

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

Elio Vinciguerra, Maurizio Palesi, Giuseppe Ascia

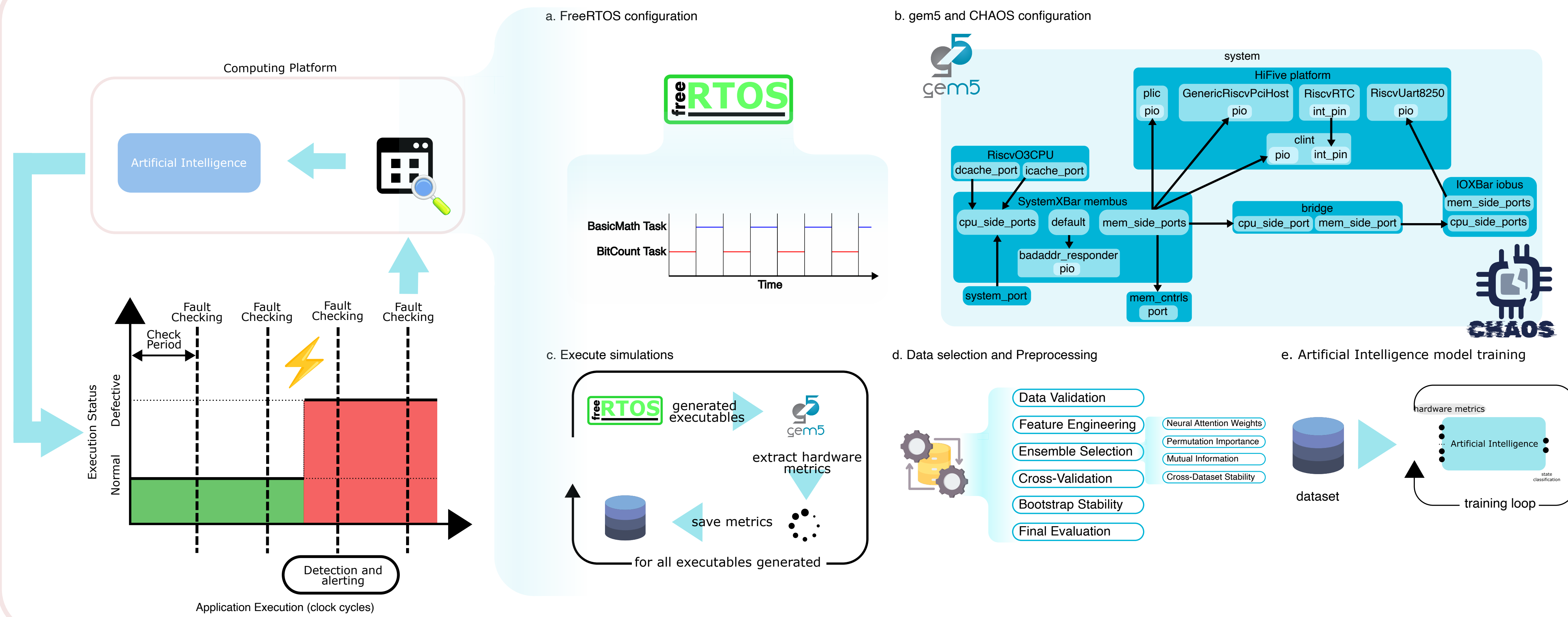
## Abstract

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.

## Background



## Proposed Methodology



## CHAOS

### Algorithm 1: Integration of CHAOS\_registers into gem5.

```

Data: cpu, probability, start, end, fault_type, mask,
      faulty_bits, target_class, PC_target
1 : enable ← random(0, 1);
2 if enable ≥ probability then
3   if start ≤ cpu.curCycle ≤ end then
4     if mask is NULL then
5       : mask ← generateMask(faulty_bits);
6       : injectFault(cpu, fault_type, mask,
                    target_class, PC_target);

```

### Algorithm 2: Integration of CHAOS\_cache into gem5.

```

Data: probability, inject_on_read, inject_on_write,
      start, end, fault_type, faulty_bits
1 : pkt ← isThereANewPkt();
2 if pkt then
3   : enable ← random(0, 1);
4   if enable ≥ probability and inject_on_write
5   then
6     if start ≤ cpu.curCycle ≤ end then
7       : mask ← generateMask(faulty_bits);
8       : injectFault(pkt, fault_type, mask);
9   : accessToCache(pkt);
10  if enable ≥ probability and inject_on_read then
11    if start ≤ cpu.curCycle ≤ end then
12      : mask ← generateMask(faulty_bits);
13      : injectFault(pkt, fault_type, mask);

