

000 001 002 003 004 005 RETHINKING HEAVY MODELS IN 006 MULTIVARIATE TIME SERIES ANOMALY DETECTION 007 008 009

010 **Anonymous authors**
011 Paper under double-blind review
012
013
014
015
016
017
018
019
020
021
022
023
024
025

ABSTRACT

026 Multivariate time series anomaly detection (MTS-AD) is widely used, but real-
027 world deployments often face tight computational budgets that limit the practical-
028 ability of deep learning. We revisit whether heavy deep models (high-FLOPs archi-
029 tectures) are necessary to achieve strong detection performance in such settings.
030 We conduct a systematic, compute-aware comparison of statistical, classical ma-
031 chine learning, and deep learning methods across diverse MTS-AD benchmarks,
032 measuring detection with AUROC (threshold-free, thus application-agnostic) and
033 cost with FLOPs (a hardware-agnostic proxy enabling fair cross-method com-
034 parison). We find that traditional approaches often match or surpass deep models,
035 which appear less frequently among the top performers, and that the effectiveness-
036 efficiency trade-off commonly favors non-deep alternatives under limited budgets.
037 These results indicate that deep learning is not uniformly superior for MTS-AD
038 and that heavy architectures can be counterproductive in resource-constrained de-
039 ployments. These findings offer practical guidance for practitioners designing
040 anomaly monitoring systems under compute constraints, highlighting cases where
041 lightweight models are sufficient and heavy deep models may be worth the cost.
042
043

1 INTRODUCTION

044 Time series anomaly detection is a fundamental task in machine learning with wide-ranging ap-
045 plications in domains such as industrial control systems, aerospace telemetry, and cyber security
046 (Kim et al., 2023; Hundman et al., 2018; Landauer et al., 2025). In practice, anomalies are rare
047 and difficult to label, which makes unsupervised anomaly detection methods trained on normal data
048 an essential approach. Over the past decade, deep learning methods have gained prominence for
049 anomaly detection, achieving impressive performance across a variety of benchmark datasets (Za-
050 manzadeh Darban et al., 2024).
051

052 However, real-world deployment environments often impose severe hardware and operational con-
053 straints. For example, safety-critical systems may need to operate without external connectivity due
054 to security restrictions, preventing the use of cloud-based solutions (Bhamare et al., 2020). Simi-
055 larly, embedded monitoring devices may lack GPUs or operate under strict thermal and power lim-
056 itations, making it impractical to deploy computationally intensive deep learning methods (Shuvo
057 et al., 2023; Singh & Gill, 2023). In such cases, the assumption that deep learning is the univer-
058 sally superior solution becomes questionable. While recent research has emphasized novel neural
059 architectures, comparatively little work has jointly examined both effectiveness and efficiency under
060 constrained computing conditions. Most studies focus on accuracy alone (Jia et al., 2025), with
061 only a few recent benchmarks considering accuracy together with runtime and memory usage (Qiu
062 et al., 2025). In addition, several works have highlighted inconsistencies in evaluation protocols and
063 metrics for time series anomaly detection in industry (Si et al., 2024).
064

065 This gap encourages us to consider two central questions: (i) What are the most effective options for
066 time series anomaly detection under limited computational resources, and are deep learning methods
067 always the best options? (ii) Does a trade-off between detection performance and computational cost
068 truly exist in practice? Our own experience in industrial applications, including monitoring of air
069 defense systems and equipment in manufacturing settings, has made clear the difficulty of balancing
070 computational demands with detection performance. We believe that many practitioners working in
071 real-world deployments encounter the same challenge.
072
073

054
055
056 Table 1: Summary of resource-constrained environments
057
058
059
060
061

Typical Domain	Representative Constraint	Common Approach	Key References
Limited memory			
IoT/Wireless Sensor Network nodes; streaming telemetry in manufacturing and equipment monitoring	devices cannot buffer long histories; models must adapt with small working sets under drift	online/streaming learning; feature selection; memory-efficient summaries/sketches	Bifet & Gavaldà (2007) Nancy et al. (2020) Jain et al. (2022) Chatterjee & Ahmed (2022) Mfondoum et al. (2024)
GPU unavailable			
ICS and OT; manufacturing cells; safety/certification-constrained environments	power/thermal, enclosure, and certification constraints preclude accelerators; inference must be CPU-only on-prem	quantization; pruning/compression; CPU-optimized runtime	Han et al. (2016) Jacob et al. (2018) Sipola et al. (2022) Das & Luo (2023) Singh & Gill (2023) Liu et al. (2024a) Fährmann et al. (2025)
Limited CPU capacity			
PLC/RTU-adjacent controllers; fanless industrial PCs; battery-powered sensing	very limited CPU cycles and RAM; strict cycle-time determinism	quantization; low-FLOPs model design; online/streaming updates; selective features	Liu et al. (2008) Goldstein & Dengel (2012) Singh & Gill (2023)
Restricted communication			
air-gapped ICS/OT; secure manufacturing cells; remote or intermittently connected sites	on-prem/offline operation and stringent latency disallow cloud round-trips; data egress may be restricted	local inference at the edge; federated/on-site adaptation; minimal upstream telemetry	Belenguer et al. (2022) Das & Luo (2023) Dehlaghi-Ghadim et al. (2023) Stouffer et al. (2023)

077
078 In this paper, we therefore address the questions by conducting a systematic comparative study of
079 unsupervised anomaly detection methods that range from traditional approaches to deep learning
080 methods. Unlike prior studies that have primarily emphasized accuracy, we introduce an evalua-
081 tion framework that considers both detection performance and computational cost (Mejri et al.,
082 2024). This perspective enables a fair comparison across different methodological approaches. Our
083 evaluation covers diverse real-world datasets drawn from industrial, server, and aerospace domains,
084 ensuring that our findings generalize across multiple application settings.

085 2 LITERATURE REVIEW

086 2.1 RESOURCE CONSTRAINED ENVIRONMENTS

087 The deployment of models in industrial system is not solely governed by algorithmic accuracy but
088 is equally constrained by system-level limitations. As summarized in Table 1, the literature consis-
089 tently highlights four recurring scenarios in resource-constrained environments, which encompass
090 limited memory, GPU unavailable, limited CPU capacity, and restricted communication. These sce-
091 narios illustrate the historical progression of research toward resource-aware solutions and motivate
092 the comparative analysis conducted in this study.

093 **Limited memory** Several studies have demonstrated that real-world industrial environments, in-
094 cluding IoT nodes, wireless sensor networks, and manufacturing telemetry systems, often operate
095 under severe limitations in storage and energy, making the buffering of long historical windows
096 infeasible (Jain et al., 2022; Mfondoum et al., 2024; Chatterjee & Ahmed, 2022). In time series
097 anomaly detection, such constraints require models to process data incrementally while maintaining
098 only a limited working set. Consequently, prior research has emphasized memory-efficient repre-
099 sentations, dimensionality reduction through feature selection (Nancy et al., 2020), and online or
100 adaptive windowing techniques, which dynamically adjust to evolving conditions (Bifet & Gavaldà,
101 2007).

102 **GPU unavailable** In operational technology (OT) and industrial control system (ICS) environ-
103 ments, the use of GPU accelerators is often infeasible due to strict power, thermal, and certifica-
104 tion constraints (Das & Luo, 2023; Liu et al., 2024a; Singh & Gill, 2023; Sipola et al., 2022). As a
105 result, inference is typically performed on CPUs, making model-level optimization essential. In

108 this context, two advances stand out as particularly effective. Integer-only quantization executes
 109 inference entirely with INT8 arithmetic, while deep compression combines pruning and quantization
 110 to reduce both computation and memory requirements. These techniques together enable efficient
 111 CPU-centric optimization (Fährmann et al., 2025; Jacob et al., 2018; Han et al., 2016).
 112
 113

114 **Limited CPU capacity** A related constraint emerges when devices rely on modest CPUs, such as
 115 PLC/RTU-adjacent controllers, fanless industrial PCs, or battery-powered IoT nodes. In these set-
 116 tings, the need for cycle time determinism combined with limited computational throughput requires
 117 fundamentally low complexity designs. Traditional detectors such as Isolation Forest or histogram-
 118 based method remain attractive due to their favorable time and memory complexity (Liu et al.,
 119 2008; Goldstein & Dengel, 2012). Recent surveys on Edge AI further emphasize that such CPU-
 120 constrained deployments require models explicitly tailored to minimize FLOPs while preserving
 121 detection capability (Singh & Gill, 2023; Sipola et al., 2022).
 122
 123

124 **Restricted communication** In many industrial and critical infrastructure domains, data transfer
 125 to the cloud is either infeasible or prohibited due to latency requirements and strict security policies.
 126 Authoritative guideline from National Institute of Standard and Technology explicitly recommend
 127 isolation of OT networks, reinforcing the necessity of on-device or on-premise models (Stouffer
 128 et al., 2023). Research has therefore explored federated learning approaches to enable collaborative
 129 learning without raw data sharing, particularly in distributed industrial environments, as well as
 130 lightweight edge frameworks (Belenguer et al., 2022; Dehlaghi-Ghadim et al., 2023).
 131

132 2.2 TIME SERIES ANOMALY DETECTION ALGORITHMS

133
 134 Early research on time series anomaly detection was largely grounded in basic statistical models,
 135 which assumed stationary distributions and relatively simple dependency structures. Among the
 136 most influential were multivariate monitoring methods such as Hotelling’s T^2 statistic that leveraged
 137 distributional thresholds to detect deviations in industrial manufacturing processes (H.Hotelling,
 138 1947; Ye & Chen, 2001; Zheng et al., 2016). However, their reliance on linearity and stationarity
 139 assumptions rendered them less effective when confronted with the high-dimensional, noisy, and
 140 non-stationary signals that characterize modern industrial systems.
 141

142 The subsequent wave of research introduced non-parametric machine learning approaches that re-
 143 laxed restrictive distributional assumptions. Distance and density-based detectors identified anom-
 144 alies as local deviations within the data manifold (Angiulli & Pizzuti, 2002; Breunig et al., 2000),
 145 while clustering-based techniques grouped time series patterns to distinguish normal from abnormal
 146 behavior (He et al., 2003). Ensemble-based strategies, such as Isolation Forest, improved robustness
 147 and scalability through randomized partitioning and aggregation (Liu et al., 2008).
 148

149 Driven by advances in representation learning, the field has recently shifted toward deep learning
 150 approaches that explicitly model sequential dependencies and nonlinear structures. Early works
 151 employed recurrent neural networks and detected anomalies by reconstructing temporal sequences
 152 (Malhotra et al., 2016; Park et al., 2018). This paradigm was later extended by probabilistic gen-
 153 erative models, adversarially trained architectures, and attention-based models (Su et al., 2019; Li
 154 et al., 2019; Geiger et al., 2020; Zhou et al., 2019; Akcay et al., 2019; Tuli et al., 2022; Xu et al.,
 155 2022; Wu et al., 2023). Collectively, these approaches represent a clear trajectory toward increas-
 156 ingly complex and expressive models, often achieving state-of-the-art accuracy across widely used
 157 benchmarks. Nevertheless, their heavy reliance on GPU accelerators and large memory footprints
 158 has raised practical concerns regarding deployment in resource-constrained industrial environments.
 159

160 To contextualize the performance of these approaches, several benchmark studies have systemat-
 161 ically compared classical and deep learning methods (Han et al., 2022; Paparrizos et al., 2022; Si
 162 et al., 2024). However, most prior studies emphasize accuracy while giving limited attention to com-
 163 putational cost. In this work, we provide a more balanced evaluation by jointly examining detection
 164 performance and computational cost across both traditional and deep learning models. Further dis-
 165 cussion of the scope and limitations for existing benchmarks is presented in the Appendix A.
 166

162

3 EXPERIMENTAL DESIGN

164

3.1 PROBLEM DEFINITION

166

3.1.1 UNSUPERVISED ANOMALY DETECTION IN TIME SERIES

168 Unsupervised anomaly detection in multivariate time series is the task of identifying abnormal behaviors without access to anomaly labels during training (Pang et al., 2021). This setting is common
 169 in practice because anomalies are rare, labeling is costly (Blázquez-García et al., 2021).

170 A multivariate time series is defined as $x \in \mathbb{R}^{N \times D}$ and can be expressed as

$$172 \quad X = \{x_t \in \mathbb{R}^D \mid t = 1, \dots, N\} \quad (1)$$

174 where N denotes the sequence length and x_t is the D -dimensional observation at time t . Based on
 175 the learned representation of normal behaviors, the model provides a decision function that assigns
 176 each observation an anomaly score

$$177 \quad s_t = F(x_t), \quad \{s_t\}_{t=1}^N \in \mathbb{R}^N \quad (2)$$

179 where observations with higher scores are regarded as anomalies, typically determined by calibrating
 180 a threshold from training scores (Su et al., 2019; Audibert et al., 2020; Xu et al., 2022).

181 The unsupervised formulation is particularly important in real-world applications, since annotated
 182 anomalies are typically unavailable, occur infrequently, or vary significantly across domains (Salehi
 183 & Rashidi, 2018). By modeling normality directly from unlabeled data, unsupervised approaches
 184 provide a practical and general framework for anomaly detection in time series.

185

3.1.2 RESEARCH QUESTIONS

187 Building on the above definition, we aim to investigate how unsupervised anomaly detection in mul-
 188 tivariate time series can be evaluated not only in terms of effectiveness but also efficiency under
 189 realistic deployment settings. While prior work has primarily emphasized improving detection acc-
 190 curacy, comparatively less attention has been paid to the computational and operational feasibility
 191 of different methods. To address this gap, we define the following research questions:

192 **(i) What are the most effective options for time series anomaly detection under limited com-
 193 putational resources, and are deep learning methods always the best options?** This question
 194 is motivated by the observation that many deployment environments face practical constraints, in-
 195 cluding restricted computational capacity, memory, or energy availability. Although deep learn-
 196 ing methods have shown strong benchmark performance, their dependence on substantial resources
 197 raises doubts about their universal applicability. It is therefore important to examine whether tradi-
 198 tional statistical or machine learning methods may provide more practical alternatives under such
 199 constrained conditions.

201 **(ii) Does a trade-off between detection performance and computational cost truly exist in prac-
 202 tice?** Deep learning models are generally associated with higher computational cost due to their
 203 larger architectures and resource requirements. The literature also shows that traditional statistical
 204 and machine learning methods can remain competitive in certain scenarios. This raises the question
 205 of whether such cost is justified by consistently superior detection performance. This question there-
 206 fore seeks to clarify whether higher computational cost truly translates into superior performance,
 207 or whether certain approaches can offer a more balanced relationship that challenges the prevailing
 208 view.

