Production-Grade Anomaly Detection in Serverless Function Monitoring

An Empirical Study

Velin Ekupov · Roya Nasri · Cristoffer Leite
Indika Kumara · Damian Tamburri · Willem-Jan van den Heuvel

Jheronimus Academy of Data Science · TU/e · Tilburg University

MOTIVATION

Serverless Anomaly Detection

Black-Box Problem

- Difficult to debug in production
- Limited visibility into application execution.
- Need for automated fault detection

Industry Approach

- Limited pipeline transparency
- Mainly statistical models
- Performance against modern DL remains unclear

Academia Approach

- Deep Learning-based anomaly detection
- Complex detection pipelines
- Limited evaluation on practical deployment

Research Goal: Can mature open-source DL models outperform production-style statistical baselines for serverless anomaly detection?

RESEARCH QUESTIONS

Three Research Questions

RQ1

Anomaly Detection Accuracy

How accurately do production-grade baselines like (S)ARIMA and mature OSS autoencoders detect injected faults in serverless workloads?

RQ2

Stability Across Fault Classes

How stable is each detector's accuracy across distinct fault classes — namely, resource stress (CPU/memory spikes) and latency anomalies?

RQ3

Deployability & Cost

What training time and configuration effort does each detector require under a fixed resource budget, and how do these factors affect practical deployability?

Methodology

Huawei Production Trace Dataset

Dataset Summary

Functions	200
Active Days	141 / 234
Total Rows	>11.5 M
Granularity	per-second & per-minute
Regions	Multiple Huawei Regions

Metrics

Metric
Requests
Function Delay
Platform Delay
CPU Usage
Memory Usage
pods

⚠ **Challenge:** Production traces contain **no labeled faults** → **Synthetic fault injection** is required.

METHODOLOGY

Fault Injection

Fault Classes

CPU Stress	Memory Stress	Function Delay	Platform Delay
Spike in CPU usage	Spike in memory usage	Spike in function execution time	Spike in scheduling/network overhead
p=0.02/min	p=0.005/min	p=0.02/min	p=0.02/min

TSAGen-based Fault Injection

1. Fit Log-Normal distribution
2. Extract extreme values using EVT (POT + GPD)
3. Generate Type I (Spike/dip in nearby values) & Type II (Burst/decline in a window of values) anomalies
4. Inject anomalies into production traces

METHODOLOGY

Data Preprocessing & Clustering

Pipeline: Preprocessing Steps

K-Means Clustering

Cluster	Requests	Function Delay (ms)	CPU (%)	% of Funcs
0	90,630	165	0.31	23.8%
1	3,801,226	2.6	0.21	1.6%
2	24,309	140	0.04	74%
3	430	10,809	0.03	1.6%

METHODOLOGY

Anomaly Detection Models & Evaluation

Models

(S)ARIMA	Dense AE	VAE	AER	LSTM-AE
Statistical baseline	*Reconstruction Autoencoder*	*Probabilistic Autoencoder*	*Autoencoder + BiLSTM*	*Seq2Seq LSTM autoencoder*

