

Session F: Reliable and Efficient Reconfigurable Computing

PAPER 134

ADAD

Energy-efficient hardware for real-time anomaly detection

Amirhossein Sadough, Mahyar Shamsavari, Mark Wijtvliet, Marcel van Gerven

Machine Learning & Neural Computing, Radboud University

ASMP

37th IEEE ASAP · Imperial College London · 2-4 September 2026



Real-Time Time-Series Anomaly Detection

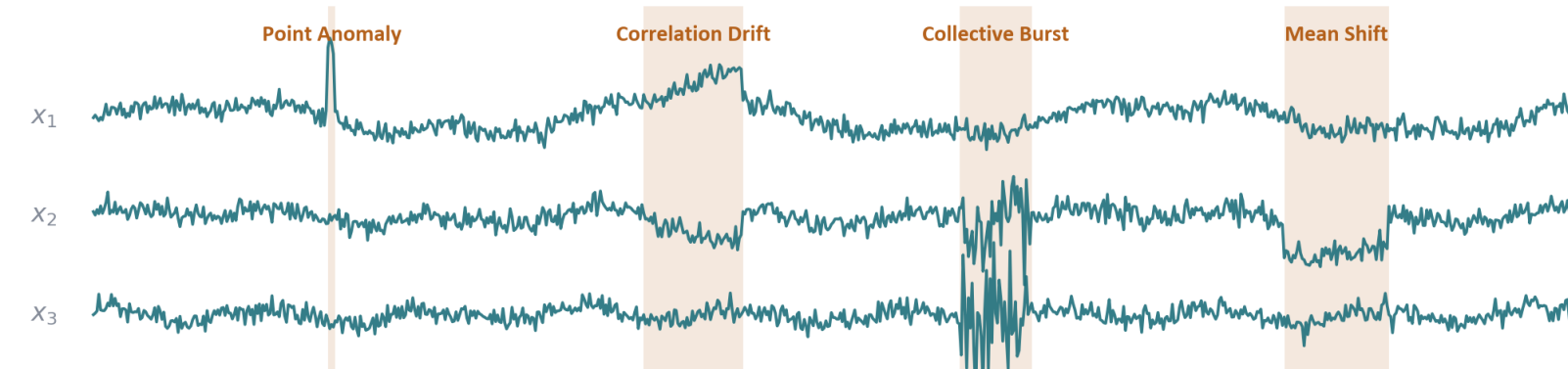
Anomaly?

- ❑ Samples that deviate from a normal behavior

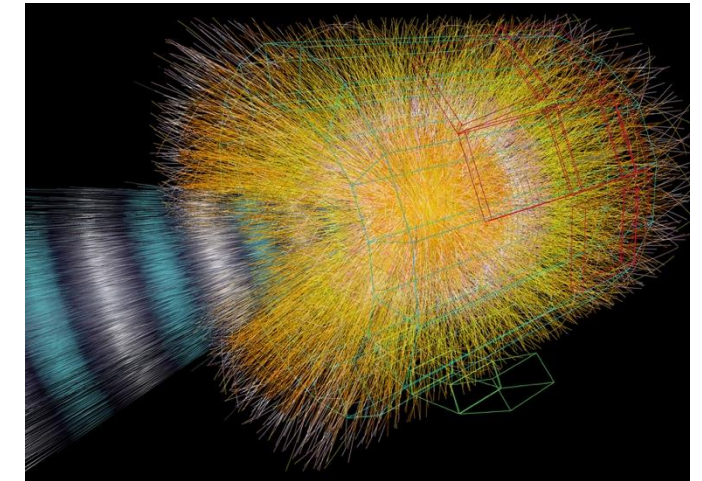
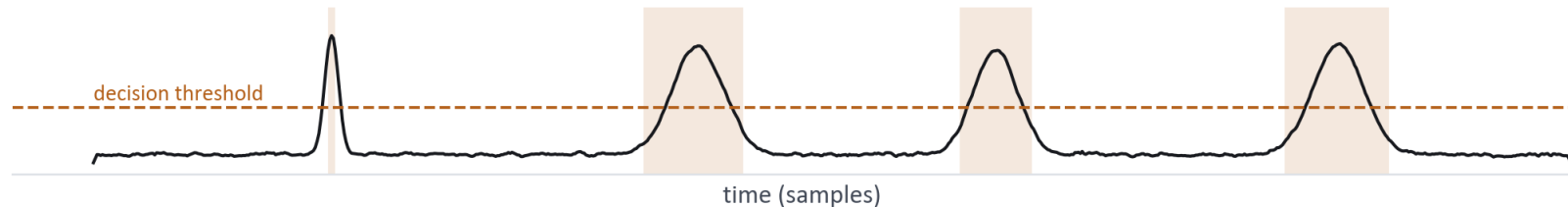
Anomaly Types?