```

### Algorithm 3: Integration of CHAOS\_memory into gem5.

```

Data: memory, probability, start, end, fault_type,
      mask, faulty_bits
1 : enable ← random(0, 1);
2 if enable ≥ probability then
3   if start ≤ cpu.curCycle ≤ end then
4     if mask is NULL then
5       : mask ← generateMask(faulty_bits);
6       : target ← getRandomAddress(memory);
7       : injectFault(memory, target, fault_type,
                    mask);

```

## Hardware metrics

HPM counters		gem5 statistics name
name	description	
mcycle	cycles number	system.cpu.numCycles
mtime	time	simSeconds
minstret	instructions retired	simInsts
mhpmcounter4	integer load instruction retired	system.cpu.executeStats0.numLoadInsts
mhpmcounter5	integer store instruction retired	system.cpu.executeStats0.numStoreInsts
mhpmcounter7	system instruction retired	system.cpu.commitStats0.committedInstType:prAccess
mhpmcounter8	Integer arithmetic instruction retired	system.cpu.commitStats0.committedInstType:intAlu
mhpmcounter9	Conditional branch retired	system.cpu.commitStats0.committedControl:lsCondControl
mhpmcounter10	JAL instruction retired	system.cpu.commitStats0.committedControl:lsCall
mhpmcounter11	JALR instruction retired	system.cpu.commitStats0.committedControl:lsUncondControl
mhpmcounter12	Integer multiplication instruction retired	system.cpu.commitStats0.committedInstType:intMult
mhpmcounter13	Integer division instruction retired	system.cpu.commitStats0.committedInstType:intDiv
mhpmcounter14	Floating point load & store instructions retired	system.cpu.commitStats0.committedInstType:FloatMemRead + system.cpu.commitStats0.committedInstType:FloatMemWrite
mhpmcounter15	Floating point other instructions retired	(system.cpu.commitStats0.committedInstType:FloatMemRead + system.cpu.commitStats0.committedInstType:FloatMemWrite)
mhpmcounter22	Branch/jump target misprediction	system.cpu.branchPred.BTBMispredicted
mhpmcounter27	Instruction cache miss	system.cpu.dcache.overallMisses:total
mhpmcounter28	Data cache miss or memory-mapped I/O access	system.cpu.dcache.overallMisses:total
mhpmcounter29	Data cache writeback	system.cpu.dcache.writebacks:total
mhpmcounter30	Instruction TLB miss	system.cpu.mmu.itlb.misses
mhpmcounter31	Data TLB miss	system.cpu.mmu.dtlb.misses