209

3.2 EVALUATION PROTOCOL

211

3.2.1 CHOICE OF METHODS AND DATASETS

213 **Algorithms** We survey the historical development of unsupervised time series anomaly detec-
 214 tion and curate a representative set of models for evaluation. The selection of models follow three
 215 principles. First, to ensure chronological coverage, we consider landmark contributions from the
 early multivariate statistical monitoring methods through contemporary deep learning methods, so

216 that each major period is represented. Second, to reflect differences in operating mechanisms, we
 217 partition the candidate methods into three families, namely statistical, one-class classification, and
 218 reconstruction-based, and we select representative exemplars from each family. Third, we focus on
 219 models that are well-established in the field and have been widely adopted in prior studies. Detailed
 220 descriptions of the included methods are provided in the Appendix B.1.

221
 222 **Datasets** To support reproducible and deployment relevant evaluation, we prioritize datasets that
 223 capture real operational complexity and are widely used in the literature. We also focus on datasets
 224 from machinery and electronic system contexts where anomaly detection techniques are routinely
 225 deployed. Accordingly, we include SMD (Su et al., 2019) and PSM (Abdulaal et al., 2021) from
 226 server environments, SMAP and MSL from NASA engineering telemetry (Hundman et al., 2018),
 227 and SWaT (Mathur & Tippenhauer, 2016) and WADI (Ahmed et al., 2017) from industrial water
 228 treatment and distribution testbeds. Each dataset is multivariate and displays cross-channel depen-
 229 dencies and non-stationary dynamics, with rare anomalies as in real-world environments. Detailed
 230 descriptions of the included datasets are provided in the Appendix B.2.

231 3.2.2 METRICS
 232

233 **Detection Performance** We compare detection performance across models using a threshold-
 234 agnostic metric, the Area Under the Receiver Operating Characteristic Curve (AUROC), to ensure
 235 fairness. Although certain methods are often paired with post hoc thresholding procedures such as
 236 Peaks Over Threshold, we do not apply such schemes to evaluate the intrinsic ranking quality of
 237 each model. To ensure a uniform basis of comparison, each implementation outputs a real-valued
 238 anomaly score at every time step, aligned with the original sequence length. AUROC is therefore
 239 computed at the same temporal resolution for all methods.

240 **Computational Cost** To enable a fair comparison of computational demands across both tradi-
 241 tional algorithms and deep learning models, we adopt floating-point operations (FLOPs) as a uni-
 242 fying metric. FLOPs are model-agnostic and can be meaningfully related to hardware capabilities,
 243 making them particularly suitable for analyzing hardware-constrained scenarios. Unlike elapsed real
 244 time, FLOPs isolate algorithmic complexity from hardware variability. However, estimating FLOPs
 245 is nontrivial because the operations that dominate computational cost differ substantially across
 246 models and are highly sensitive to hyperparameter settings (e.g. tree depth in ensemble methods,
 247 number of neighbors in k -NN, or hidden dimension in neural networks). Consequently, prior work
 248 has rarely reported FLOPs for traditional machine learning methods, focusing only on deep learning
 249 models. We therefore derive closed-form or tight counting formulas for each traditional method and
 250 instantiate them with the exact hyperparameters used in our experiments. For deep learning models,
 251 we employed the PyTorch-based package `calflops`, which provides automatic FLOPs accounting
 252 given model architectures and input shapes (Ye, 2023).

253 In conducting FLOPs estimation, we adhere to the following principles:

- 254 • **Dataset specificity** FLOPs are computed separately for each dataset, as input dimensionality,
 255 sequence length, and sample size directly affect operation counts.
- 256 • **Training vs. inference** We compute FLOPs for both training and inference phases, reflecting
 257 their distinct computational characteristics.
- 258 • **Epoch sensitivity in deep learning models** Because training until convergence is ambiguous
 259 and dependent on hyperparameter optimization, we report FLOPs for a single epoch of training
 260 and the epoch at which the best AUROC is achieved during hyperparameter search.
- 261 • **Fair treatment of comparison operations** While FLOPs conventionally account only for
 262 addition and multiplication, algorithms such as k -NN and distance-based methods (ABOD,
 263 LOF) are dominated by comparison operations. Excluding these will unfairly understate their
 264 complexity, therefore, we count each comparison as a single FLOP.
- 265 • **Data-dependent structures** For models where computational complexity depends on data dis-
 266 tribution such as clustering-based models, FLOPs are computed from the fitted model structure
 267 on the actual data rather than theoretical worst-case bounds.
- 268 • **Approximation for non-primitive operations** For procedures such as sorting, where exact
 269 FLOPs are impractical to enumerate, we adopt widely accepted complexity-based approxima-
 270 tions (e.g. $O(n \log n)$, where n is the number of instances).

270 Table 2: Operation counts (FLOPs) for each method. We count additions, multiplications, and comparisons
 271 equally as 1 FLOP. The notation n_{tr} , n_{inf} , and d denote the number of training instances,
 272 the number of test instances, and the input dimension, respectively. Model-specific notations are
 273 explained in the Notation column.

Model	Type	FLOPs	Notation
Hotelling	Train	$2n_{\text{tr}}d^2 + 2n_{\text{tr}}d + d^3$	
	Inference	$n_{\text{inf}}(2d^2 + 2d - 1)$	
PCA	Train	$2n_{\text{tr}}d^2 + 2n_{\text{tr}}d + 3d^2$	$\bullet p$: # of PCA components
	Inference	$n_{\text{inf}}(4pd - p + 2d - 1)$	
ABOD		$1.5n_{\text{tr}}(n_{\text{tr}} - 1)d$	
	Train	$+n_{\text{tr}}(n_{\text{tr}} - 1) \log_2(n_{\text{tr}} - 1)$	
		$+n_{\text{tr}}k(k - 1)(d + 2) + n_{\text{tr}}$	$\bullet k$: # of neighbors
		$1.5n_{\text{inf}}(n_{\text{inf}} - 1)d$	
	Inference	$+n_{\text{inf}}(n_{\text{inf}} - 1) \log_2(n_{\text{inf}} - 1)$	
LOF		$+n_{\text{inf}}k(k - 1)(d + 2) + n_{\text{inf}}$	
		$1.5n_{\text{tr}}(n_{\text{tr}} - 1)d$	
	Train	$+n_{\text{tr}}(n_{\text{tr}} - 1) \log_2(n_{\text{tr}} - 1)$	
		$+n_{\text{tr}}k + n_{\text{tr}}(k + 1) + 2n_{\text{tr}}k$	$\bullet k$: # of neighbors
		$1.5n_{\text{inf}}(n_{\text{inf}} - 1)d$	
CBLOF	Inference	$+n_{\text{inf}}(n_{\text{inf}} - 1) \log_2(n_{\text{inf}} - 1)$	
		$+n_{\text{inf}}k + n_{\text{inf}}(k + 1) + 2n_{\text{inf}}k$	
	Train	$n_{\text{tr}}I(3Cd + d - 1) + 3d((n_{\text{tr}} - LC)L + LC)$	$\bullet I$: max iterations for clustering $\bullet C$: # of clusters $\bullet L$: # of large clusters $\bullet LC $: # of instances in large clusters
	Inference	$n_{\text{inf}}I(3Cd + d - 1) + 3d((n_{\text{inf}} - LC)L + LC)$	
HBOS	Train	$2n_{\text{tr}}d + 5bd$	$\bullet b$: # of bins
	Inference	$3n_{\text{inf}}d + 2bd$	
LODA	Train	$n_{\text{tr}}c(2\sqrt{d} + \log_2 b - 1)$	$\bullet b$: # of bins $\bullet c$: # of random cuts
	Inference	$n_{\text{inf}}(2c\sqrt{d} + c\log_2 b + 1)$	
Isolation Forest	Train	$T(2s \log_2 s)$	$\bullet T$: # of estimators $\bullet s$: max samples per estimator $\bullet \gamma$: Euler-Mascheroni const.
	Inference	$n_{\text{inf}}(T(2 \log s - 2 + \gamma) - 2(1 - 1/s) + (T + 2))$	
HS-Tree	Train	$T(\psi(h + 1) + 5(2^{h+1} - 1))$	$\bullet T$: # of estimators $\bullet h$: max depth of tree $\bullet \psi$: reference window size
	Inference	$n_{\text{inf}}T(5h + 7)$	

311 We summarize the final FLOPs formulations of the chosen methods in Table 2 and present their
 312 detailed derivations in Appendix C.

314 4 RESULTS AND ANALYSES

316 In this section we present three experiments to examine performance versus **computational cost**,
 317 estimated time comparison under hardware-constrained settings, and model scalability. The first ex-
 318 periment evaluates the trade-off between detection **performance**, **mainly** measured by AUROC, and
 319 the computational cost quantified by training and inference FLOPs. For traditional models, training
 320 and inference FLOPs are given in Table 2. For deep learning models, we report training FLOPs per
 321 epoch, and full-training FLOPs correspond to the sum overall epochs. In subsequent analyses, we
 322 use full-training FLOPs for deep learning models. The second experiment leverages the fact that,
 323 compared to other efficiency metrics, FLOPs have the key advantage of being directly compara-
 324 ble to hardware performance, typically expressed as floating-point operations per second (FLOPS),

324
 325 Table 3: **Top five models for each dataset ranked in descending order of AUROC. All computational**
 326 **costs are reported in gigaFLOPs (GFLOPs). AUROC and AUPRC scores are reported as the mean**
 327 **over five runs with different random seeds.**

328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362	328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362	328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362	328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362						
			Train	Inference	Full-training	AUROC \uparrow	AUPRC \uparrow		
331 332 333 334 335	331 332 333 334 335	331 332 333 334 335	Hotelling	Statistical	0.17	0.11	—	0.77 \pm 0.00	0.49 \pm 0.00
			LSTM-AE	Reconstruction	12.23	2.76	1223.5	0.76 \pm 0.01	0.50 \pm 0.04
			ABOD	Statistical	926.33	421.87	—	0.75 \pm 0.00	0.43 \pm 0.00
			LOF	One-class	917.76	416.07	—	0.73 \pm 0.00	0.42 \pm 0.00
			HBOS	Statistical	0.01	0.01	—	0.73 \pm 0.00	0.50 \pm 0.00
336 337 338 339 340	336 337 338 339 340	336 337 338 339 340	CBLOF	One-class	0.89	2.60	—	0.65 \pm 0.01	0.20 \pm 0.01
			HS-Tree	One-class	< 0.01	0.08	—	0.64 \pm 0.03	0.14 \pm 0.03
			TimesNet	Reconstruction	21851.23	9208.69	437024.6	0.64 \pm 0.03	0.16 \pm 0.01
			ABOD	Statistical	334.71	536.74	—	0.63 \pm 0.00	0.17 \pm 0.00
			HBOS	Statistical	0.01	0.01	—	0.62 \pm 0.00	0.16 \pm 0.00
			Isolation Forest	One-class	< 0.01	0.48	—	0.64 \pm 0.01	0.16 \pm 0.00
341 342 343 344 345	341 342 343 344 345	341 342 343 344 345	ABOD	Statistical	998.15	10281.98	—	0.64 \pm 0.00	0.17 \pm 0.00
			LOF	One-class	996.78	10277.64	—	0.62 \pm 0.00	0.17 \pm 0.00
			CBLOF	One-class	2.48	10.76	—	0.62 \pm 0.01	0.16 \pm 0.00
			HBOS	Statistical	0.01	0.03	—	0.61 \pm 0.00	0.15 \pm 0.00
			TimesNet	Reconstruction	25325.13	8441.89	506502.64	0.77 \pm 0.00	0.17 \pm 0.00
346 347 348 349 350	346 347 348 349 350	346 347 348 349 350	LSTM-AE	Reconstruction	82.70	27.57	8269.64	0.77 \pm 0.01	0.18 \pm 0.01
			Hotelling	Statistical	2.10	2.10	—	0.73 \pm 0.00	0.16 \pm 0.00
			CBLOF	One-class	36.30	34.65	—	0.72 \pm 0.01	0.16 \pm 0.01
			ABOD	Statistical	38426.94	38428.58	—	0.71 \pm 0.00	0.10 \pm 0.00
			HBOS	Statistical	0.05	0.07	—	0.85 \pm 0.00	0.75 \pm 0.00
351 352 353 354 355	351 352 353 354 355	351 352 353 354 355	Isolation Forest	One-class	< 0.01	0.36	—	0.83 \pm 0.00	0.73 \pm 0.01
			OmniAnomaly	Reconstruction	51.30	15.49	1026.05	0.83 \pm 0.00	0.73 \pm 0.00
			LODA	Statistical	1.39	1.39	—	0.82 \pm 0.02	0.73 \pm 0.02
			PCA	Reconstruction	2.64	2.42	—	0.82 \pm 0.00	0.73 \pm 0.00
			HBOS	Statistical	0.19	0.06	—	0.74 \pm 0.00	0.18 \pm 0.00
356 357 358 359	356 357 358 359	356 357 358 359	Isolation Forest	One-class	< 0.01	0.33	—	0.74 \pm 0.02	0.18 \pm 0.01
			LODA	Statistical	1.46	0.34	—	0.72 \pm 0.04	0.27 \pm 0.05
			TimesNet	Reconstruction	24899.00	1828.07	497979.94	0.66 \pm 0.01	0.20 \pm 0.00
			HS-Tree	One-class	< 0.01	0.20	—	0.63 \pm 0.05	0.09 \pm 0.02

363 where FLOPs denote algorithmic operation counts and FLOPS denote hardware throughput. Dividing
 364 algorithmic FLOPs by the FLOPS of a target hardware allows us to approximate training and
 365 inference time under GPU-free, hardware-constrained deployments. The third experiment examines
 366 scalability by varying the number of instances and feature dimensions, thereby observing how
 367 computational cost changes with dataset size. Model hyperparameters are chosen as those yielding the
 368 highest AUROC within the defined search space, with the detailed specification of the search space
 369 given in Appendix D.

370 4.1 PERFORMANCE VS. EFFICIENCY

371 Table 3 summarizes the main results, listing the top five models ranked by AUROC and allowing a
 372 direct inspection of which methods simultaneously offer high detection performance and favorable
 373 computational properties. Across individual datasets, high AUROC values are frequently achieved
 374 by approaches with relatively low computational cost. Although deep learning models often appear
 375 among the top performers in certain cases, their advantages are neither consistent nor substantial
 376 across datasets.