- ❑ Point-anomaly, correlation drift, collective burst, mean shift

Multivariate input stream with diverse anomaly types



Streaming anomaly score



CERN's Large Hadron Collider (LHC)

Real-Time Time Series Anomaly Detection

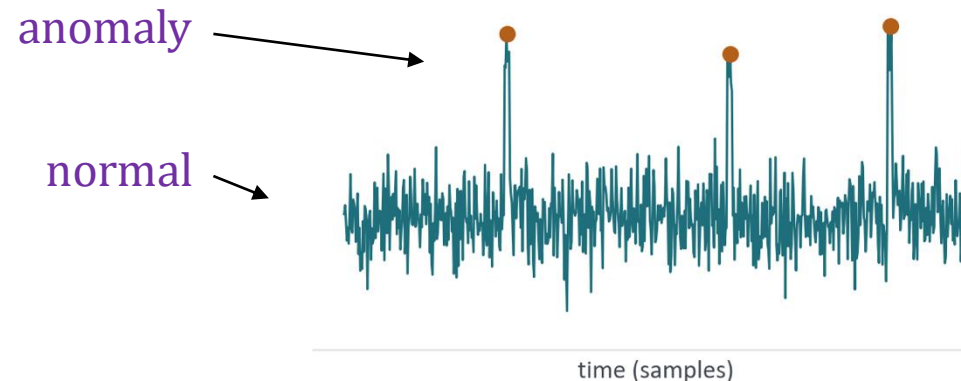
Dynamics?

- ❑ Stationary background (time-invariant)
 - Statistics (normal) are learned once
- ❑ Non-stationary background (time-variant)
 - Statistics (normal) are learned and updated continuously in real-time

A Real-Time Anomaly Detector

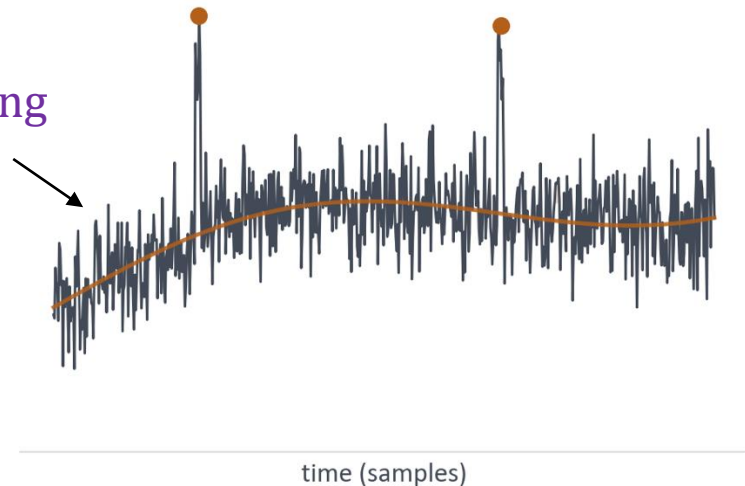
- Should support both dynamics:
- In practice normal distribution may drift over time

Stationary: Time-invariant background



Non-stationary: Time-variant background

normal, drifting over time



Real-Time Time Series Anomaly Detection

Anomaly Detection Methods

- ☐ Unsupervised (rely on no labeled samples)
- ☐ Semi-supervised (rely on normal samples)
- ☐ Supervised (rely on labeled normal and abnormal samples)

Prior Anomaly Detectors

- ☐ Sliding-window of past and present samples

A Real-Time Anomaly Detector

Should be unsupervised:

- Labels are not always available

Should work sample-wise:

- Less memory
- Less compute

Towards an Efficient and Generic Real-Time Anomaly Detector

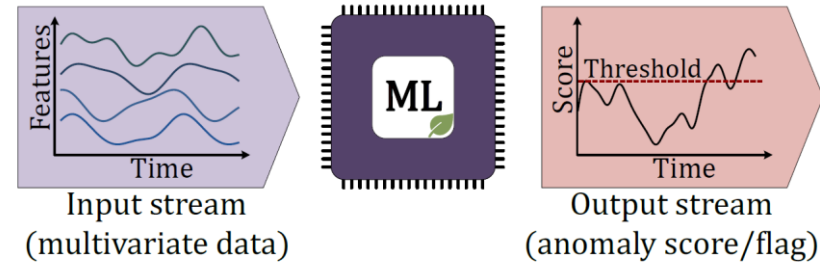


Fig. 1. Real-time AD on multivariate input streams.

- ✓ Dynamic Adaptation (both stationary and non-stationary time series)
- ✓ Unsupervised Operation (no need for labeled samples)
- ✓ Sample-Wise Processing (no past samples being used for the scoring of current sample)
- ✓ Dimensionality Scalability (i.e., support varying number of inputs/sensors)
- ✓ Energy and Computational Footprint (Meets edge hardware limits (power, area, latency) across edge and industrial IoT platforms)

Prior Work: Decorrelation-based Anomaly Detector (DAD)

Core Concept & Intuition

Feature Whitening: Normal multivariate signals share strong cross-correlations (e.g., synchronized sensors). DAD learns a linear matrix to suppress off-diagonal covariance.

Sample-Wise Streaming: Processes each sample online with zero historical buffers (runtime memory).

4-Step Operational Pipeline

1. Whitening Pass

$$\hat{x}_t \leftarrow x_t R_{t-1}^T$$

Projects incoming sample through current transform matrix.



3. Matrix Adaptation

$$R_t \leftarrow R_{t-1} - \eta U_t$$

Updates matrix per sample via gradient step rate.



2. Residual Error

$$U_t \leftarrow [(\hat{x}_t^T \hat{x}_t) - \text{diag}(\hat{x}_t^T \hat{x}_t)] R_{t-1} / (F - 1)$$

Isolates off-diagonal cross-correlation error matrix.



4. Gradient Scoring

$$s_t \leftarrow (1 - \gamma) \cdot s_{t-1} + \gamma \cdot ||R_t|| - ||R_{t-1}||$$

Anomalies cause sudden weight change spikes in .

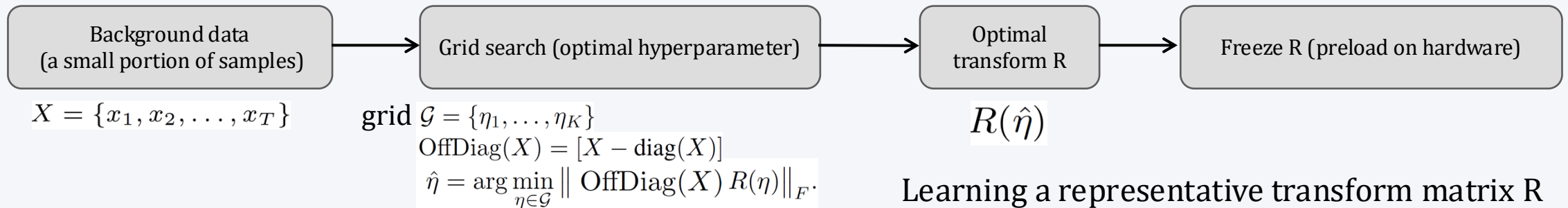


Extended-DAD (EDAD): Adding Freeze-after-Learn Regime

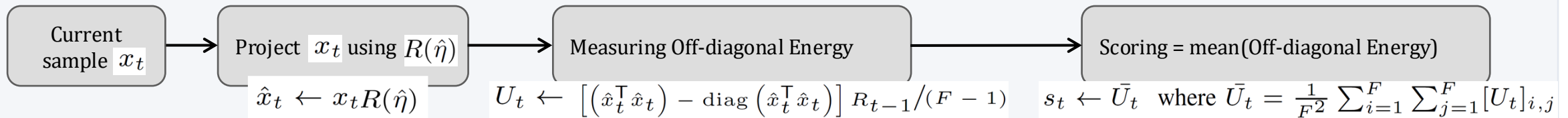
FAL extends DAD to *stationary* environments by leveraging available small portion of background data