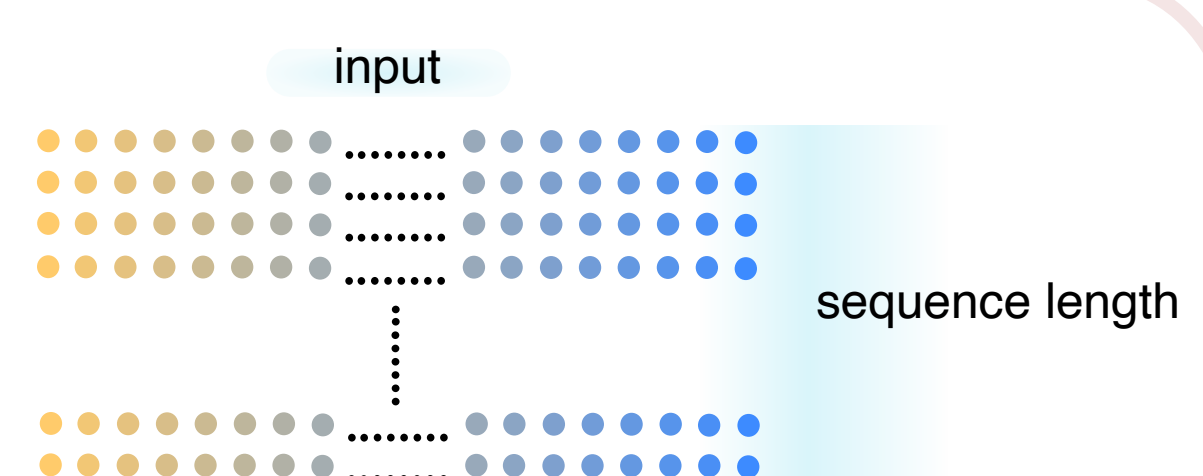
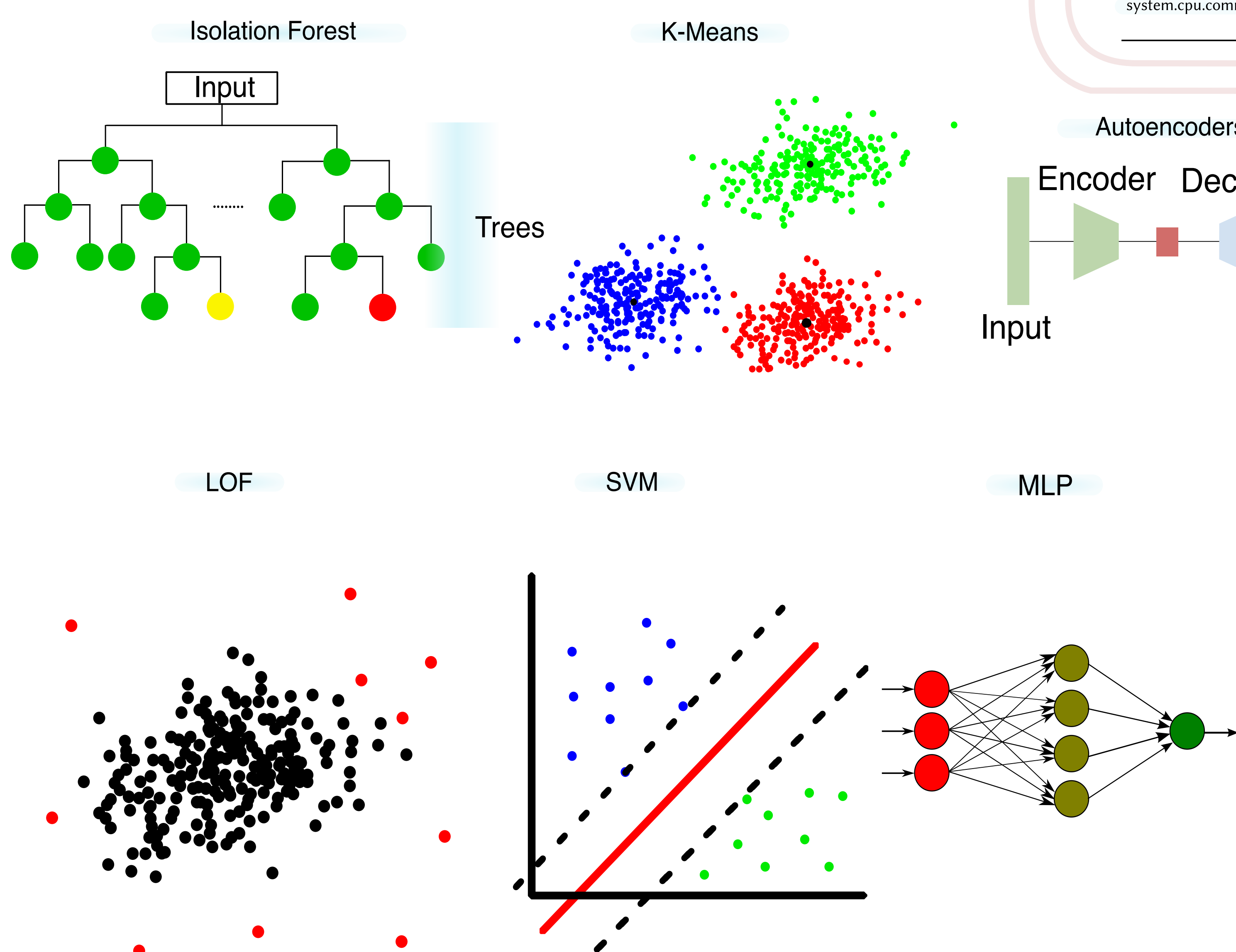
extended hardware metrics	description
system.cpu.dcache.replacements	number of data cache line replacements
system.cpu.dcache.demandMisses:cpu.data	demand load misses in the data cache
system.cpu.icache.tags.avgOcCs:cpu.inst	average tag array occupancy in instruction cache
icache_pressure_index	synthetic index estimating instruction cache pressure
system.l2cache.replacements	number of L2 cache line replacements
system.mem_cntrl.dram.readRowHits	DRAM read accesses hitting already open rows
system.mem_cntrl.dram.writeRowHits	DRAM write accesses hitting already open rows
system.cpu.idleCycles	number of CPU cycles where no instruction was executed
system.cpu.numSwp	number of context switches or swaps
system.mem_cntrl.dram.dramBytesRead	total bytes read from DRAM
system.l2cache.overallHits:cpu.data	L2 cache hits on data references
system.l2cache.overallMissRate:cpu.data	miss rate for data accesses in L2 cache
system.mem_cntrl.dram.pageHitRate	ratio of DRAM accesses hitting active (open) pages
system.l2cache.overallMisses:cpu.inst	L2 cache misses on instruction fetches
system.cpu.branchPred.mispredicted_0:total	total number of branch mispredictions
system.cpu.branchPred.committed_0:total	total number of committed branch instructions
combined_cache_pressure_index	aggregate index capturing overall cache pressure
system.cpu.branchPred.condPredictedTaken	number of conditional branches predicted as taken
system.cpu.executeStats0.instRate	instructions executed per cycle (instantaneous)
system.mem_cntrl.dram.numReads:total	total number of DRAM read transactions
system.l2cache.WritebackDirty:hits:writebacks	number of writeback hits on dirty L2 cache lines
system.cpu.commit.committedInstType_0:intMult	number of committed integer multiplication instructions
system.l2cache.overallMisses:total	total number of L2 cache misses (instructions + data)
system.cpu.commit.committedInstType_0:intDiv	number of committed integer division instructions
system.mem_cntrl.dram.readBursts	number of DRAM read bursts (burst-level transactions)
system.cpu.dcache.tags.avgOcCs:cpu.data	average tag array occupancy in data cache
system.mem_cntrl.dram.dramBytesWritten	total bytes written to DRAM
system.cpu.dcache.overallHits:total	total data cache hits (all accesses)
system.cpu.dcache.overallHits:cpu.data	data cache hits on demand data accesses
system.cpu.dcache.overallMisses:cpu.data	data cache misses on demand data accesses
system.mem_cntrl.dram.writeBursts	number of DRAM write bursts
system.cpu.dcache.demandMisses:total	total demand misses in data cache
system.cpu.commit.committedInstType_0:intAlu	number of committed integer ALU instructions
system.cpu.ipc	average instructions per cycle (IPC)