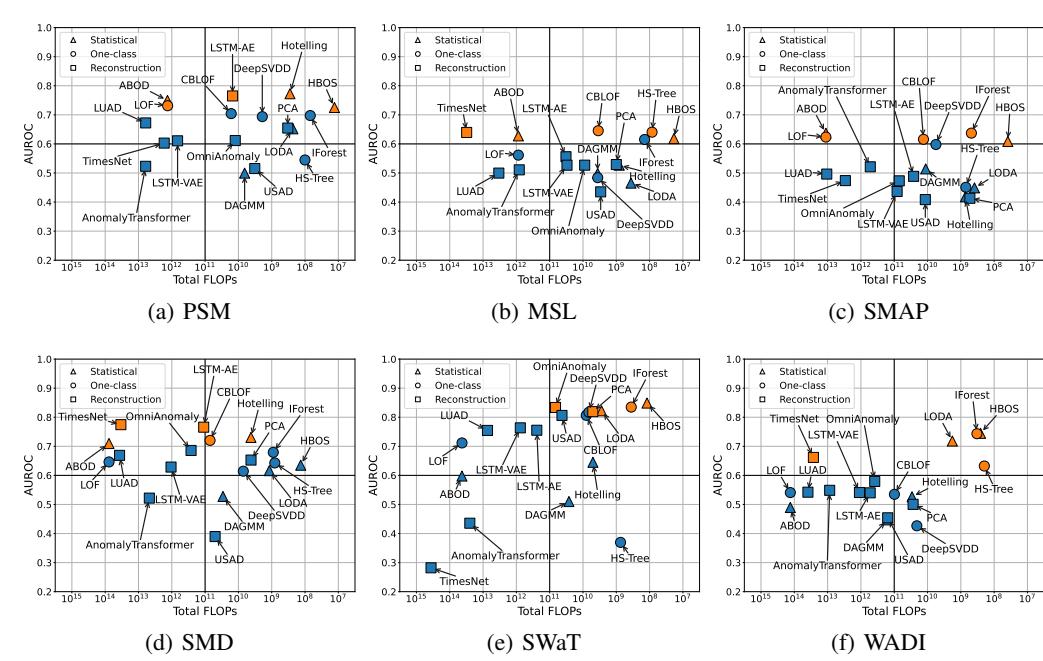


Figure 1: AUROC vs. Total FLOPs (sum of training and inference FLOPs) for each dataset. Orange markers denote the top five models in terms of AUROC.

Figure 1 further illustrates this trend by plotting AUROC against the total FLOPs, obtained by summing training and inference FLOPs, with the top-performing models highlighted in orange. A closer examination of the highlighted points shows that many traditional methods, aside from the k -NN family including ABOD and LOF, tend to attain high AUROC while remaining concentrated in the low-FLOPs region. These models therefore constitute practical and preferable alternatives in settings where computational resources are limited. It is also evident that approaches relying on k -NN exhibit noticeably higher FLOPs due to their quadratic complexity, which restricts their applicability in large-scale or resource-constrained environments.

In scenarios where computational resources are severely limited, the practical priority often shifts from maximizing raw detection accuracy to minimizing computational cost. To reflect this consideration, an additional table highlighting the five most computationally efficient models in terms of FLOPs is presented in Appendix E.1. This FLOPs-oriented view is intended for deployments where computational efficiency is essential and therefore offers a complementary perspective to the main results. We also report results for AUPRC, VUS-ROC, and VUS-PR. These metrics exhibit trends that closely align with AUROC, reinforcing the overall conclusion that models with low computational overhead generally achieve a favorable balance between detection performance and operational feasibility. Entire results for all models and datasets are also included in Appendix E.1.

4.2 COMPARISON OF ESTIMATED TIME

Estimate of the minimal execution time can be calculated as FLOPs/FLOPS. To capture variability across deployment conditions, we assume three hardware scenarios: a highly resourced environment represented by our experimental setup (Intel(R) Core(TM) i9-14900K CPU, NVIDIA GeForce RTX 5070 Ti GPU), a mobile environment corresponding to Samsung Galaxy A32 (Cortex-A75 & Cortex-A55 CPU, Arm Mali-G52 MC2 GPU), and a resource-constrained edge environment represented by Raspberry Pi 3B+ (Asutkar et al., 2023; Trilles et al., 2024). The FLOPS of each device are computed using only the number of cores C and clock frequency f , as formalized in Equation 3.

$$\text{FLOPS}_{CPU}^{peak} = \sum_{t \in \{\text{Performance, Efficient}\}} C_t \times f_t, \quad \text{FLOPS}_{GPU}^{peak} = C \times f \quad (3)$$

A practically significant observation emerging from Figure 2 is that deep learning models are feasible only in highly resourced environments, where the abundance of GPU cores substantially mit-

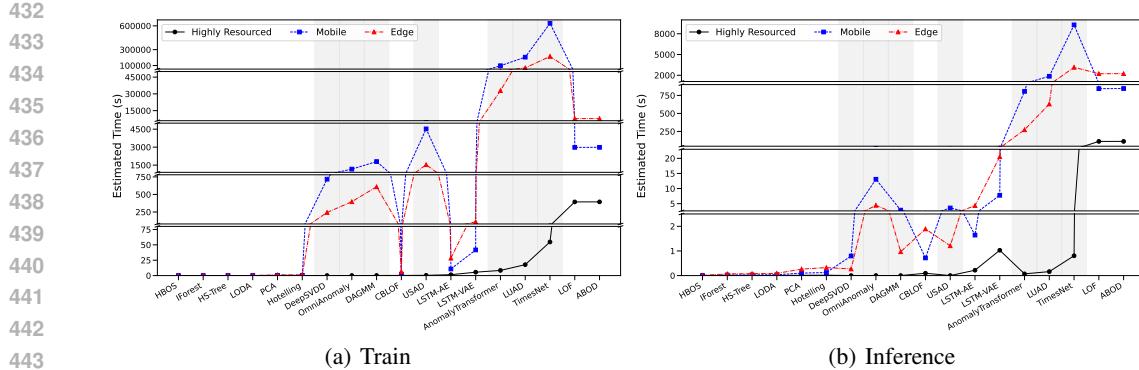


Figure 2: Estimated execution time of anomaly detection models under three hardware scenarios (Highly Resourced, Mobile, and Edge). Models are ordered in ascending runtime according to the Highly Resourced scenario, with deep learning models highlighted by gray shading. The results are averaged across datasets.

igates their computational burden. In sharp contrast, under mobile or edge scenarios these models incur prohibitively high costs, making their deployment virtually infeasible. By comparison, tree-based algorithms such as Isolation Forest and HS-Tree, along with histogram-based algorithms such as HBOS and LODA, consistently maintain extremely low computational overhead during both training and inference across all hardware settings. Finally, approaches that rely on k -NN, including ABOD and LOF, exhibit comparatively elevated costs, underscoring their limited practicality in data-abundant contexts. Results for individual datasets are provided in Appendix E.2.

To strengthen the practical relevance of these findings, we additionally report real runtime measurements under our highly resourced environment while disabling GPU acceleration to ensure a fair, CPU-only comparison. Using the SMD dataset and selecting representative models from each methodological category, HBOS required 3.27 seconds for training and 0.77 seconds for inference, while Isolation Forest required 4.97 seconds and 2.01 seconds, respectively. In contrast, the deep learning model USAD incurred 1047.6 seconds for training and 5.45 seconds for inference under the same setting. Finally, we also evaluated ABOD, which recorded the highest estimated time among all methods, and observed prohibitively high costs of 4700.05 seconds for training and 6331.64 seconds for inference. These results align closely with the FLOPs based trends and further substantiate the practical infeasibility of heavy models in resource-constrained deployments.

4.3 MODEL SCALABILITY

As illustrated in Figure 3, tree-based methods such as Isolation Forest and HS-Tree demonstrate high scalability, with computational requirements scaling sublinearly with respect to n , thereby making them well-suited for large-scale deployment. In contrast, methods using k -NN such as ABOD and LOF, exhibit a steep rise in both training and inference FLOPs **due to pairwise distance computations, making them impractical for large datasets**. Deep learning models exhibit a consistently linear increase in FLOPs with data size. Although they scale more favorably than k -NN approaches, their computational cost still poses a substantial burden as n becomes large.

A similar pattern appears when varying the feature dimension. Tree-based and histogram-based approaches, such as HS-Tree, Isolation Forest and HBOS, show only less sensitivity to increasing dimensionality, maintaining relatively low FLOPs across the range of dimensions. By contrast, k -NN methods increase linearly with dimension because each distance computation scales directly with the number of features. Deep learning models also scale approximately linearly, while their overall computational demand remains high. A particularly distinctive trend in dimensional scalability test is that Hotelling’s T^2 grows even more rapidly as dimension increases, since estimating and inverting the covariance matrix requires $O(d^2)$ and $O(d^3)$ operations, respectively.

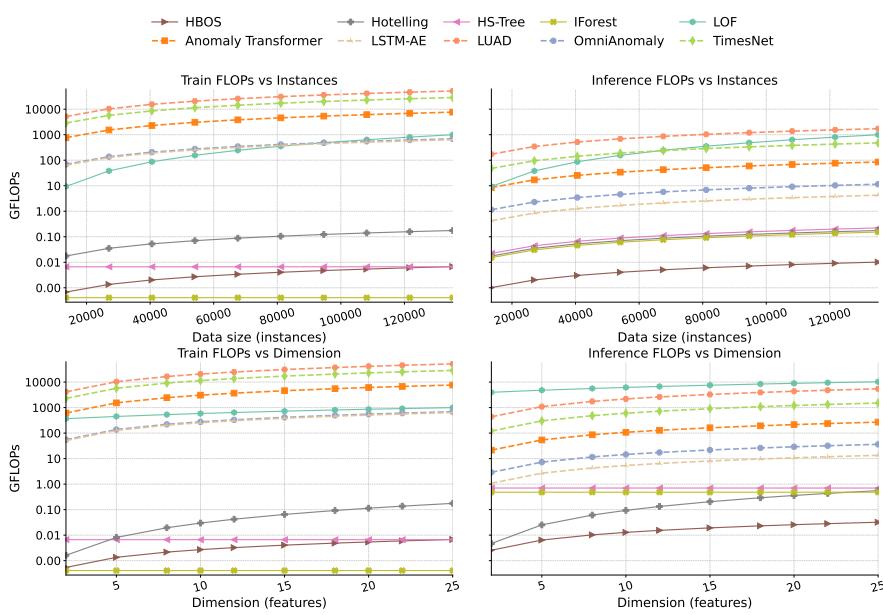


Figure 3: **Training (left) and inference (right) FLOPs of ten representative models on the SMAP dataset, plotted against the number of instances (top) and feature dimension (bottom). Both axes for FLOPs use a logarithmic scale.**

This divergence underscores the importance of considering algorithmic scalability when selecting anomaly detection models for real-world applications where data volumes can be substantial. Entire results for all models and datasets are provided in Appendix E.3.

5 CONCLUSION

This work presented a systematic comparison of traditional and deep learning methods for unsupervised time series anomaly detection under **resource**-constrained settings. By jointly evaluating AUROC and FLOPs, we explored two central questions concerning the effectiveness of models under limited resources and whether and how performance differences arise between traditional and deep learning methods. Our results showed that traditional models often rank among the top performers, deep learning models **tend to** do not consistently surpass them, and the balance between effectiveness and efficiency favors traditional approaches **except for k -NN methods**. Estimated time analysis further revealed that while deep learning models are feasible only in highly resourced environments, traditional models remain practical under resource-constrained settings while offering comparable detection performance. Overall, these findings challenge the view that deep learning is always the superior choice and emphasize the continued viability of traditional methods for real-world deployment. In doing so, we aim for this work to inspire new avenues of research while also providing practitioners with a useful point of reference when building anomaly detection systems under real-world constraints.

REPRODUCIBILITY STATEMENT

We have undertaken several efforts to ensure the reproducibility of our work. FLOPs derivations for all algorithms are provided in Appendix C, and the definition of hyperparameter search spaces is given in Appendix D. Extensive experimental results across datasets are presented in Appendix E. The supplementary material contains our full experimental pipeline, including model implementations, configuration files, and data preprocessing scripts, thereby facilitating independent reproduction of our findings.

540 REFERENCES
541

542 Ahmed Abdulaal, Zhuanghua Liu, and Tomer Lancewicki. Practical approach to asynchronous
543 multivariate time series anomaly detection and localization. In *Proceedings of the 27th ACM*
544 *SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 2485–2494, New
545 York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383325. doi: 10.
546 1145/3447548.3467174.

547 Chuadry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur. Wadi: a water distri-
548 bution testbed for research in the design of secure cyber physical systems. In *Proceedings of*
549 *the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, CySWA-
550 TER '17, pp. 25–28, New York, NY, USA, 2017. Association for Computing Machinery. ISBN
551 9781450349758. doi: 10.1145/3055366.3055375.

552 Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. Gandomaly: Semi-supervised
553 anomaly detection via adversarial training. In C. V. Jawahar, Hongdong Li, Greg Mori, and
554 Konrad Schindler (eds.), *Computer Vision – ACCV 2018*, pp. 622–637. Springer International
555 Publishing, 2019. ISBN 978-3-030-20893-6.

556 Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In Tapan
557 Elomaa, Heikki Mannila, and Hannu Toivonen (eds.), *Principles of Data Mining and Knowledge*
558 *Discovery*, pp. 15–27, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-
559 45681-0.

560 Supriya Asutkar, Chaitravi Chalke, Kajal Shivgan, and Siddharth Tallur. Tinyml-enabled edge
561 implementation of transfer learning framework for domain generalization in machine fault di-
562 agnosis. *Expert Systems with Applications*, 213:119016, 2023. ISSN 0957-4174. doi: <https://doi.org/10.1016/j.eswa.2022.119016>.

563 Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga.
564 Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th*
565 *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20,
566 pp. 3395–3404, New York, NY, USA, 2020. Association for Computing Machinery. ISBN
567 9781450379984. doi: 10.1145/3394486.3403392.

568 Aitor Belenguer, Javier Navaridas, and Jose A. Pascual. A review of federated learning in intrusion
569 detection systems for iot, 2022.

570 Deval Bhamare, Maede Zolanvari, Aiman Erbad, Raj Jain, Khaled Khan, and Nader Meskin. Cyber-
571 security for industrial control systems: A survey. *Computers & Security*, 89:101677, 2020. ISSN
572 0167-4048. doi: <https://doi.org/10.1016/j.cose.2019.101677>.

573 Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In
574 *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*, pp. 443–448,
575 2007. doi: 10.1137/1.9781611972771.42.

576 Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. A review on outlier/anomaly
577 detection in time series data. *ACM Comput. Surv.*, 54(3), April 2021. ISSN 0360-0300. doi:
578 10.1145/3444690.

579 Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: identifying density-
580 based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on*
581 *Management of Data*, SIGMOD '00, pp. 93–104, New York, NY, USA, 2000. Association for
582 Computing Machinery. ISBN 1581132174. doi: 10.1145/342009.335388.

583 Ayan Chatterjee and Bestoun S. Ahmed. Iot anomaly detection methods and applications: A survey.
584 *Internet of Things*, 19:100568, 2022. ISSN 2542-6605. doi: <https://doi.org/10.1016/j.iot.2022.100568>.

585 Ronit Das and Tie Luo. Lightesd: Fully-automated and lightweight anomaly detection framework
586 for edge computing. In *2023 IEEE International Conference on Edge Computing and Communi-
587 cations (EDGE)*, pp. 150–158, 2023. doi: 10.1109/EDGE60047.2023.00032.

594 Alireza Dehlaghi-Ghadim, Tijana Markovic, Miguel Leon, David Söderman, and Per Erik Strand-
 595 berg. Federated learning for network anomaly detection in a distributed industrial environment.
 596 In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pp. 218–225,
 597 2023. doi: 10.1109/ICMLA58977.2023.00038.