Freeze-after-Learn (FAL)

Calibration phase



Deployment phase

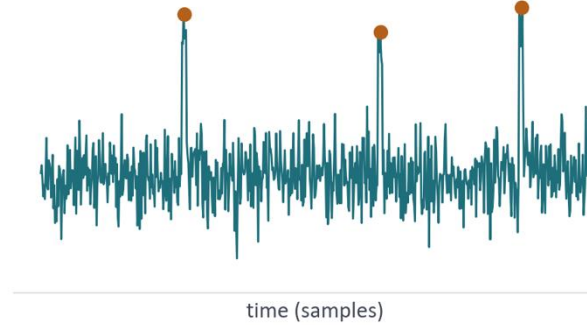


Static scoring using the learnt transform matrix R with no update

EDAD now supports both stationary and non-stationary environments

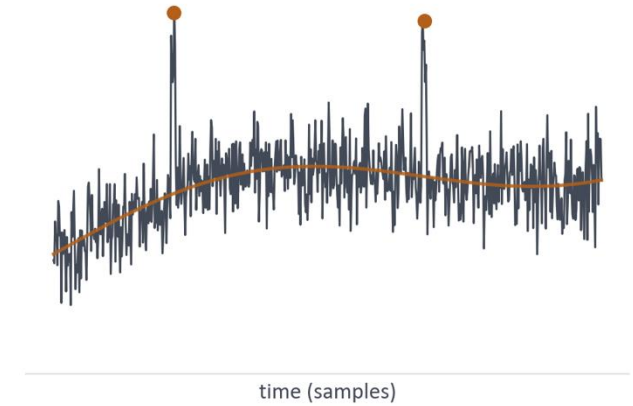
Freeze-after-Learn (FAL)

Stationary: Time-invariant background



Continual-Updating (CU)

Non-stationary: Time-variant background



So far, we built a unified algorithm supporting:

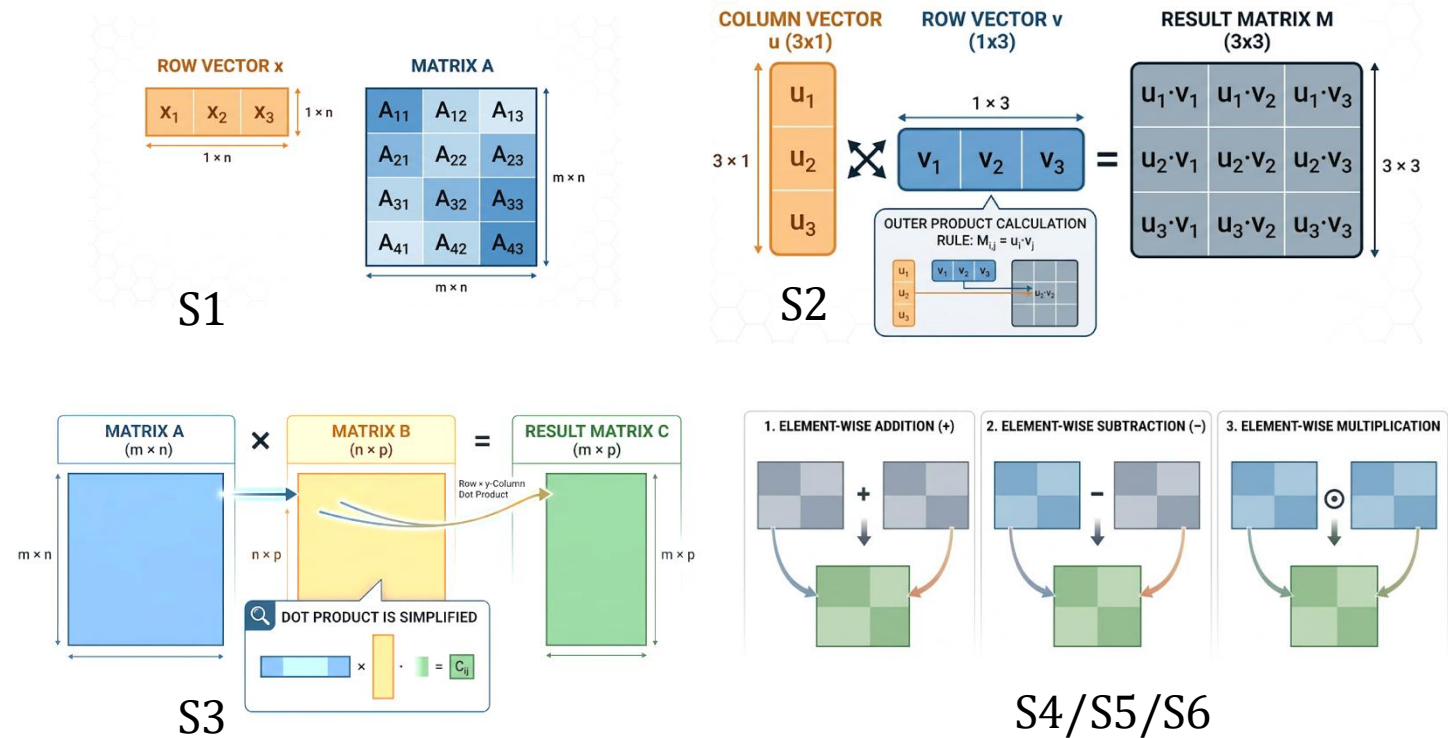
- ✓ Dynamic Adaptation (both stationary and non-stationary time series)
- ✓ Unsupervised Operation (no need for labeled samples)
- ✓ Sample-Wise Processing (no past samples being used for the scoring of current sample)