## Experimental setup

parameter	value
platform	HiFive
processor	Out-of-Order
clock frequency	1 GHz
L1 cache	16 KiB
L1D cache	64 KiB
L2 cache	256 KiB
main memory	DDR4 2048 MiB
check period	1,000,000 clock cycles
fault probability	from $10^{-1}$ to $10^{-8}$

FreeRTOS tasks: BasicMath, BitCount

## Algorithms employed



algorithm	complexity	d	n	v	L	h	i
KNN	$\mathcal{O}(n \cdot d)$	dataset size	features number	support vectors	sequence length	hidden state dimension	input dimension
LOF	$\mathcal{O}(n \cdot d^2)$						
SVM	$\mathcal{O}(n \cdot v)$						
GRU-Autoencoder	$\mathcal{O}(L \cdot (h^2 + h \cdot i))$						

### 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	99.67	94.99	94.99	89.99	94.99	94.99	100.0	95.0	98.91	94.99	∖∖	94.99	∖∖	100.0	100.0
CPU Registers [%]	81.37	86.11	83.79	82.99	84.64	84.68	85.38	39.29	85.34	86.28	84.11	∖∖	83.97	∖∖	35.43	69.5
L1D Cache [%]	94.48	98.28	97.79	94.4	98.54	95.49	99.13	29.26	98.05	96.26	98.26	∖∖	98.2	∖∖	25.54	90.27
L1I Cache [%]	94.69	98.44	97.64	94.18	98.4	95.46	99.05	29.56	98.07	96.25	98.12	∖∖	98.03	∖∖	25.78	98.64
L2 Cache [%]	91.6	98.58	93.46	92.41	95.09	96.12	96.42	67.72	94.05	98.05	94.03	∖∖	93.84	∖∖	65.2	95.21
Main Memory [%]	30.0	35.59	30.9	34.27	30.49	34.54	32.44	35.14	31.81	36.81	31.41	∖∖	31.33	∖∖	30.67	35.46
Overall F1 Score [0-1]	0.8	0.84	0.82	0.81	0.82	0.82	0.84	0	0.83	0.84	0.83	∖∖	0.83	∖∖	0	0.6

### 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	99.51	94.99	94.99	89.99	94.99	94.99	100.0	95.0	98.55	95.0	∖∖	94.99	∖∖	100.0	100.0
CPU Registers [%]	78.3	84.92	83.21	82.47	84.99	85.19	85.21	39.29	85.38	86.47	83.85	∖∖	83.91	∖∖	35.43	39.51
L1D Cache [%]	90.79	96.05	97.38	94.11	98.53	95.6	98.81	29.26	98.09	96.38	98.02	∖∖	98.11	∖∖	25.54	29.4
L1I Cache [%]	92.16	96.8	97.25	93.85	98.39	95.54	98.72	29.56	98.09	96.35	97.87	∖∖	97.97	∖∖	25.77	29.97
L2 Cache [%]	87.5	96.42	92.6	91.47	95.07	96.29	95.75	67.72	94.25	98.08	93.4	∖∖	93.7	∖∖	65.2	67.51
Main Memory [%]	28.78	34.91	57.15	57.79	71.72	83.47	77.65	35.14	59.2	72.65	59.31	∖∖	71.26	∖∖	30.67	35.26
Overall F1 Score [0-1]	0.77	0.82	0.87	0.85	0.9	0.92	0.92	0	0.88	0.91	0.88	∖∖	0.9	∖∖	0	0.01