598 Daniel Dobos, Tien Thanh Nguyen, Truong Dang, Allan Wilson, Helen Corbett, John McCall,
 599 and Phil Stockton. A comparative study of anomaly detection methods for gross error detec-
 600 tion problems. *Computers & Chemical Engineering*, 175:108263, 2023. ISSN 0098-1354. doi:
 601 <https://doi.org/10.1016/j.compchemeng.2023.108263>.

603 Jin Fan, Zhentao Liu, Huifeng Wu, Jia Wu, Zhanyu Si, Peng Hao, and Tom H. Luan. Luad: A
 604 lightweight unsupervised anomaly detection scheme for multivariate time series data. *Neurocom-
 605 puting*, 557:126644, 2023. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2023.126644>.

606 Daniel Fährmann, Malte Ihlefeld, Arjan Kuijper, and Naser Damer. Resource-efficient anomaly
 607 detection in industrial control systems with quantized recurrent variational autoencoder. *IET
 608 Collaborative Intelligent Manufacturing*, 7(1):e70032, 2025. doi: [https://doi.org/10.1049/cim2-70032](https://doi.org/10.1049/cim2-

 609 70032).

611 Alexander Geiger, Dongyu Liu, Sarah Alnegheimish, Alfredo Cuesta-Infante, and Kalyan Veera-
 612 machaneni. Tadgan: Time series anomaly detection using generative adversarial networks.
 613 In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 33–43, 2020. doi:
 614 [10.1109/BigData50022.2020.9378139](https://doi.org/10.1109/BigData50022.2020.9378139).

615 Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised
 616 anomaly detection algorithm. *KI-2012: poster and demo track 9*, 2012.

618 Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network
 619 with pruning, trained quantization and huffman coding. In *International Conference on Learning
 620 Representations (ICLR)*, 2016.

622 Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. Adbench: Anomaly
 623 detection benchmark. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh
 624 (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 32142–32159. Curran
 625 Associates, Inc., 2022.

626 Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern
 627 Recognition Letters*, 24(9):1641–1650, 2003. ISSN 0167-8655. doi: [https://doi.org/10.1016/S0167-8655\(03\)00003-5](https://doi.org/10.1016/S0167-8655(03)00003-5).

630 H.Hotelling. Multivariate quality control illustrated by air testing of sample bombsights. *Techniques
 631 of statistical analysis*, pp. 111, 1947.

632 Thi Kieu Khanh Ho and Narges Armanfard. Contaminated multivariate time-series anomaly detec-
 633 tion with spatio-temporal graph conditional diffusion models. In *The 41st Conference on Uncer-
 634 tainty in Artificial Intelligence*, 2025.

636 Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom.
 637 Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Pro-
 638 ceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data
 639 Mining*, KDD '18, pp. 387–395, New York, NY, USA, 2018. Association for Computing Machinery.
 640 ISBN 9781450355520. doi: 10.1145/3219819.3219845.

641 Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard,
 642 Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for
 643 efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer
 644 Vision and Pattern Recognition (CVPR)*, June 2018.

646 Prarthi Jain, Seemandhar Jain, Osmar R. Zaïane, and Abhishek Srivastava. Anomaly detection in
 647 resource constrained environments with streaming data. *IEEE Transactions on Emerging Topics
 in Computational Intelligence*, 6(3):649–659, 2022. doi: 10.1109/TETCI.2021.3070660.

648 Xudong Jia, Peng Xun, Wei Peng, Baokang Zhao, Haojie Li, and Chiran Shen. Deep anomaly
 649 detection for time series: A survey. *Computer Science Review*, 58:100787, 2025. ISSN 1574-
 650 0137. doi: <https://doi.org/10.1016/j.cosrev.2025.100787>.

651 Bedeuro Kim, Mohsen Ali Alawami, Eunsoo Kim, Sanghak Oh, Jeongyong Park, and Hyoungshick
 652 Kim. A comparative study of time series anomaly detection models for industrial control systems.
 653 *Sensors*, 23(3), 2023. ISSN 1424-8220. doi: 10.3390/s23031310.

654 Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. Angle-based outlier detection in high-
 655 dimensional data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowl-
 656 edge Discovery and Data Mining*, KDD '08, pp. 444–452, New York, NY, USA, 2008. Associa-
 657 tion for Computing Machinery. ISBN 9781605581934. doi: 10.1145/1401890.1401946.

658 Max Landauer, Florian Skopik, Branka Stojanović, Andreas Flatscher, and Torsten Ullrich. A review
 659 of time-series analysis for cyber security analytics: from intrusion detection to attack prediction.
 660 *International Journal of Information Security*, 24(1):3, 2025.

661 Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. Mad-gan: Mul-
 662 tivariate anomaly detection for time series data with generative adversarial networks. In Igor V.
 663 Tetko, Věra Kůrková, Pavel Karpov, and Fabian Theis (eds.), *Artificial Neural Networks and
 664 Machine Learning – ICANN 2019: Text and Time Series*, pp. 703–716, Cham, 2019. Springer
 665 International Publishing. ISBN 978-3-030-30490-4.

666 Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE Interna-
 667 tional Conference on Data Mining*, pp. 413–422, 2008. doi: 10.1109/ICDM.2008.17.

668 Hou-I Liu, Marco Galindo, Hongxia Xie, Lai-Kuan Wong, Hong-Han Shuai, Yung-Hui Li, and
 669 Wen-Huang Cheng. Lightweight deep learning for resource-constrained environments: A survey.
 670 *ACM Comput. Surv.*, 56(10), June 2024a. ISSN 0360-0300. doi: 10.1145/3657282.

671 Qinghua Liu and John Paparrizos. The elephant in the room: Towards a reliable time-series anomaly
 672 detection benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tom-
 673 czak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp.
 674 108231–108261. Curran Associates, Inc., 2024.

675 Zhe Liu, Xiang Huang, Jingyun Zhang, Zhifeng Hao, Li Sun, and Hao Peng. Multivariate time-
 676 series anomaly detection based on enhancing graph attention networks with topological analysis.
 677 In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Man-
 678 agement*, CIKM '24, pp. 1555–1564, New York, NY, USA, 2024b. Association for Computing
 679 Machinery. ISBN 9798400704369. doi: 10.1145/3627673.3679614.

680 Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and
 681 Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *CoRR*,
 682 abs/1607.00148, 2016.

683 Aditya P. Mathur and Nils Ole Tippenhauer. Swat: a water treatment testbed for research and
 684 training on ics security. In *2016 International Workshop on Cyber-physical Systems for Smart
 685 Water Networks (CySWater)*, pp. 31–36, 2016. doi: 10.1109/CySWater.2016.7469060.

686 Nesryne Mejri, Laura Lopez-Fuentes, Kankana Roy, Pavel Chernakov, Enjie Ghorbel, and Djamila
 687 Aouada. Unsupervised anomaly detection in time-series: An extensive evaluation and analysis of
 688 state-of-the-art methods. *Expert Systems with Applications*, 256:124922, 2024. ISSN 0957-4174.
 689 doi: <https://doi.org/10.1016/j.eswa.2024.124922>.

690 Roland N. Mfondoum, Antoni Ivanov, Pavlina Koleva, Vladimir Poulkov, and Agata Manolova.
 691 Outlier detection in streaming data for telecommunications and industrial applications: A survey.
 692 *Electronics*, 13(16), 2024. ISSN 2079-9292. doi: 10.3390/electronics13163339.

693 Periasamy Nancy, S. Muthurajkumar, S. Ganapathy, S.V.N. Santhosh Kumar, M. Selvi, and Kannan
 694 Arputharaj. Intrusion detection using dynamic feature selection and fuzzy temporal decision
 695 tree classification for wireless sensor networks. *IET Communications*, 14:888–895, 2020. doi:
 696 10.1049/iet-com.2019.0172.

702 Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for
 703 anomaly detection: A review. *ACM Comput. Surv.*, 54(2), March 2021. ISSN 0360-0300. doi:
 704 10.1145/3439950.

705 John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S. Tsay, Themis Palpanas, and Michael J. Franklin.
 706 Tsb-ud: an end-to-end benchmark suite for univariate time-series anomaly detection. *Proc.
 707 VLDB Endow.*, 15(8):1697–1711, April 2022. ISSN 2150-8097. doi: 10.14778/3529337.
 708 3529354.

709 Daehyung Park, Yuuna Hoshi, and Charles C. Kemp. A multimodal anomaly detector for robot-
 710 assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation
 711 Letters*, 3(3):1544–1551, 2018. doi: 10.1109/LRA.2018.2801475.

712 Tomáš Pevný. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–
 713 304, 2016.

714 Xiangfei Qiu, Zhe Li, Wanghui Qiu, Shiyuan Hu, Lekui Zhou, Xingjian Wu, Zhengyu Li, Chenjuan
 715 Guo, Aoying Zhou, Zhenli Sheng, Jilin Hu, Christian S. Jensen, and Bin Yang. TAB: Unified
 716 benchmarking of time series anomaly detection methods. In *Proc. VLDB Endow.*, 2025.

717 Ferdinand Rewicki, Joachim Denzler, and Julia Niebling. Is it worth it? comparing six deep and
 718 classical methods for unsupervised anomaly detection in time series. *Applied Sciences*, 13(3),
 719 2023. ISSN 2076-3417. doi: 10.3390/app13031778.

720 Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexan-
 721 der Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy
 722 and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learn-
 723 ing*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4393–4402. PMLR, 10–15 Jul
 724 2018.

725 Mahsa Salehi and Lida Rashidi. A survey on anomaly detection in evolving data: [with application
 726 to forest fire risk prediction]. *SIGKDD Explor. Newsl.*, 20(1):13–23, May 2018. ISSN 1931-0145.
 727 doi: 10.1145/3229329.3229332.

728 Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: a
 729 comprehensive evaluation. *Proc. VLDB Endow.*, 15(9):1779–1797, May 2022. ISSN 2150-8097.
 730 doi: 10.14778/3538598.3538602.

731 Md. Maruf Hossain Shuvo, Syed Kamrul Islam, Jianlin Cheng, and Bashir I. Morshed. Efficient
 732 acceleration of deep learning inference on resource-constrained edge devices: A review. *Pro-
 733 ceedings of the IEEE*, 111(1):42–91, 2023. doi: 10.1109/JPROC.2022.3226481.

734 Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly
 735 detection scheme based on principal component classifier. *Technical report, Miami Univ Coral
 736 Gables Fl Dept of Electrical and Computer Engineering*, 2003.

737 Haotian Si, Jianhui Li, Changhua Pei, Hang Cui, Jingwen Yang, Yongqian Sun, Shenglin Zhang,
 738 Jingjing Li, Haiming Zhang, Jing Han, Dan Pei, and Gaogang Xie. Timeseriesbench: An
 739 industrial-grade benchmark for time series anomaly detection models. In *2024 IEEE 35th In-
 740 ternational Symposium on Software Reliability Engineering (ISSRE)*, pp. 61–72, 2024. doi:
 741 10.1109/ISSRE62328.2024.00017.

742 Raghubir Singh and Sukhpal Singh Gill. Edge ai: A survey. *Internet of Things and Cyber-Physical
 743 Systems*, 3:71–92, 2023. ISSN 2667-3452. doi: <https://doi.org/10.1016/j.iotcps.2023.02.004>.

744 Tuomo Sipola, Janne Alatalo, Tero Kokkonen, and Mika Rantonen. Artificial intelligence in the iot
 745 era: A review of edge ai hardware and software. In *2022 31st Conference of Open Innovations
 746 Association (FRUCT)*, pp. 320–331, 2022. doi: 10.23919/FRUCT54823.2022.9770931.

747 Keith Stouffer, Keith Stouffer, Michael Pease, CheeYee Tang, Timothy Zimmerman, Victoria Pillit-
 748 teri, Suzanne Lightman, Adam Hahn, Stephanie Saravia, Aslam Sherule, et al. Guide to opera-
 749 tional technology (ot) security. *National Institute of Standards and Technology*, 2023.

756 Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection
 757 for multivariate time series through stochastic recurrent neural network. In *Proceedings of the*
 758 *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD
 759 '19, pp. 2828–2837, New York, NY, USA, 2019. Association for Computing Machinery. ISBN
 760 9781450362016. doi: 10.1145/3292500.3330672.

761 Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. Fast anomaly detection for streaming data.
 762 In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence -*
 763 *Volume Volume Two*, IJCAI'11, pp. 1511–1516. AAAI Press, 2011. ISBN 9781577355144.

764

765 Sergio Trilles, Sahibzada Saadoon Hammad, and Ditsuhi Iskandaryan. Anomaly detection based on
 766 artificial intelligence of things: A systematic literature mapping. *Internet of Things*, 25:101063,
 767 2024. ISSN 2542-6605. doi: <https://doi.org/10.1016/j.iot.2024.101063>.

768 Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. Tranad: deep transformer networks
 769 for anomaly detection in multivariate time series data. *Proc. VLDB Endow.*, 15(6):1201–1214,
 770 February 2022. ISSN 2150-8097. doi: 10.14778/3514061.3514067.

771

772 Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet:
 773 Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International*
 774 *Conference on Learning Representations*, 2023.

775 Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series
 776 anomaly detection with association discrepancy. In *International Conference on Learning Repre-*
 777 *sentations*, 2022.

778 Zheng Xu, Yumeng Yang, Xinwen Gao, and Min Hu. Dcff-mtd: A multivariate time-series anomaly
 779 detection model based on dual-channel feature fusion. *Sensors*, 23(8), 2023. ISSN 1424-8220.
 780 doi: 10.3390/s23083910.

781

782 Nong Ye and Qiang Chen. An anomaly detection technique based on a chi-square statistic for
 783 detecting intrusions into information systems. *Quality and Reliability Engineering International*,
 784 17(2):105–112, 2001. doi: <https://doi.org/10.1002/qre.392>.

785 Xiaoju Ye. Calflops: A flops and params calculate tool for neural networks in pytorch framework,
 786 2023.

787

788 Zahra Zamanzadeh Darban, Geoffrey I. Webb, Shirui Pan, Charu Aggarwal, and Mahsa Salehi.
 789 Deep learning for time series anomaly detection: A survey. *ACM Comput. Surv.*, 57(1), October
 790 2024. ISSN 0360-0300. doi: 10.1145/3691338.

791

792 Dequan Zheng, Fenghuan Li, and Tiejun Zhao. Self-adaptive statistical process control for anomaly
 793 detection in time series. *Expert Systems with Applications*, 57:324–336, 2016. ISSN 0957-4174.
 794 doi: <https://doi.org/10.1016/j.eswa.2016.03.029>.

795

796 Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. Beatgan: anomalous rhythm
 797 detection using adversarially generated time series. In *Proceedings of the 28th International*
 798 *Joint Conference on Artificial Intelligence*, IJCAI'19, pp. 4433–4439. AAAI Press, 2019. ISBN
 9780999241141.

799

800 Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng
 801 Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *Inter-*
 802 *national Conference on Learning Representations*, 2018.

803

804

805

806

807

808

809

810
811
812
813
814
A PREVIOUS BENCHMARKS815
816
817
818
819
820
821
822
823
824
825
826
827
Table 4: Previous benchmark papers considering computational cost.