Hardware Accelerator for EDAD Algorithm

TABLE I
DETAILED COMPUTATIONAL STAGES OF ADAD.

Stage	#Operations	Expression
S1	$F^2 + F(F - 1)$	$\hat{x} = xR^T = [x_0 \ x_1 \ x_2 \ x_3] \begin{bmatrix} R_{00} & R_{10} & R_{20} & R_{30} \\ R_{01} & R_{11} & R_{21} & R_{31} \\ R_{02} & R_{12} & R_{22} & R_{32} \\ R_{03} & R_{13} & R_{23} & R_{33} \end{bmatrix}$
S2	F^2	$\hat{x}^T \hat{x}$
S3	$F^3 + F^2(F - 1)$	$R' = (\hat{x}^T \hat{x} - \text{diag}(\hat{x}^T \hat{x})) R$
S4	$2F^2$	$\bar{R} = R - nR'$
S5	$2F^2 - 1$	$\ \bar{R}\ = \sqrt{\bar{R}_{00}^2 + \bar{R}_{01}^2 + \dots + \bar{R}_{33}^2}$
S6	$2F^2 - 1$	$\bar{U} = (\rho R'_{00} + \rho R'_{01} + \dots + re R'_{33})$

Instance EDAD computational stages
for dimensionality = 4



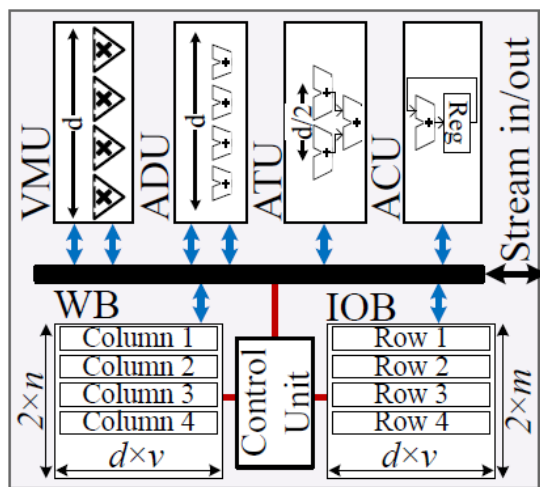
What operations the hardware should support?

