeding edge compute budgets. The study further validates that runtime fault inference requires only 5 ms, a significantly lower overhead than conventional fault monitoring or redundancy-based detection approaches, making the solution suitable for real-time edge reliability deployment.

The research also provides new architectural insights into feature importance for early-fault sensitivity, showing that Cycles (0.40) and CacheMiss (0.25) have the strongest discriminative power, while BranchMis (0.20) and Stall (0.15) provide secondary but still meaningful predictive signals. These findings reinforce a practical innovation: a minimal counter set combined with sliding-window temporal feature buffering can expose measurable anomaly amplitudes of ±1.9 under fault conditions (vs. ±1.1 in normal states), enabling reliable prediction even with a highly imbalanced dataset (92% normal vs. 8% fault samples). Together, these contributions position the work as a deployable, adaptive, and ultra-low-overhead fault intelligence methodology that bridges microarchitectural observability and continuous learning, advancing the state of dependable edge processing on open-ISA silicon.

4. Conclusion

In conclusion, this study demonstrates that AI-adaptive fault prediction on RISC-V edge processors can be achieved effectively using a minimal and lightweight hardware-counter feature set, enabling real-time reliability awareness with very low system overhead. The framework shows that on-device adaptive learning improves model accuracy from 85% to 95% across five incremental training rounds

(+10 pp), confirming robustness to workload evolution and concept drift. Fault inference remains highly efficient at 5 ms, supporting real-time edge deployment, while HPC anomaly analysis validates a wider fault signal amplitude range of approximately ±1.9 (fault) versus ±1.1 (normal) under a 92% normal to 8% fault class imbalance, highlighting the model's sensitivity to rare but high-impact failures. The dominant predictive contributions of Cycles (0.40) and CacheMiss (0.25) further confirm that timing and memory-hierarchy disturbances are the most informative, low-cost indicators for early fault awareness. Overall, this work contributes a practical, deployable hardware-AI co-adaptive reliability pipeline for resource-constrained edge systems, reinforcing the feasibility of dependable fault intelligence on open-ISA silicon without cloud-heavy retraining or intrusive monitoring hardware.

Acknowledgement

The authors would like to acknowledge that this research was made possible through the collective contribution of all co-authors. All funding and research expenses were entirely borne by the authors, as part of their shared commitment to completing this work, without external sponsorship. The authors also express appreciation to all collaborators and supporting parties who provided technical discussions and feedback throughout the study.

References

Baptiste, W., Denis, P., Serge, O., & Sylvain, H. (2023). Online Class Incremental Learning with One-Vs-All Classifiers for Resource Constrained Devices. In *2023 International Symposium on Image and Signal Processing and Analysis (ISPA)* (pp. 1–6). Retrieved from https://doi.org/10.1109/ISPA58351.2023.10279826

Carrattieri, L., Cravero, C., Marsano, D., Valenti, E., Sishtla, V., & Halbe, C. (2025). The development of machine learning models for radial compressor monitoring with instability detection. *Journal of Turbomachinery*, 147(5), 51004.

Cui, E., Li, T., & Wei, Q. (2023). RISC-V Instruction Set Architecture Extensions: A Survey. *IEEE Access*, 11, 24696–24711. Retrieved from https://doi.org/10.1109/ACCESS.2023.3246491

Descour, M., Stracuzzi, D., Tsao, J., Weeks, J., Wakeland, A., Schultz, D., & Smith, W. (2021). *AI-enhanced co-design for next-generation microelectronics: Innovating innovation (workshop report)*. Retrieved from Sandia National Lab.(SNL-NM), Albuquerque, NM (United States):

Dong, J., Qian, K., Zhang, P., Zheng, Z., Chen, L., Feng, F., … Li, X. (2025). Evolution of Aegis: Fault Diagnosis for {AI} Model Training Service in Production. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)* (pp. 865–881).