Paper Type	Paper	Cost Metric	Coverage Model Type			Data Source	Dimension
			Stat.	ML	DL		
Benchmark	Schmidl et al. (2022)	Memory, Time	✓	✓	✓	✓	✓
	Paparrizos et al. (2022)	Time	✓	✓	✓	✓	✗
	Han et al. (2022)	Time	✓	✓	✓	✓	✓
	Dobos et al. (2023)	Time	✓	✓	✓	✗	✓
	Rewicki et al. (2023)	Time	✓	✓	✓	✓	✗
	Liu & Paparrizos (2024)	Time	✓	✓	✓	✓	✓
	Si et al. (2024)	Time	✓	✗	✓	✓	✓
Methodology	Qiu et al. (2025)	Memory, Time	✓	✓	✓	✗	✓
	Xu et al. (2023)	Params, FLOPs, Time	✗	✗	✓	✓	✓
	Liu et al. (2024b)	Params, FLOPs, Time	✗	✗	✓	✓	✓
	Ho & Armanfard (2025)	FLOPs, Time	✗	✗	✓	✓	✓
Ours		FLOPs	✓	✓	✓	✓	✓

828
829 As summarized in Table 4, a number of benchmark studies on multivariate time series anomaly
830 detection have undertaken extensive comparisons across models and datasets. Despite this broad
831 of coverage, the evaluation of computational cost in these works remains limited in scope. Most
832 benchmarks have relied primarily on execution time as the cost metric, with a few additionally con-
833 sidering memory usage. However, runtime measurements are inherently dependent on hardware
834 specifications and experimental settings, which constrains their comparability across studies. Like-
835 wise, memory consumption does not fully capture the algorithmic complexity of the models and
836 therefore provides only a partial view of computational efficiency.

837 Beyond benchmark papers, some methodology studies have employed hardware-agnostic measures
838 such as parameter counts or FLOPs. However, these comparisons have typically been restricted to
839 deep learning models, leaving traditional statistical and machine learning approaches unexamined
840 even though they have compatible detection performance with deep learning models. To bridge this
841 gap, we adopt FLOPs, a hardware-agnostic metric, and apply it to both traditional and deep learning
842 models. This unified treatment enables fair and reproducible comparisons of computational cost
843 across paradigms and provides practitioners with a principled basis for model selection in resource-
844 constrained environments.

845
846 B ALGORITHMS AND DATASETS
847848 B.1 ALGORITHMS
849

850 **Hotelling** (H. Hotelling, 1947). A multivariate statistical process control method that scores each
851 observation via its Mahalanobis distance under a Gaussian reference model, thereby capturing cor-
852 related variation across variables.

853 **PCA** (Shyu et al., 2003). Principal Component Analysis models normal structure in a low-
854 dimensional subspace. Deviations are quantified through reconstruction error. Anomalies arise
855 when observations project poorly onto the principal subspace.

856 **ABOD** (Kriegel et al., 2008). Angle-Based Outlier Detection ranks points by the variance of an-
857 gles formed with all other points, exploiting the geometric insight that outliers yield concentrated
858 angle distributions in high dimensions. Practical variants use subsampling or k -nearest neighbor-
859 hoods to reduce the quadratic cost while preserving discrimination.

860 **LOF** (Breunig et al., 2000). Local Outlier Factor contrasts a point’s local reachability density
861 with that of its neighbors to assess how isolated it is within its immediate neighborhood. Large LOF

864 scores indicate locally sparse regions, enabling detection of context-dependent anomalies that global
 865 density models often miss.
 866

867 **CBLOF** (He et al., 2003). Cluster-Based LOF assigns each point to a large or small cluster and
 868 computes scores from cluster size and distance to representative large clusters. Points in small,
 869 distant clusters receive high scores, capturing both rarity and separation.
 870

871 **HBOS** (Goldstein & Dengel, 2012). Histogram-Based Outlier Score approximates feature-wise
 872 densities with univariate histograms and aggregates inverse densities across dimensions, implicitly
 873 assuming weak dependence.
 874

875 **LODA** (Pevný, 2016). The Lightweight online Detector of Anomalies ensembles sparse ran-
 876 dom projections, building one-dimensional histograms in each projected space and combining their
 877 anomaly evidences.
 878

879 **Isolation Forest** (Liu et al., 2008). Random partitioning via isolation trees isolates anomalies
 880 with fewer splits, producing shorter expected path lengths than normal points. Scores are obtained
 881 by normalizing path lengths against the average in random trees, enabling fast, distribution-agnostic
 882 detection.
 883

884 **HS-Tree** (Tan et al., 2011). Half-Space Trees construct randomized, axis-aligned partitions geared
 885 for streaming one-class detection. Points that consistently fall into underpopulated half-spaces ob-
 886 tain higher anomaly scores.
 887

888 **DAGMM** (Zong et al., 2018). The Deep Autoencoding Gaussian Mixture Model jointly learns
 889 a compact representation and a GMM density in an end-to-end fashion, combining reconstruction
 890 features with mixture-based energy for scoring. This coupling allows the representation to align
 891 with density estimation, improving separability of rare patterns.
 892

893 **DeepSVDD** (Ruff et al., 2018). A deep one-class objective trains a network to map normal data
 894 into a minimal radius hypersphere in latent space, penalizing distances from a fixed center. Samples
 895 that lie far from this center at test time are flagged as anomalies, avoiding reconstruction bias inherent
 896 to autoencoders.
 897

898 **LSTM-AE** (Malhotra et al., 2016). A sequence-to-sequence LSTM autoencoder learns normal
 899 temporal dynamics and emits reconstruction errors over sliding windows. Sustained or abrupt in-
 900 creases in error indicate departures from learned patterns, capturing both gradual drifts and transient
 901 spikes.
 902

903 **LSTM-VAE** (Park et al., 2018). A variational sequence model with LSTM encoder-decoder es-
 904 timates a probabilistic generative process, enabling anomaly scoring via low evidence lower bound
 905 or high reconstruction error.
 906

907 **USAD** (Audibert et al., 2020). A dual-autoencoder architecture trained with an adversarial-
 908 inspired objective where two decoders reconstruct each other’s outputs to improve robustness. At
 909 inference, a calibrated combination of the two reconstruction errors yields stable anomaly scores
 910 with strong generalization across regimes.
 911

912 **OmniAnomaly** (Su et al., 2019). A stochastic recurrent VAE augmented with normalizing flows
 913 models complex temporal dependencies and heteroscedastic noise, producing likelihood-based
 914 anomaly scores, negative log-probability, its latent dynamics capture both short and long range de-
 915 pendencies.
 916

917 **LUAD** (Fan et al., 2023). A lightweight unsupervised detector that combines efficient temporal
 918 encoder, TCN, with a compact probabilistic module and an auxiliary diagnosis head.
 919

918
919
920
921
922
923
924
925
926
927
928
929
930
Table 5: Dataset overview.
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972

Dataset	Dimensions	Train size	Test size	Anomaly Ratio (%)
PSM	25	129784	87841	27.76
MSL	55	58317	73729	10.53
SMAP	25	135183	427617	12.79
SMD	38	708405	708420	4.16
SWaT	51	496800	449919	12.14
WADI	123	784537	172801	5.77

Anomaly Transformer Xu et al. (2022). Anomaly Transformer models time point relationships through a Gaussian prior and self attention based series association, using their association discrepancy as a discriminative anomaly score.

TimesNet Wu et al. (2023). TimesNet transforms one dimensional sequences into multi scale two dimensional periodic representations and learns temporal patterns through convolution blocks, providing string reconstruction features for anomaly detection.

B.2 DATASETS

Table 5 summarizes the key statistics of the benchmark datasets, including the input dimension, the size of the train set, the size of the test set, and the anomaly ratio. Note that anomalies appear only in test sets, and the reported ratios are calculated with respect to the test instances. In addition to these summary statistics, we provide a description of the key characteristics of each dataset below.

PSM (Abdulaal et al., 2021). The Pooled Server Metrics dataset consists of multivariate time series monitoring server behavior, including signals such as CPU utilization and memory usage. It contains 13 weeks of training data and 8 weeks of testing data. Anomalies are present in both splits, while labels are provided only for the test set and include both planned and unplanned events.

MSL (Hundman et al., 2018). The Mars Science Laboratory dataset was constructed from telemetry of NASA’s Curiosity rover. Anomalies were extracted from Incident Surprise, Anomaly reports (ISA) and manually labeled across channels.

SMAP (Hundman et al., 2018). The Soil Moisture Active Passive dataset was derived from telemetry collected during NASA’s satellite mission. Anomalies were identified through ISA, which document unexpected spacecraft events during post-launch operations.

SMD (Su et al., 2019). The Server Machine Dataset is a 5 week multivariate time series collection gathered from a large Internet company. It comprises logs with metrics such as CPU load, memory usage, disk activity, and network traffic. The dataset is partitioned into training and testing halves, with anomalies in the testing portion labeled by domain experts based on incident reports.

SWaT (Mathur & Tippenhauer, 2016). The Secure Water Treatment dataset was collected from a fully operational 6 stage water treatment testbed. It comprises readings from sensors and actuators recorded every second over 11 consecutive days, including 7 days of normal operation and 4 days with controlled cyber-physical attacks. All attack instances were labeled by experts.

WADI (Ahmed et al., 2017). The Water Distribution dataset is derived from a scaled-down water distribution network testbed that simulates real industrial control systems. It contains multivariate time series of sensor and actuator signals across different stages of water storage and distribution. The dataset includes normal operations as well as periods with cyber-physical attack scenarios, with labels provided for the anomalous events.

972 **C DERIVATIONS OF FLOPs**
 973

974 This section provides a detailed account of the FLOPs computations for both traditional and deep
 975 learning models. For traditional models, FLOPs are derived from the algorithmic procedures, with
 976 training and inference costs obtained by applying the formulas to the entire training and test sets,
 977 respectively. In contrast, deep learning models operate on sequences generated by a rolling window.
 978 To ensure consistency, all calculations used the same input shape, determined by the window length
 979 and feature dimension of each dataset. The FLOPs of deep learning models were computed using the
 980 `calflops` package, applied to our predefined model architectures. Training FLOPs were estimated
 981 as the sum of forward and backward passes, whereas inference FLOPs were measured from the
 982 forward pass alone. Thus, training FLOPs are calculated as the operations required to process one
 983 epoch of windowed training data with both forward and backward passes, while inference FLOPs
 984 correspond to a forward pass over the windowed test set. The full-training FLOPs are then obtained
 985 by multiplying the training FLOPs per epoch by the number of epochs used for each model. For
 986 comparability across models, all FLOPs are expressed in GFLOPs.
 987

988 **Global notation and counting rule** Given a common vector $x \in \mathbb{R}^d$, we use following notations:
 989 n_{tr} denotes the number of training instances, n_{inf} the number of inference instances, and d the input
 990 dimensionality. Also, we denote FLOPs per sample by f , and the total FLOPs over the dataset by
 991 F . If an algorithm supports per-sample inference, we report both f and F . For models that do not
 992 involve a distinct training phase, we set $n = n_{\text{tr}} = n_{\text{inf}}$.

993 Additions, multiplications, divisions, and comparisons are all counted as 1 FLOP, and the FLOPs
 994 required for matrix multiplication between $A \in \mathbb{C}^{M \times N}$ and $B \in \mathbb{C}^{N \times L}$ are calculated as
 995

$$F_{AB} = 2MNL - ML$$

996 which consists of MNL multiplication and $ML(N - 1)$ additions.
 997

998 **C.1 HOTELLING**
 999

1000 **Training FLOPs** Hotelling's statistic is

$$T^2 = (x - \mu)^\top \Sigma^{-1} (x - \mu),$$

1003 where $\mu \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d \times d}$ are mean and covariance of the training set. In the training phase, the
 1004 inverse covariance matrix is calculated.

1005 (i) *Mean*: $\mu = \frac{1}{n_{\text{tr}}} \sum_i x_i$ costs

$$F_\mu = (n_{\text{tr}} - 1)d + d = n_{\text{tr}}d.$$

1008 (ii) *Covariance*: $\Sigma = \frac{1}{n_{\text{tr}} - 1} \sum_{i=1}^{n_{\text{tr}}} (x_i - \mu)(x_i - \mu)^\top$ costs

$$\begin{aligned} F_\Sigma &= n_{\text{tr}}d + n_{\text{tr}}d^2 + (n_{\text{tr}} - 1)d^2 + d^2 \\ &= n_{\text{tr}}(2d^2 + d) \end{aligned}$$

1013 with respect to subtraction, multiplication, addition, and division.

1014 (iii) *Inverse*: Inverting Σ costs

$$F_{\Sigma^{-1}} \approx d^3.$$

1017 Summing up,

$$\begin{aligned} F^{\text{train}} &= F_\mu + F_\Sigma + F_{\Sigma^{-1}} \\ &= 2n_{\text{tr}}d^2 + 2n_{\text{tr}}d + d^3. \end{aligned}$$

1021 **Inference FLOPs** In the inference phase, the monitoring statistics T^2 calculations are performed
 1023 sample by sample.

1024 (i) *Centering subtraction*: $v = x - \mu$ costs

$$f_{\text{cen}} = d.$$

1026 (ii) *Matrix multiplication*: $T^2 = v^\top \Sigma^{-1} v$ costs
 1027

$$f_{\text{mat}} = 2d^2 + d - 1.$$

1029 Therefore,

$$\begin{aligned} f^{\text{infer}} &= f_{\text{cen}} + f_{\text{mat}} = 2d^2 + 2d - 1, \\ 1031 F^{\text{infer}} &= n_{\text{inf}}(2d^2 + 2d - 1). \end{aligned}$$

1033 **C.2 PCA**

1035 **Model-specific notation** p : number of principal components retained ($p \leq d$).
 1036

1037 **Training FLOPs** We estimate the mean and covariance exactly as in section C.1, and then obtain
 1038 the top- p eigenspace with QR decomposition.

1039 (i) *Mean*: $\mu = \frac{1}{n_{\text{tr}}} \sum_i x_i$ costs

$$F_\mu = (n_{\text{tr}} - 1)d + d = n_{\text{tr}}d.$$

1040 (ii) *Covariance*: $\Sigma = \frac{1}{n_{\text{tr}}-1} \sum_i^{n_{\text{tr}}} (x_i - \mu)(x_i - \mu)^\top$ costs

$$F_\Sigma \approx n_{\text{tr}}(2d^2 + d).$$

1041 (iii) *QR decomposition of matrix*: Computing QR decomposition costs

$$F_{\text{dec}} \approx 3d^2.$$

1042 Summing up,

$$\begin{aligned} F^{\text{train}} &= F_\mu + F_\Sigma + F_{\text{dec}} \\ 1051 &= 2n_{\text{tr}}d^2 + 2n_{\text{tr}}d + 3d^2. \end{aligned}$$

1053 **Inference FLOPs** In the inference phase, reconstructions can be performed sample by sample.
 1054 Given $x \in \mathbb{R}^d$, let $U_p \in \mathbb{R}^{d \times p}$ be the loading matrix.