- ☐ Vector-matrix and matrix-matrix multiplication
- ☐ Outer-product
- ☐ Element-wise multiplication/addition/subtraction

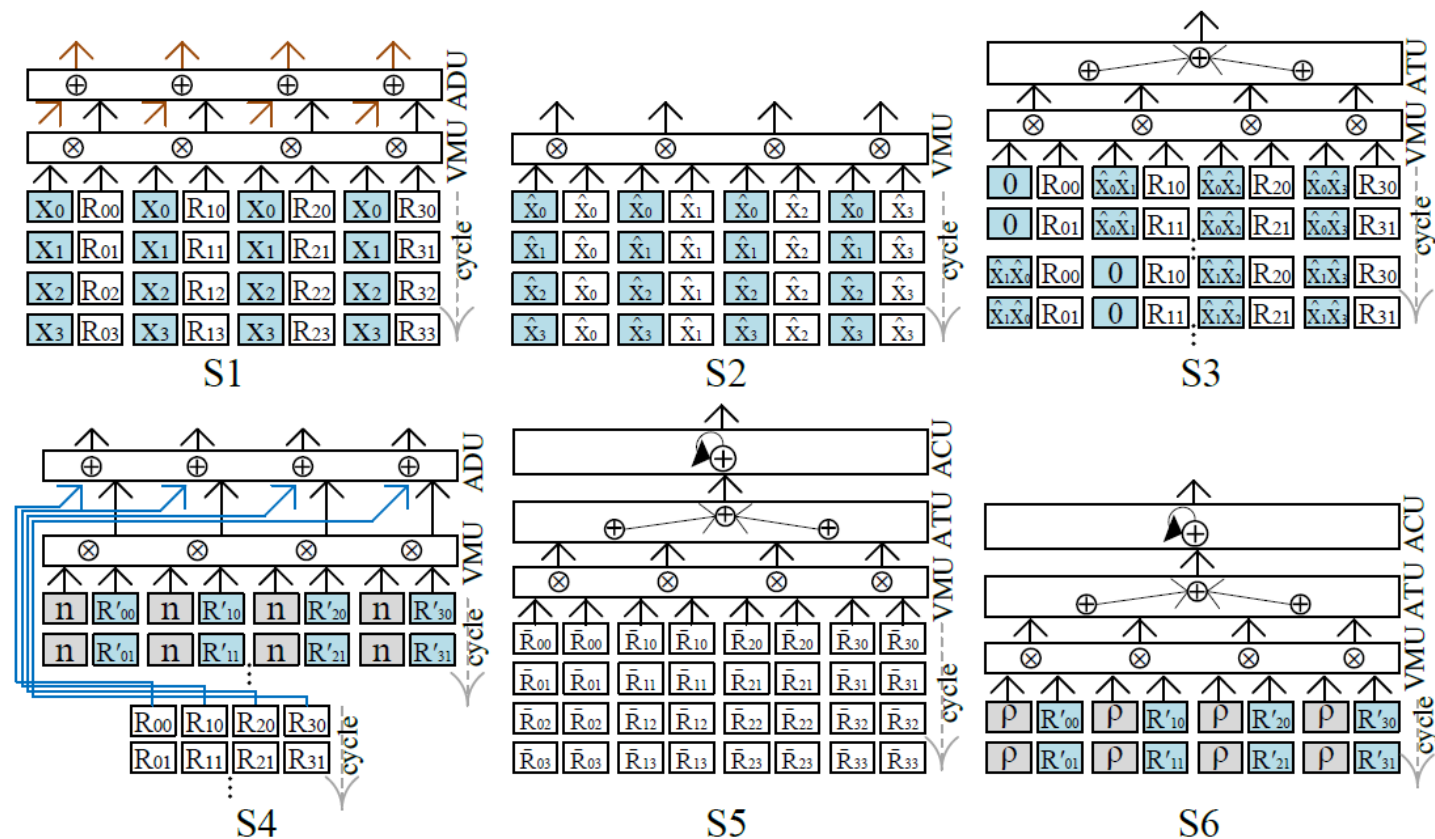
Hardware Accelerator: A Simple Yet Capable Architecture

ADAD is a generic matrix operation architecture:

- ❑ Vector multiplication unit (VMU)
- ❑ Vector adder/subtractor unit (ADU)
- ❑ Adder tree unit (ATU)
- ❑ Accumulator unit (ACU)
- ❑ IOB and WB buffers
- ❑ Reconfigurable interconnect
- ❑ Control unit (scheduling the operations)