Post-processing

1. **Regression Errors**
2. **Reconstruction Errors**
3. **Anomaly Scores**

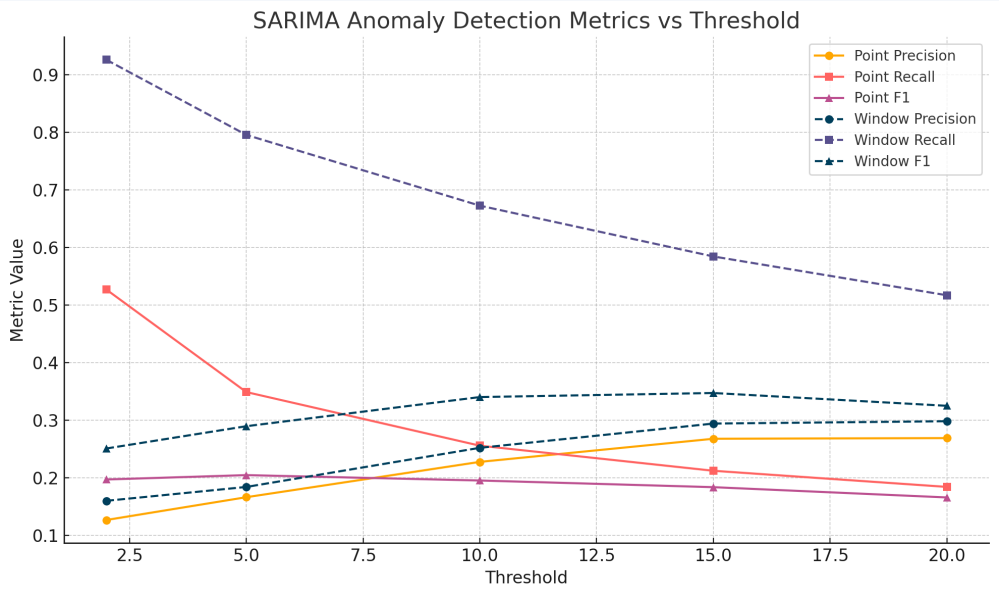
Evaluation Metrics

1. **Scoring:** Point-based, Window-based
2. **Metrics:** Precision, Recall, F1- Score

RESULTS

(S)ARIMA Baseline: Performance Analysis

Threshold Selection Trade-off



k=2: High recall, Low precision
k=20: Low recall, High Precision

Seasonality Findings

Non-seasonal
Window F1 = 0.40
Best overall

Hourly (60 min)
Window F1 = 0.30
Best seasonal

Quarterly (120 min)
Window F1 = 0.27
Worst seasonal

Limitations:

- Sensitive to threshold selection
- Better on window-level than point-level detection
- Lower performance on latency anomalies

RESULTS – RQ1

Anomaly Detection Accuracy

Model	Point-Level			Window-Level		
	F1	Precision	Recall	F1	Precision	Recall
SARIMA (baseline)	0.352 ±0.059	0.351 ±0.058	0.426 ±0.074	0.638 ±0.097	0.564 ±0.110	0.833 ±0.122
VAE	0.692 ±0.025	0.549 ±0.028	0.937 ±0.012	0.912 ±0.045	0.840 ±0.046	1.000
Dense AE	0.681 ±0.025	0.543 ±0.029	0.915 ±0.025	0.889 ±0.038	0.816 ±0.038	0.979 ±0.036
AER	0.648 ±0.036	0.555 ±0.033	0.782 ±0.059	0.826 ±0.068	0.706 ±0.068	1.000
LSTM-AE ★	0.686 ±0.038	0.547 ±0.035	0.920 ±0.049	**0.924 ±0.075**	0.876 ±0.075	0.981 ±0.033

All autoencoders outperformed SARIMA.

LSTM-AE achieved the best Window F1 (0.924).

All AEs achieve near-perfect recall (≈1.0).

RESULTS – RQ2 & Q3

Stability Across Fault Classes

- LSTM-AE remained stable across fault classes, maintaining a recall above 98% (e.g., window F1 = 0.90 for memory stress and 0.85 for platform delay).
- SARIMA performs well on memory metrics (window F1 = 0.89) but collapses on latency metrics (window F1 = <0.30).

Training Time & Configuration Effort

Model	Training Time	Configuration Effort
(S)ARIMA	Fastest	High
Autoencoders	Slower (LSTM-AE: Slowest)	Low

- Deep learning models require higher upfront computation, but they are multivariate, allowing them to capture better complexity.
- SARIMA trains faster but is fragile and requires retraining.

LIMITATIONS

Study Limitations

ARIMA Baseline

Commercial models are proprietary.
Our SARIMA model may under- or overestimate the performance of commercial monitoring solutions.

Synthetic Fault Injection

Faults are synthetically generated using EVT.
Only four FaaS fault types were simulated, so results may not fully represent real production failures.

Generalization

Evaluated on a single dataset.
Performance may vary under different workloads, cloud platforms, or unseen fault types.

Offline Evaluation

DL Models were evaluated offline.
Real-time streaming performance, tail-latency stability, and throughput were not assessed.

CONCLUSIONS

Conclusions & Future Work

RQ1

Detection Accuracy

All autoencoders outperformed SARIMA.

RQ2

Stability

Deep models remained stable across fault classes, but SARIMA struggled with latency faults

RQ3

Deployability & Cost

ARIMA trains faster, but DL models repay that investment with maintenance-free adaptive thresholds and substantially fewer false alarms.

Future Work: (1) Optimize ARIMA with SARIMAX + exogenous regressors, adaptive EVT-based thresholds.
(2) Validate on additional public/synthetic datasets for different traffic regimes and cold-start patterns.

Thank You!

Let's Stay in Touch

r.nasiri@tilburguniversity.edu

JADS Jheronimus Academy of Data Science