1055 (i) *Projection*: $z = U_p^\top x$ costs

$$f_{\text{proj}} = p(2d - 1).$$

1056 (ii) *Reconstruction*: $\hat{x} = U_p z$ costs

$$f_{\text{rec}} = d(2p - 1).$$

1057 (iii) *Error calculation*: $e = \|x - \hat{x}\|^2 = \sum_{j=1}^d (x_j - \hat{x}_j)^2$ costs

$$f_{\text{err}} = 3d - 1.$$

1058 Therefore, for each instance and total n_{inf} instances,

$$f^{\text{infer}} = f_{\text{proj}} + f_{\text{err}} + f_{\text{rec}} = 4pd - p + 2d - 1,$$

$$F^{\text{infer}} = n_{\text{inf}}(4pd - p + 2d - 1).$$

1059 **C.3 ABOD**

1060 **Model-specific notation** k : number of neighbors for Fast-ABOD.

1061 **Training/Inference FLOPs** We first prepare k -NN neighborhoods, then evaluate the Angle Based
 1062 Outlier Factor (ABOF) score for a point using its k neighbors. This is Fast-ABOD which approxi-
 1063 mate original ABOD.

1064 (i) *All-pairs of Euclidean distances*: Computing $\|x_i - x_j\|_2 = \sqrt{\sum_{l=1}^d (x_{il} - x_{jl})^2}$ costs $3d$. If
 1065 span this to all-pairs, it costs

$$F_{\text{dist}} = \binom{n}{2}(3d) = \frac{3}{2}n(n-1)d.$$

1080 (ii) *Sorting distances*: Sorting algorithms have approximated complexity of $O(n \log_2 n)$. Therefore,
 1081 sorting all-pair distances costs

$$f_{\text{sort}} \approx (n-1) \log_2(n-1), \quad F_{\text{sort}} \approx n(n-1) \log_2(n-1).$$

1082 (iii) *ABOF calculation*: ABOF is calculated by $\text{VAR}_{B,C \in N_k(A)} \left(\frac{\langle \overline{AB}, \overline{AC} \rangle}{\|\overline{AB}\|^2 \|\overline{AC}\|^2} \right)$ and $\binom{k}{2} = \frac{1}{2}k(k-1)$
 1083 is the number of neighbor pair cases.

1084 Reusing calculated distances, each pair needs one dot product and operations for multiplication and
 1085 normalization. Therefore, each pair needs $2d+1$ FLOPs and spans $\frac{1}{2}k(k-1)$ times within one
 1086 sample.

1087 With $\frac{1}{2}k(k-1)$ pairs, $\text{VAR} = \frac{1}{N} \sum v^2 - \left(\frac{1}{N} \sum v \right)^2$ costs $\frac{3}{2}k(k-1) + 1$.

1088 Then,

$$f_{\text{abof}} = \frac{1}{2}k(k-1)(2d+1) + \frac{3}{2}k(k-1) + 1 = k(k-1)(d+2) + 1$$

1089 and

$$F_{\text{abof}} = nk(k-1)(d+2) + n.$$

1090 Summing up,

$$\begin{aligned} F^{\text{train/infer}} &= F_{\text{dist}} + F_{\text{sort}} + F_{\text{abof}} \\ &= 1.5n(n-1)d + n(n-1) \log_2(n-1) + nk(k-1)(d+2) + n. \end{aligned}$$

1100 C.4 LOF

1101 **Model-specific notation** k : number of neighbors.

1102 **Training/Inference FLOPs** We build k -NN neighborhoods and then compute reachability dis-
 1103 tances, local reachability density, and the LOF score.

1104 (i) *All-pairs of Euclidean distances*: As calculated at Section C.3, it costs

$$F_{\text{dist}} = \binom{n}{2}(3d) = \frac{3}{2}n(n-1)d.$$

1105 (ii) *Sorting distances*: Sorting algorithms have approximated complexity of $O(n \log_2 n)$. Therefore,
 1106 sorting all-pair distances costs

$$f_{\text{sort}} \approx (n-1) \log_2(n-1), \quad F_{\text{sort}} \approx n(n-1) \log_2(n-1).$$

1107 (iii) *Reachability distances*: For each point x_p and $x_o \in N_k(x_p)$, comparison operation
 1108 $\text{reach_dist}(x_p, x_o) = \max\{\text{dist}_k(x_o), \text{dist}(x_p, x_o)\}$ is conducted. Total comparison costs

$$f_{\text{reach}} = k, \quad F_{\text{reach}} = nk.$$

1109 (iv) *Local reachability density (LRD)*: Formulation of LRD is

$$\text{LRD}(x_p) = \left(\frac{1}{N_k(x_p)} \sum_{x_o \in N_k(x_p)} \text{reach_dist}(x_p, x_o) \right)^{-1}.$$

1110 Per point, $k-1$ additions, 1 division, and 1 scaling is conducted with total $k+1$ FLOPs.

1111 Therefore,

$$f_{\text{lrdd}} = k+1, \quad F_{\text{lrdd}} = n(k+1).$$

1112 (v) *LOF score*: Formulation of LOF score is

$$\text{LOF}(x_p) = \frac{1}{k} \sum_{x_o \in N_k(x_p)} \frac{\text{LRD}(x_o)}{\text{LRD}(x_p)}.$$

1113 Per point, k divisions, $k-1$ additions, 1 division is conducted with total $2k$ FLOPs.

1114 Therefore,

$$f_{\text{lof}} = 2k, \quad F_{\text{lof}} = 2nk.$$

1115 Summing up,

$$\begin{aligned} F^{\text{train/infer}} &= F_{\text{dist}} + F_{\text{sort}} + F_{\text{reach}} + F_{\text{lrdd}} + F_{\text{lof}} \\ &= 1.5n(n-1)d + n(n-1) \log_2(n-1) + nk + n(k+1) + 2nk. \end{aligned}$$

1134 C.5 CBLOF
11351136 **Model-specific notation** I : maximum k-means iterations, C : number of clusters, L : number of
1137 large clusters, $|LC|$: number instances in large clusters.
11381139 **Training/Inference FLOPs** CBLOF fits C centroids with k-means and then scores samples using
1140 the large-small cluster partition.1141 (i) *K-means cluster assignment*: For each sample $x \in \mathbb{R}^d$ and each centroid c , squared distance
1142 $\|x - c\|^2$ costs $3d - 1$ FLOPs. Also, finding minimum distance centroid over C centroids contributes
1143 $C - 1$ comparisons.
11441145 Over n points and one iteration, it costs
1146

1147
$$F_{\text{assign/iter}} = n [C(3d - 1) + (C - 1)].$$

1148

1149 (ii) *K-means centroid update*: For each centroid, we accumulate assigned points and normalize
1150 once. As we have n points and C centroids, accumulation costs $d(n - C)$ and normalization costs
1151 Cd making total FLOPs for centroid update is nd .
11521153 Over n points and one iteration, it costs
1154

1155
$$F_{\text{update/iter}} = nd.$$

1156

1157 Therefore,
1158

1159
$$F_{\text{kmeans}} = I(F_{\text{assign/iter}} + F_{\text{update/iter}})$$

1160
$$= I(3Cd - n + nd).$$

1161

1162 (iii) *Scoring with large/small partition*: For each point x_p , score is computed as
1163

1164
$$\text{Score}(x_p) = \begin{cases} |C_i| \times \min_{j \in LC} \text{dist}(x_p, C_j), & \text{if } x_p \in C_i, C_i \in SC \text{ and } C_j \in LC \\ |C_i| \times \text{dist}(x_p, C_i), & \text{if } x_p \in C_i, \text{ and } C_i \in LC \end{cases}$$

1165

1166 where C_i denotes i^{th} cluster and $|C_i|$ denotes the number of points in each cluster. Also, $|LC|$ is the
1167 number of instances that belong to large clusters and $|SC|$ is the number of instances that belong to
1168 small clusters, formally, $|SC| = n - |LC|$.
11691170 If $p \in SC$: As L large clusters exist, calculating distances to L cluster centroid costs $L(3d - 1)$
1171 and comparing costs $L - 1$. Therefore, each point in small cluster need $3dL$ FLOPs, including
1172 multiplication operation of cluster size.
11731174 If $p \in LC$: Calculating distances to their own centroid costs $3d - 1$ and multiplication costs 1
1175 FLOPs. Therefore, each point in large cluster need $3d$ FLOPs.
11761177 Total scoring FLOPs for all n samples is
1178

1179
$$F_{\text{score}} = 3d(|SC| \cdot L + |LC|)$$

1180
$$= 3d((n - |LC|)L + |LC|).$$

1181

1182 Combining k-means and scoring,
1183

1184
$$F^{\text{train/infer}} = nI(3Cd + d - 1) + 3d((n - |LC|)L + |LC|).$$

1185

1186 C.6 HBOS
11871188 **Model-specific notation** b : number of bins per feature.
11891190 **Training FLOPs** The training cost consists of histogram construction. Each sample-feature value
1191 is assigned to a bin with one subtraction and one division, giving $2n_{\text{trd}}$ FLOPs in total. Converting
1192 counts to densities, computing bin widths, and performing the normalization check together require
1193 $5bd$ FLOPs.
11941195 Thus,
1196

1197
$$F^{\text{train}} = 2n_{\text{trd}} + 5bd.$$

1198

1188 **Inference FLOPs** At inference time, the score of each sample is computed based on the histograms. Each feature requires the computation of $\log_2(\text{hist} + \alpha)$, which costs 2 FLOPs per bin and
 1189 yields $2bd$ FLOPs in total. For every sample and feature, assigning the score with boundary checks
 1190 adds about 3 FLOPs, giving $3n_{\text{inf}}d$ FLOPs.
 1191

1192 Therefore,

$$F^{\text{infer}} = 3n_{\text{inf}}d + 2bd.$$

1195 C.7 LOMA

1197 **Model-specific notation** b : number of bins, c : number of random projections.

1199 **Training FLOPs** The training cost consists of sparse random projections and histogram construction,
 1200 covering both the computation of projected values and the assignment of sample to bins for
 1201 density estimation.

1203 (i) *Sparse projection*: Each projection vector has \sqrt{d} nonzero entries. Computing one projection
 1204 value $z_{ij} = x_j^\top w_i$ requires $2\sqrt{d} - 1$ FLOPs. With n_{tr} training samples and c projections, the cost is
 1205

$$F_{\text{proj}} = n_{\text{tr}}c(2\sqrt{d} - 1).$$

1207 (ii) *Histogram construction*: Each projected value must be assigned to a histogram bin. Using binary
 1208 search over the b bin edges requires $\log_2 b$ comparisons per assignment. The cost is therefore
 1209

$$F_{\text{bin}} = n_{\text{tr}}c \log_2 b.$$

1211 Summing up, the training FLOPs are

$$\begin{aligned} F^{\text{train}} &= F_{\text{proj}} + F_{\text{bin}} \\ &= n_{\text{tr}}c(2\sqrt{d} + \log_2 b - 1). \end{aligned}$$

1216 **Inference FLOPs** During inference, each sample is projected onto the c , its bin is determined, and
 1217 the corresponding density values are used to compute the anomaly score.

1219 (i) *Sparse projection*: Each projection requires $2\sqrt{d} - 1$ FLOPs, and the total projection cost over
 1220 all histogram is

$$f_{\text{proj}} = c(2\sqrt{d} - 1).$$

1222 (ii) *Bin lookup and score computation*: For each projection, locating the appropriate bin via binary
 1223 search and computing log-density with accumulation require $c \log_2 b + 2$ FLOPs. Over all projections
 1224 this becomes

$$f_{\text{binscore}} = c(\log_2 b + 2).$$

1227 Therefore, including the final division for averaging across projections, the inference cost is

$$\begin{aligned} f^{\text{infer}} &= f_{\text{proj}} + f_{\text{binscore}} \\ &= 2c\sqrt{d} + c \log_2 b + c + 1, \\ F^{\text{infer}} &= n_{\text{inf}}(2c\sqrt{d} + c \log_2 b + c + 1). \end{aligned}$$

1234 C.8 ISOLATION FOREST

1236 **Model-specific notation** T : number of trees, s : max samples per tree, γ : Euler-Mascheroni con-
 1237 stant ($\gamma \approx 0.5772$).

1239 **Training FLOPs** Each tree is grown on a random subsample of size s . At each internal node, we
 1240 pick a random feature, sample a split value within the feature's range, and partition the instances by
 1241 comparison. Let n_l denote the expected number of samples at a node in level l . At a level l node,
 1242 computing feature's range costs n_l , and partitioning the instances costs n_l , since the tree is binary.

Also, we approximate the tree height as $h \approx \log_2 s$, the number of nodes in level l as 2^l , and number of samples processed in a level l node as $n_l \approx \frac{s}{2^l}$. Therefore, computation at each node in level l costs $2 \times \frac{s}{2^l}$. The total cost for a single tree is

$$\sum_{l=0}^{h-1} \left(2 \times \frac{s}{2^l} \times 2^l\right) = 2sh = 2s \log_2 s.$$

For T trees the training FLOPs are approximated as

$$F^{\text{train}} = T(2s \log_2 s).$$

Inference FLOPs A sample is routed from the root to a leaf in every tree. The expected path length $c(s)$ for subsample size s is presented by authors of Isolation Forest (Liu et al., 2008).

$$c(s) = 2H_{s-1} - \frac{2(s-1)}{s} \approx 2(\ln(s-1) + \gamma) - 2 + \frac{2}{s}$$

Each step down the tree costs one comparison, hence costs per sample across T trees are $T \cdot c(s)$.

Therefore,

$$f_{\text{step_down}} = T \cdot c(s) = T \left[2\{\ln(s-1) + \gamma\} - 2\left(1 - \frac{1}{s}\right) \right]$$

Anomaly score of the model is calculated by $2^{-\frac{\mathbb{E}(h(x))}{c(s)}}$.

Score calculation is performed by aggregating path lengths across the T trees. This requires $T - 1$ additions, followed by a normalization step that introduces two more scalar operations, division and exponentiation. We thus fold these into an overall $T + 2$ overhead.

Therefore, per sample cost is

$$f^{\text{infer}} = T \cdot c(s) + (T + 2) = T \left[2\ln(s-1) + 2\gamma - 2 + \frac{2}{s} \right] + (T + 2).$$

For n_{inf} instances,

$$\begin{aligned} F^{\text{infer}} &= n_{\text{inf}}(T \cdot c(s) + (T + 2)) \\ &= n_{\text{inf}} \left(T \left[2\ln(s-1) + 2\gamma - 2 + \frac{2}{s} \right] + (T + 2) \right). \end{aligned}$$

C.9 HS-TREE

Model-specific notation T : number of trees, h : maximum depth of tree, ψ : reference window size.

Training FLOPs Let $|Node| = \sum_{l=0}^h 2^l = 2^{h+1} - 1$ be the number of nodes in a full binary tree of height h . Each tree is built by updating simple per-node statistics and routing the ψ reference samples level by level.