ADAD architecture



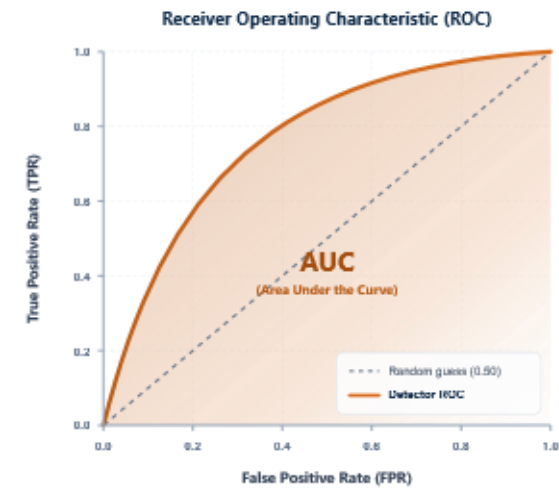
Anomaly Detection Performance

Accuracy validation on the Large Hadron Collider (LHC) physics dataset

Task: Real-time 40 MHz level-1 trigger anomaly detection across 56 feature dimensions.

TABLE II
AUC COMPARISONS ON THE LHC DATASET.

Dataset	[5]	[13]	EDAD (Ours)
$LQ \rightarrow b\tau$	0.92	0.93	0.95
$A \rightarrow 4\ell$	0.94	0.93	0.91
$h^0 \rightarrow \tau\tau$	0.81	0.85	0.86
$h^\pm \rightarrow \tau\nu$	0.94	0.94	0.96

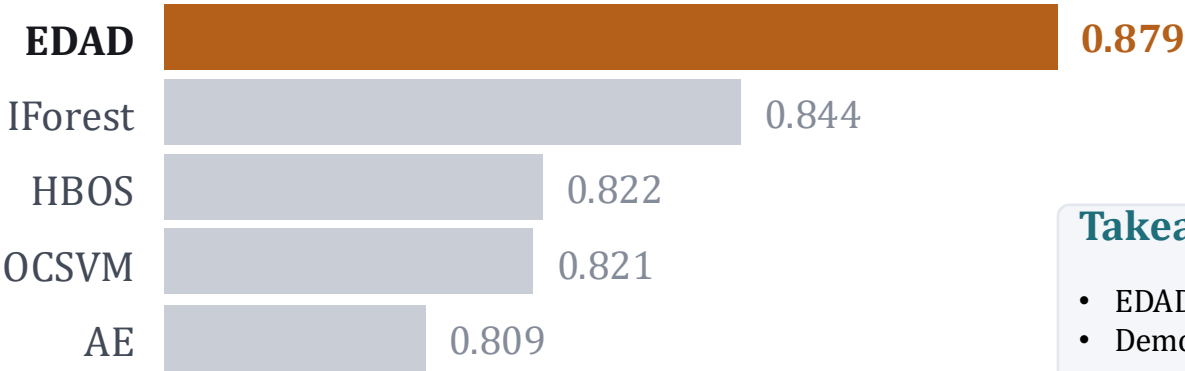


Takeaway

EDAD delivers higher accuracy across three scenarios against autoencoder-based [5] and decision-tree [13] anomaly detectors on such a stationary data

ODDS Anomaly Benchmark: highest average AUC of five detectors

Domains: healthcare, robotics, satellite images, and linguistics



Takeaway

- EDAD (Ours): 0.879 (Highest)
- Demonstrates strong domain generalization.

ADAD Hardware Performance:

Compared against a prior FPGA-based anomaly detector

	fSEAD [9]	ADAD
Platform	Xilinx ZCU111	
Freq. (MHz)	188	225
Precision	Fixed-32	Float-32
LUTs	117,481	15,554
FFs	80,346	2,604
BRAMs	381.5	49.5
DSPs	854	64
AUC	0.9311	0.9985[†]
Power (W)	6.456	1.570
Ex. Time (ms)	4.63	10.73 [†]
Energy (mJ/s)	29.89	16.84

[†]In continual-updating mode.