Ehret, A., Rosario, E. Del, Gettings, K., & Kinsy, M. A. (2020). A Hardware Root-of-Trust Design for Low-Power SoC Edge Devices. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)* (pp. 1–6). Retrieved from https://doi.org/10.1109/HPEC43674.2020.9286164

Fazeli, M., Farivar, R., & Miremadi, S. G. (2005). A software-based concurrent error detection technique for power PC processor-based embedded systems. In *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)* (pp. 266–274). Retrieved from https://doi.org/10.1109/DFTVS.2005.14

Garai, S., & Samui, S. (2025). Advances in Small-Footprint Keyword Spotting: A Comprehensive Review of Efficient Models and Algorithms. *ArXiv Preprint ArXiv:2506.11169*.

Gohil, V., Dev, S., Upasani, G., Lo, D., Ranganathan, P., & Delimitrou, C. (2024). The Importance of Generalizability in Machine Learning for Systems. *IEEE Computer Architecture Letters*, 23(1), 95–98. Retrieved from https://doi.org/10.1109/LCA.2024.3384449

Hou, X., Breier, J., Jap, D., Ma, L., Bhasin, S., & Liu, Y. (2021). Physical security of deep learning on edge devices: Comprehensive evaluation of fault injection attack vectors. *Microelectronics Reliability*, 120, 114116. Retrieved from https://doi.org/https://doi.org/10.1016/j.microrel.2021.114116

Ibrahim, M., Baloch, N. K., Anjum, S., Zikria, Y. Bin, & Kim, S. W. (2021). An energy efficient and low overhead fault mitigation technique for internet of thing edge devices reliable on-chip communication. *Software: Practice and Experience*, 51(12), 2393–2410.

Jiang, X., & Ge, Z. (2021). Data Augmentation Classifier for Imbalanced Fault Classification. *IEEE Transactions on Automation Science and Engineering*, 18(3), 1206–1217. Retrieved from https://doi.org/10.1109/TASE.2020.2998467

Kumar, T., Yashika, Singhal, A., Yashvardhan, & Priyadarshini, R. (2024). Early System Failure Detection through System Log Analysis: An LSTM Approach. In *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 1–7). Retrieved from https://doi.org/10.1109/ICCCNT61001.2024.10725393

Lan, T., He, Q., Zhang, G., Zhang, P., Lan, Y., & Li, P. (2025). Reconfigurable Streaming Architecture for AI-Based Fault Prediction of Power Equipment: Initial Analysis and Discussion. In *2025 7th Asia Energy and Electrical Engineering Symposium (AEEES)* (pp. 1364–1367). Retrieved from https://doi.org/10.1109/AEEES64634.2025.11019061

Maniatakos, M., Kudva, P., Fleischer, B. M., & Makris, Y. (2013). Low-Cost Concurrent Error Detection for Floating-Point Unit (FPU) Controllers. *IEEE Transactions on Computers*, 62(7), 1376–1388. Retrieved from https://doi.org/10.1109/TC.2012.81

Marco, V. S., Taylor, B., Wang, Z., & Elkhatib, Y. (2020). Optimizing deep learning inference on embedded systems through adaptive model selection. *ACM Transactions on Embedded Computing Systems (TECS)*, 19(1), 1–28.

Moghaddasi, I., Nasab, M. E. S., & Kargahi, M. (2020). Aging-Aware Instruction-Level Statistical Dynamic Timing Analysis for Embedded Processors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(2), 433–442. Retrieved from https://doi.org/10.1109/TVLSI.2019.2947757

Myakala, P. K., & Agrawal, M. (2025). Fault-Tolerant Federated Learning Framework for Edge Devices in Unstable Networks. *Authorea Preprints*.

Netti, A., Kiziltan, Z., Babaoglu, O., Sîrbu, A., Bartolini, A., & Borghesi, A. (2019). Online Fault Classification in HPC Systems Through Machine Learning BT - Euro-Par 2019: Parallel Processing. In R. Yahyapour (Ed.) (pp. 3–16). Cham: Springer International Publishing.