(i) *Per-node statistic updates*: For every node we find work range, yielding a constant cost of about 5 FLOPs per node.

Therefore, with a single tree,

$$F_{\text{stat/tree}} = 5(2^{h+1} - 1).$$

(ii) *Routing the ψ reference samples*: At level l there are 2^l nodes and, on average, each processes $\psi/2^l$ samples. With one comparison per routed sample, the cost per level is ψ .

Across levels,

$$F_{\text{routing/tree}} = \sum_{l=0}^h \psi = \psi(h+1).$$

Total FLOPs calculated is

$$\begin{aligned} F^{\text{train}} &= T(F_{\text{stat/tree}} + F_{\text{routing/tree}}) \\ &= T(5(2^{h+1} - 1) + \psi(h+1)). \end{aligned}$$

1296 **Inference FLOPs** For each sample we route to level h and calculate score, and update leaf mass
 1297 statistics.

1298 (i) *Path routing and scoring*: Per sample, one comparison per level is conducted with h levels and
 1299 scoring is conducted with the cumulative sum of $\text{Node.r} \times 2^{\text{Node.k}}$ which costs 3 FLOPs.

1300 Thus,

$$f_{\text{routing/tree}} = h + 3.$$

1301 (ii) *On-path mass updates*: Along the visited path, we update mass with 4 operations including two
 1302 comparison for lower and upper bound, addition of count, and depth comparison.

1303 As each point pass $h + 1$ nodes,

$$f_{\text{update/tree}} = 4(h + 1).$$

1304 Therefore,

$$f^{\text{infer}} = T(f_{\text{routing/tree}} + f_{\text{update/tree}})$$

$$= T(5h + 7),$$

$$F^{\text{infer}} = n_{\text{inf}} T(5h + 7).$$

1314 D DETAILED EXPERIMENT SETTINGS

1315 **Hyperparameter Tuning** For both traditional and deep learning models, we defined hyperparameter
 1316 search spaces based on ranges commonly adopted in prior benchmark studies. For each model,
 1317 the parameters of each algorithm (e.g. number of estimators, number of bins, neighborhood size,
 1318 clustering parameters, latent dimensions, dropout rates, training epochs) were specified as candi-
 1319 date sets. These search spaces were predetermined and systematically explored through grid search
 1320 across all datasets. For each dataset and model pair, all hyperparameter combinations were eval-
 1321 uated, and the configuration yielding the highest AUROC was selected as the final setting. This
 1322 procedure ensured that every model was tuned in a consistent and performance-oriented manner
 1323 while remaining faithful to the parameter ranges established in the literature.

1324

1325 E ADDITIONAL EXPERIMENT RESULTS

1326 E.1 ADDITIONAL RESULTS OF SECTION 4.1

1327 To complement the AUROC-based summary in Table 3, we further provide an additional table that
 1328 reorganizes the results from the perspective of computational efficiency, which is presented in Ta-
 1329 ble 6. Specifically, for each dataset, we first report the top five models ranked in ascending order of
 1330 total FLOPs, allowing a direct comparison of which approaches remain feasible under strict resource
 1331 budgets.

1332 Following these efficiency-focused tables, we provide the complete results for each dataset. The
 1333 following tables report AUROC, AUPRC, VUS-ROC, VUS-PR, and FLOPs across all evaluated
 1334 models. We observe that GFLOPs for deep learning models are generally much larger than those of
 1335 traditional models, despite yielding comparable accuracy. For clarity, the best **performance score** in
 1336 each table is highlighted in **bold**, while the second-best score is underlined.

1337

1338

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350
 1351 **Table 6: Top five models for each dataset ranked in ascending order of total FLOPs. All computa-**
 1352 **tional costs are reported in gigaFLOPs (GFLOPs). AUROC and AUPRC scores are reported as the**
 1353 **mean over five runs with different random seeds.**

Dataset	Model	Type	GFLOPs ↓			AUROC ↑	AUPRC ↑
			Train	Inference	Full-training		
PSM	HBOS	Statistical	0.01	0.01	—	0.73 ± 0.00	0.50 ± 0.00
	Isolation Forest	One-class	< 0.01	0.07	—	0.70 ± 0.02	0.47 ± 0.03
	HS-Tree	One-class	< 0.01	0.10	—	0.54 ± 0.02	0.32 ± 0.02
	LODA	Statistical	0.13	0.10	—	0.65 ± 0.03	0.45 ± 0.03
	Hotelling	Statistical	0.17	0.11	—	0.77 ± 0.00	0.49 ± 0.00
MSL	HBOS	Statistical	0.01	0.01	—	0.62 ± 0.00	0.16 ± 0.00
	HS-Tree	One-class	< 0.01	0.08	—	0.64 ± 0.03	0.14 ± 0.03
	Isolation Forest	One-class	< 0.01	0.14	—	0.62 ± 0.01	0.15 ± 0.01
	LODA	Statistical	0.15	0.21	—	0.47 ± 0.02	0.11 ± 0.02
	Hotelling	Statistical	0.36	0.45	—	0.53 ± 0.00	0.13 ± 0.00
SMAP	HBOS	Statistical	0.01	0.03	—	0.61 ± 0.00	0.15 ± 0.00
	LODA	Statistical	0.08	0.31	—	0.45 ± 0.09	0.12 ± 0.02
	Isolation Forest	One-class	< 0.01	0.48	—	0.64 ± 0.01	0.16 ± 0.00
	PCA	Reconstruction	0.18	0.36	—	0.41 ± 0.00	0.11 ± 0.00
	HS-Tree	One-class	0.01	0.70	—	0.45 ± 0.01	0.11 ± 0.00
SMD	HBOS	Statistical	0.05	0.08	—	0.63 ± 0.00	0.14 ± 0.00
	HS-Tree	One-class	< 0.01	0.81	—	0.64 ± 0.02	0.07 ± 0.01
	Isolation Forest	One-class	< 0.01	0.90	—	0.68 ± 0.01	0.16 ± 0.01
	LODA	Statistical	0.55	0.63	—	0.62 ± 0.02	0.11 ± 0.01
	PCA	Reconstruction	2.1	2.09	—	0.65 ± 0.00	0.11 ± 0.00
SWaT	HBOS	Statistical	0.05	0.07	—	0.85 ± 0.00	0.75 ± 0.00
	Isolation Forest	One-class	< 0.01	0.36	—	0.83 ± 0.00	0.73 ± 0.01
	HS-Tree	One-class	0.01	0.74	—	0.37 ± 0.08	0.11 ± 0.01
	LODA	Statistical	1.39	1.39	—	0.82 ± 0.02	0.73 ± 0.02
	Hotelling	Statistical	2.64	2.39	—	0.65 ± 0.00	0.17 ± 0.00
WADI	HS-Tree	One-class	< 0.01	0.20	—	0.63 ± 0.05	0.09 ± 0.02
	HBOS	Statistical	0.19	0.06	—	0.74 ± 0.00	0.18 ± 0.00
	Isolation Forest	One-class	< 0.01	0.33	—	0.74 ± 0.02	0.18 ± 0.01
	LODA	Statistical	1.46	0.34	—	0.72 ± 0.04	0.27 ± 0.05
	DeepSVDD	One-class	19.5	1.43	4874.05	0.43 ± 0.03	0.05 ± 0.00

1387
 1388 **Table 7: All results on the PSM dataset with performance metrics and GFLOPs.**
 1389

Model	Type	GFLOPs ↓			AUROC ↑	AUPRC ↑	VUS-ROC ↑	VUS-PR ↑
		Train	Inference	Full-training				
Hotelling	Statistical	0.17	0.11	—	0.77 ± 0.00	0.49 ± 0.00	0.71 ± 0.00	0.49 ± 0.00
ABOD	Statistical	926.33	421.87	—	0.75 ± 0.00	0.43 ± 0.00	0.66 ± 0.00	0.45 ± 0.00
LOF	One-class	917.76	416.07	—	0.73 ± 0.00	0.42 ± 0.00	0.63 ± 0.00	0.43 ± 0.00
CBLOF	One-class	6.56	9.91	—	0.70 ± 0.02	0.45 ± 0.01	0.64 ± 0.02	0.44 ± 0.01
PCA	Reconstruction	0.17	0.16	—	0.65 ± 0.00	0.47 ± 0.00	0.59 ± 0.00	0.43 ± 0.00
HBOS	Statistical	0.01	0.01	—	0.73 ± 0.00	0.50 ± 0.00	0.68 ± 0.00	0.48 ± 0.00
LODA	Statistical	0.13	0.10	—	0.65 ± 0.03	0.45 ± 0.03	0.58 ± 0.03	0.41 ± 0.02
Isolation Forest	One-class	< 0.01	0.07	—	0.70 ± 0.02	0.47 ± 0.03	0.67 ± 0.02	0.46 ± 0.02
HS-Tree	One-class	< 0.01	0.10	—	0.54 ± 0.02	0.32 ± 0.02	0.55 ± 0.02	0.34 ± 0.02
DAGMM	Statistical	5.31	1.20	530.62	0.50 ± 0.03	0.28 ± 0.01	0.47 ± 0.03	0.30 ± 0.01
DeepSVDD	One-class	1.57	0.35	156.61	0.69 ± 0.01	0.37 ± 0.01	0.57 ± 0.01	0.40 ± 0.01
LSTM-AE	Reconstruction	12.23	2.76	1223.50	0.76 ± 0.01	0.50 ± 0.04	0.70 ± 0.01	0.48 ± 0.01
LSTM-VAE	Reconstruction	547.41	123.50	136852.56	0.61 ± 0.06	0.42 ± 0.07	0.57 ± 0.06	0.40 ± 0.06
USAD	Reconstruction	2.63	0.59	657.91	0.52 ± 0.01	0.34 ± 0.01	0.44 ± 0.01	0.32 ± 0.00
OmniAnomaly	Reconstruction	10.15	2.29	203.05	0.61 ± 0.00	0.44 ± 0.00	0.58 ± 0.01	0.41 ± 0.00
LUAD	Reconstruction	4956.20	1118.16	148685.87	0.67 ± 0.01	0.48 ± 0.01	0.62 ± 0.01	0.44 ± 0.01
Anomaly Transformer	Reconstruction	4965.57	1120.27	148967.04	0.52 ± 0.00	0.32 ± 0.01	0.56 ± 0.00	0.33 ± 0.00
TimesNet	Reconstruction	1380.51	311.45	27610.22	0.60 ± 0.00	0.40 ± 0.00	0.65 ± 0.00	0.45 ± 0.00

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

1427

1428

1429

1430

1431

1432

1433

1434

1435

1436

1437

1438

1439

1440

1441

1442

1443

1444

1445

1446

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

Table 8: All results on the MSL dataset with **performance metrics** and GFLOPs.

Model	Type	GFLOPs ↓			AUROC ↑	AUPRC ↑	VUS-ROC ↑	VUS-PR ↑
		Train	Inference	Full-training				
Hotelling	Statistical	0.36	0.45	—	0.53 ± 0.00	0.13 ± 0.00	0.58 ± 0.00	0.15 ± 0.00
ABOD	Statistical	334.71	536.74	—	0.63 ± 0.00	0.17 ± 0.00	0.63 ± 0.00	0.18 ± 0.00
LOF	One-class	334.41	536.36	—	0.56 ± 0.00	0.12 ± 0.00	0.59 ± 0.00	0.14 ± 0.00
CBLOF	One-class	0.89	2.60	—	0.65 ± 0.01	0.20 ± 0.01	0.67 ± 0.01	0.20 ± 0.01
PCA	Reconstruction	0.36	0.64	—	0.53 ± 0.00	0.14 ± 0.00	0.59 ± 0.00	0.15 ± 0.00
HBOS	Statistical	0.01	0.01	—	0.62 ± 0.00	0.16 ± 0.00	0.66 ± 0.00	0.18 ± 0.00
LODA	Statistical	0.15	0.21	—	0.47 ± 0.02	0.11 ± 0.02	0.52 ± 0.03	0.12 ± 0.01
Isolation Forest	One-class	< 0.01	0.14	—	0.62 ± 0.01	0.15 ± 0.01	0.66 ± 0.01	0.16 ± 0.01
HS-Tree	One-class	< 0.01	0.08	—	0.64 ± 0.03	0.14 ± 0.03	0.68 ± 0.02	0.17 ± 0.02
DAGMM	Statistical	2.55	1.08	127.74	0.50 ± 0.02	0.13 ± 0.01	0.52 ± 0.01	0.12 ± 0.01
DeepSVDD	One-class	2.54	1.07	253.56	0.48 ± 0.03	0.13 ± 0.00	0.48 ± 0.02	0.12 ± 0.01
LSTM-AE	Reconstruction	22.77	9.60	455.49	0.56 ± 0.00	0.15 ± 0.00	0.61 ± 0.00	0.17 ± 0.00
LSTM-VAE	Reconstruction	21.23	8.95	5307.31	0.53 ± 0.00	0.13 ± 0.01	0.59 ± 0.00	0.15 ± 0.00
USAD	Reconstruction	2.07	0.87	206.77	0.44 ± 0.00	0.13 ± 0.00	0.47 ± 0.00	0.12 ± 0.00
OmniAnomaly	Reconstruction	6.25	2.63	124.94	0.53 ± 0.00	0.14 ± 0.00	0.59 ± 0.00	0.14 ± 0.00
LUAD	Reconstruction	2362.72	995.71	23627.17	0.50 ± 0.00	0.13 ± 0.00	0.56 ± 0.00	0.14 ± 0.00
Anomaly Transformer	Reconstruction	576.01	242.75	17280.39	0.51 ± 0.00	0.12 ± 0.00	0.54 ± 0.01	0.13 ± 0.00
TimesNet	Reconstruction	21851.23	9208.69	437024.60	0.64 ± 0.03	0.16 ± 0.01	0.68 ± 0.03	0.19 ± 0.01

Table 9: All results on the SMAP dataset with **performance metrics** and GFLOPs.