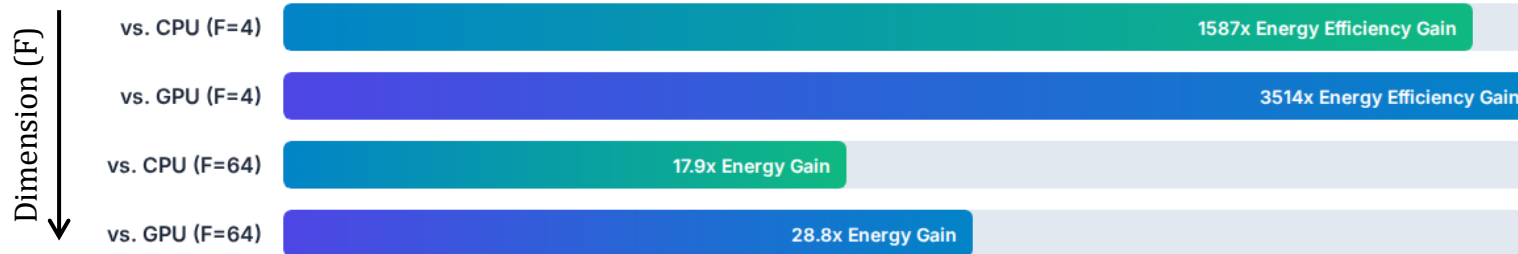
- Substantially fewer resources and higher energy efficiency than fSEAD
- Notably higher accuracy

ADAD Hardware Performance:

Benchmarked against CPU and GPU deployment

F	CPU		GPU		ADAD-FPGA		Efficiency Gain	
	Perf.	Pwr	Perf.	Pwr	Perf.	Pwr	vs CPU	vs GPU
4	25.78	13.39	6.85	7.87	3,578	1.17	↑ 1587	↑ 3514
8	25.49	13.49	6.88	8.13	1,638	1.25	↑ 693	↑ 1547
16	25.09	13.56	6.77	7.83	569.6	1.42	↑ 217	↑ 464
32	24.07	13.40	6.42	8.15	170.6	1.57	↑ 60.5	↑ 138
64	19.60	15.93	6.44	8.43	40.7	1.85	↑ 17.9	↑ 28.8
128	12.06	18.98	6.42	9.95	8.83	2.42	↑ 5.7	↑ 5.6
256	5.40	29.86	6.81	21.35	2.11	3.11	↑ 3.8	↑ 2.1
512	1.15	29.93	5.47	43.35	0.37	4.29	↑ 2.3	↓ 0.7

Perf. in KS/s; Pwr in W; Efficiency Gain = (ADAD-FPGA KS/J) / (platform KS/J).



- Higher throughput over CPU&GPU at up to F=128 (e.g., #sensors)
- Higher efficiency across dimensions up to 512 over CPU and up to 256 over GPU

Conclusion

What is ADAD

Algorithm & Methodology (EDAD)

- ✓ Unified Dual-Regime Algorithm: Covers stationary and non-stationary
- ✓ Label-Free Calibration
- ✓ Strictly Sample-Wise
- ✓ Cross-Domain Validation (higher accuracy)

Hardware Architecture & Scaling (ADAD)

- ✓ High-Dimensional Scalability
- ✓ General-Purpose Matrix Operation Platform
- ✓ Substantial Resource Savings and Energy Efficiency
 - Significantly fewer resources and higher efficiency than prior FPGA work
 - Higher throughput and efficiency over CPU and GPU baselines (strong candidate for edge deployment)

Limitations and Future Works

Current Hardware Bottlenecks

1D Parallelism Bottleneck

Vector-level parallelism becomes insufficient for very high feature dimensions.

FPGA Routing Congestion

Reconfigurable interconnect creates high routing pressure at large dimensions, degrading clock frequency from 290MHz down to 100MHz.

Stage S3 Workload Dominance

$\mathcal{O}(F^3)$ matrix-matrix multiplication dominates runtime at high dimensions.

Future Research Directions

Higher-Dimensional Parallelism

Stage S3 Algorithmic Alternatives

Formulate low-rank or sparse alternatives to bypass full $\mathcal{O}(F^3)$ Stage S3 matrix-matrix multiplication.

Congestion-Aware Interconnects

Design optimized interconnect topology to eliminate FPGA routing bottlenecks.

Questions?