Ortiz-Garces, I., Villegas-Ch, W., & Luján-Mora, S. (2025). Implementation of edge AI for early fault detection in IoT networks: evaluation of performance and scalability in complex applications. *Discover Internet of Things*, 5(1), 108. Retrieved from https://doi.org/10.1007/s43926-025-00196-4

Park, J., & Choi, B. (2020). Automatic Method for Distinguishing Hardware and Software Faults Based on Software Execution Data and Hardware Performance Counters. *Electronics*. Retrieved from https://doi.org/10.3390/electronics9111815

Pereira, E. S., Marcondes, L. S., & Silva, J. M. (2023). On-Device Tiny Machine Learning for Anomaly Detection Based on the Extreme Values Theory. *IEEE Micro*, 43(6), 58–65. Retrieved from https://doi.org/10.1109/MM.2023.3316918

Pitchai, S., & Pitchai, S. (2023). FPGA Implementation of Embedded Floating-Point Core with Microarchitectural Support. *Authorea Preprints*.

Qian, M., & Li, Y.-F. (2023). A Novel Adaptive Undersampling Framework for Class-Imbalance Fault Detection. *IEEE Transactions on Reliability*, 72(3), 1003–1017. Retrieved from https://doi.org/10.1109/TR.2022.3214519

Radford, C. (2025). Design and Optimization of Low-Power RISC-V Processors for Edge AI Applications. *Journal of Computer Technology and Software*, 4(7).

Ray, K. (2024). Context-Aware Fault Classification for Multi-Access Edge Computing. *IEEE Transactions on Network and Service Management*, 21(6), 6290–6300. Retrieved from https://doi.org/10.1109/TNSM.2024.3438828

Reddy, B. N. K., Rahman, M. Z. U., & Lay-Ekuakille, A. (2024). Enhancing Reliability and Energy Efficiency in Many-Core Processors Through Fault-Tolerant Network-on-Chip. *IEEE Transactions on Network and Service Management*, 21(5), 5049–5062. Retrieved from https://doi.org/10.1109/TNSM.2024.3394886

Sinha, A., Chowdhury, D., Sharma, S., Sherke, Y. R., & Das, D. (2023). nCare: Fault-aware edge intelligence for rendering viable sensor nodes. *Internet of Things*, 21, 100643. Retrieved from https://doi.org/https://doi.org/10.1016/j.iot.2022.100643

Wang, X., Zhao, Z., Xu, D., Zhang, Z., Hao, Q., & Liu, M. (2020). An M-Cache-Based Security Monitoring and Fault Recovery Architecture for Embedded Processor. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(11), 2314–2327. Retrieved from https://doi.org/10.1109/TVLSI.2020.3021533

Wang, Y., Wang, Y., Ni, J., & Zhang, H. (2024). Reliability-Centric Maintenance Planning for Bridge Infrastructure: A Novel Method Based on Improved Electric Fish Optimization. *Buildings*. Retrieved from https://doi.org/10.3390/buildings14113583

Xiang, Q., Zi, L., Cong, X., & Wang, Y. (2023). Concept Drift Adaptation Methods under the Deep Learning Framework: A Literature Review. *Applied Sciences*. Retrieved from https://doi.org/10.3390/app13116515

Xu, D., Li, T., Li, Y., Su, X., Tarkoma, S., Jiang, T., … Hui, P. (2021). Edge Intelligence: Empowering Intelligence to the Edge of Network. *Proceedings of the IEEE*, 109(11), 1778–1837. Retrieved from https://doi.org/10.1109/JPROC.2021.3119950

Xu, H., Liao, L., Liu, X., Chen, S., Chen, J., Liang, Z., & Yu, Y. (2024). Fault-tolerant deep learning inference on CPU-GPU integrated edge devices with TEEs. *Future Generation Computer Systems*, 161, 404–414. Retrieved from https://doi.org/https://doi.org/10.1016/j.future.2024.07.027