Model	Type	GFLOPs ↓			AUROC ↑	AUPRC ↑	VUS-ROC ↑	VUS-PR ↑
		Train	Inference	Full-training				
Hotelling	Statistical	0.18	0.56	—	0.42 ± 0.00	0.11 ± 0.00	0.42 ± 0.00	0.11 ± 0.00
ABOD	Statistical	998.15	10281.98	—	0.64 ± 0.00	0.17 ± 0.00	0.58 ± 0.00	0.17 ± 0.00
LOF	One-class	996.78	10277.64	—	0.62 ± 0.00	0.17 ± 0.00	0.56 ± 0.00	0.17 ± 0.00
CBLOF	One-class	2.48	10.76	—	0.62 ± 0.01	0.16 ± 0.00	0.58 ± 0.01	0.16 ± 0.00
PCA	Reconstruction	0.18	0.36	—	0.41 ± 0.00	0.11 ± 0.00	0.42 ± 0.00	0.11 ± 0.00
HBOS	Statistical	0.01	0.03	—	0.61 ± 0.00	0.15 ± 0.00	0.57 ± 0.00	0.16 ± 0.00
LODA	Statistical	0.08	0.31	—	0.45 ± 0.09	0.12 ± 0.02	0.45 ± 0.08	0.12 ± 0.02
Isolation Forest	One-class	< 0.01	0.48	—	0.64 ± 0.01	0.16 ± 0.00	0.59 ± 0.01	0.17 ± 0.00
HS-Tree	One-class	0.01	0.70	—	0.45 ± 0.01	0.11 ± 0.00	0.43 ± 0.01	0.12 ± 0.00
DAGMM	Statistical	5.53	5.83	552.69	0.51 ± 0.01	0.13 ± 0.00	0.51 ± 0.00	0.14 ± 0.00
DeepSVDD	One-class	2.73	2.88	272.94	0.60 ± 0.00	0.15 ± 0.00	0.55 ± 0.00	0.17 ± 0.00
LSTM-AE	Reconstruction	12.74	13.44	637.20	0.49 ± 0.03	0.12 ± 0.01	0.50 ± 0.03	0.13 ± 0.01
LSTM-VAE	Reconstruction	39.87	42.04	9966.77	0.44 ± 0.02	0.11 ± 0.00	0.45 ± 0.02	0.11 ± 0.00
USAD	Reconstruction	5.68	5.99	1419.32	0.41 ± 0.01	0.10 ± 0.00	0.41 ± 0.01	0.11 ± 0.00
OmniAnomaly	Reconstruction	34.43	36.30	688.56	0.47 ± 0.00	0.12 ± 0.00	0.49 ± 0.00	0.13 ± 0.00
LUAD	Reconstruction	5162.37	5443.28	51623.73	0.50 ± 0.01	0.12 ± 0.00	0.51 ± 0.01	0.13 ± 0.00
Anomaly Transformer	Reconstruction	254.83	268.70	7644.90	0.52 ± 0.02	0.13 ± 0.01	0.54 ± 0.02	0.14 ± 0.01
TimesNet	Reconstruction	1428.44	1506.16	28568.71	0.47 ± 0.00	0.11 ± 0.00	0.48 ± 0.00	0.12 ± 0.00

Table 10: All results on the SMD dataset with **performance metrics** and GFLOPs.

Model	Type	GFLOPs ↓			AUROC ↑	AUPRC ↑	VUS-ROC ↑	VUS-PR ↑
		Train	Inference	Full-training				
Hotelling	Statistical	2.10	2.10	—	0.73 ± 0.00	0.16 ± 0.00	0.73 ± 0.00	0.14 ± 0.00
ABOD	Statistical	38426.94	38428.58	—	0.71 ± 0.00	0.10 ± 0.00	0.69 ± 0.00	0.10 ± 0.00
LOF	One-class	38357.60	38359.24	—	0.65 ± 0.00	0.07 ± 0.00	0.65 ± 0.00	0.08 ± 0.00
CBLOF	One-class	36.30	34.65	—	0.72 ± 0.01	0.16 ± 0.01	0.68 ± 0.01	0.12 ± 0.01
PCA	Reconstruction	2.10	2.09	—	0.65 ± 0.00	0.11 ± 0.00	0.67 ± 0.00	0.10 ± 0.00
HBOS	Statistical	0.05	0.08	—	0.63 ± 0.00	0.14 ± 0.00	0.62 ± 0.00	0.10 ± 0.00
LODA	Statistical	0.55	0.63	—	0.62 ± 0.02	0.11 ± 0.01	0.62 ± 0.02	0.09 ± 0.01
Isolation Forest	One-class	< 0.01	0.90	—	0.68 ± 0.01	0.16 ± 0.01	0.66 ± 0.00	0.11 ± 0.00
HS-Tree	One-class	< 0.01	0.81	—	0.64 ± 0.02	0.07 ± 0.01	0.64 ± 0.02	0.08 ± 0.01
DAGMM	Statistical	21.79	7.26	2178.62	0.53 ± 0.05	0.04 ± 0.01	0.53 ± 0.02	0.06 ± 0.01
DeepSVDD	One-class	5.34	1.78	1336.00	0.61 ± 0.02	0.11 ± 0.02	0.54 ± 0.01	0.08 ± 0.00
LSTM-AE	Reconstruction	82.70	27.57	8269.64	0.77 ± 0.01	0.18 ± 0.01	0.75 ± 0.00	0.15 ± 0.00
LSTM-VAE	Reconstruction	791.06	263.69	197764.01	0.63 ± 0.01	0.10 ± 0.00	0.62 ± 0.01	0.09 ± 0.00
USAD	Reconstruction	37.57	12.52	9392.39	0.39 ± 0.00	0.03 ± 0.00	0.41 ± 0.00	0.05 ± 0.00
OmniAnomaly	Reconstruction	198.12	66.04	3962.46	0.69 ± 0.02	0.11 ± 0.02	0.66 ± 0.02	0.11 ± 0.01
LUAD	Reconstruction	27766.94	9255.84	833008.11	0.67 ± 0.00	0.11 ± 0.00	0.67 ± 0.01	0.10 ± 0.00
Anomaly Transformer	Reconstruction	3545.62	1181.90	106368.46	0.52 ± 0.01	0.07 ± 0.00	0.63 ± 0.01	0.08 ± 0.00
TimesNet	Reconstruction	25325.13	8441.89	506502.64	0.77 ± 0.00	0.17 ± 0.00	0.79 ± 0.00	0.15 ± 0.00

1458
1459

Table 11: All results on the SWaT dataset with performance metrics GFLOPs.

1460
1461

Model	Type	GFLOPs ↓			AUROC ↑	AUPRC ↑	VUS-ROC ↑	VUS-PR ↑
		Train	Inference	Full-training				
Hotelling	Statistical	2.64	2.39	—	0.65 ± 0.00	0.17 ± 0.00	0.59 ± 0.00	0.17 ± 0.00
ABOD	Statistical	23615.66	19345.49	—	0.60 ± 0.00	0.15 ± 0.00	0.55 ± 0.00	0.15 ± 0.00
LOF	One-class	23551.21	19287.13	—	0.71 ± 0.00	0.32 ± 0.00	0.51 ± 0.00	0.21 ± 0.00
CBLOF	One-class	3.95	4.01	—	0.81 ± 0.01	0.70 ± 0.01	0.56 ± 0.01	0.34 ± 0.00
PCA	Reconstruction	2.64	2.42	—	0.82 ± 0.00	0.73 ± 0.00	0.61 ± 0.00	0.43 ± 0.00
HBOS	Statistical	0.05	0.07	—	0.85 ± 0.00	0.75 ± 0.00	0.68 ± 0.00	0.48 ± 0.00
LODA	Statistical	1.39	1.39	—	0.82 ± 0.02	0.73 ± 0.02	0.66 ± 0.04	0.47 ± 0.04
Isolation Forest	One-class	< 0.01	0.36	—	0.83 ± 0.00	0.73 ± 0.01	0.70 ± 0.01	0.48 ± 0.01
HS-Tree	One-class	0.01	0.74	—	0.37 ± 0.08	0.11 ± 0.01	0.37 ± 0.07	0.11 ± 0.01
DAGMM	Statistical	20.24	6.11	2023.86	0.51 ± 0.00	0.12 ± 0.00	0.51 ± 0.01	0.13 ± 0.00
DeepSVDD	One-class	5.11	1.54	1277.45	0.82 ± 0.03	0.72 ± 0.03	0.59 ± 0.10	0.37 ± 0.07
LSTM-AE	Reconstruction	187.15	56.50	9357.33	0.76 ± 0.01	0.28 ± 0.03	0.61 ± 0.01	0.25 ± 0.01
LSTM-VAE	Reconstruction	584.52	176.45	146130.74	0.76 ± 0.11	0.59 ± 0.26	0.58 ± 0.08	0.37 ± 0.13
USAD	Reconstruction	32.79	9.90	3278.73	0.81 ± 0.00	0.69 ± 0.01	0.54 ± 0.00	0.33 ± 0.01
OmniAnomaly	Reconstruction	51.30	15.49	1026.05	0.83 ± 0.00	0.73 ± 0.00	0.78 ± 0.01	0.56 ± 0.01
LUAD	Reconstruction	5653.12	1706.55	169593.69	0.75 ± 0.01	0.25 ± 0.01	0.66 ± 0.03	0.26 ± 0.02
Anomaly Transformer	Reconstruction	19166.41	5785.92	574992.39	0.44 ± 0.07	0.21 ± 0.04	0.43 ± 0.07	0.15 ± 0.01
TimesNet	Reconstruction	280014.48	84530.22	5600289.64	0.28 ± 0.00	0.09 ± 0.00	0.30 ± 0.00	0.09 ± 0.00

1473

1474
1475

Table 12: All results on the WADI dataset with performance metrics GFLOPs.

1476
1477

Model	Type	GFLOPs ↓			AUROC ↑	AUPRC ↑	VUS-ROC ↑	VUS-PR ↑
		Train	Inference	Full-training				
Hotelling	Statistical	23.93	5.27	—	0.53 ± 0.00	0.06 ± 0.00	0.53 ± 0.00	0.06 ± 0.00
ABOD	Statistical	125696.96	6047.49	—	0.49 ± 0.00	0.12 ± 0.00	0.44 ± 0.00	0.07 ± 0.00
LOF	One-class	125611.74	6028.72	—	0.54 ± 0.00	0.09 ± 0.00	0.48 ± 0.00	0.08 ± 0.00
CBLOF	One-class	97.02	1.66	—	0.53 ± 0.01	0.21 ± 0.00	0.46 ± 0.01	0.11 ± 0.00
PCA	Reconstruction	23.93	3.18	—	0.50 ± 0.00	0.05 ± 0.00	0.46 ± 0.00	0.06 ± 0.00
HBOS	Statistical	0.19	0.06	—	0.74 ± 0.00	0.18 ± 0.00	0.67 ± 0.00	0.17 ± 0.00
LODA	Statistical	1.46	0.34	—	0.72 ± 0.04	0.27 ± 0.05	0.68 ± 0.04	0.22 ± 0.04
Isolation Forest	One-class	< 0.01	0.33	—	0.74 ± 0.02	0.18 ± 0.01	0.67 ± 0.01	0.16 ± 0.01
HS-Tree	One-class	< 0.01	0.20	—	0.63 ± 0.05	0.09 ± 0.02	0.62 ± 0.06	0.10 ± 0.02
DAGMM	Statistical	150.17	11.03	15017.03	0.45 ± 0.05	0.05 ± 0.01	0.40 ± 0.06	0.05 ± 0.01
DeepSVDD	One-class	19.50	1.43	4874.05	0.43 ± 0.03	0.05 ± 0.00	0.37 ± 0.04	0.05 ± 0.00
LSTM-AE	Reconstruction	490.76	36.03	49075.61	0.54 ± 0.00	0.21 ± 0.01	0.49 ± 0.00	0.11 ± 0.01
LSTM-VAE	Reconstruction	1031.52	75.73	257880.45	0.54 ± 0.01	0.19 ± 0.03	0.49 ± 0.01	0.12 ± 0.01
USAD	Reconstruction	146.25	10.74	36562.76	0.45 ± 0.01	0.20 ± 0.00	0.40 ± 0.01	0.10 ± 0.00
OmniAnomaly	Reconstruction	358.51	26.32	7170.23	0.58 ± 0.00	0.24 ± 0.00	0.51 ± 0.00	0.14 ± 0.00
LUAD	Reconstruction	36349.70	2668.78	1090490.86	0.54 ± 0.00	0.21 ± 0.00	0.49 ± 0.00	0.11 ± 0.00
Anomaly Transformer	Reconstruction	8076.86	593.00	242305.76	0.55 ± 0.03	0.09 ± 0.01	0.56 ± 0.02	0.07 ± 0.00
TimesNet	Reconstruction	24899.00	1828.07	497979.94	0.66 ± 0.01	0.20 ± 0.00	0.67 ± 0.01	0.19 ± 0.00

1489

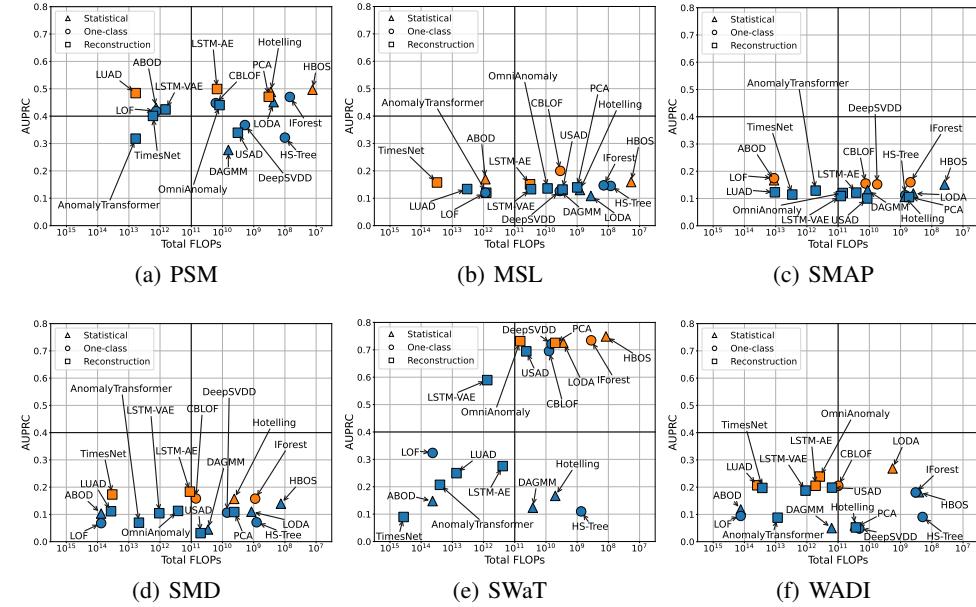
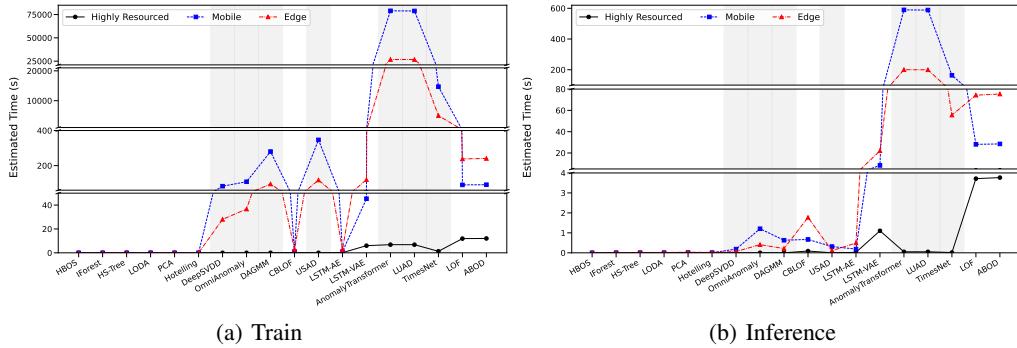
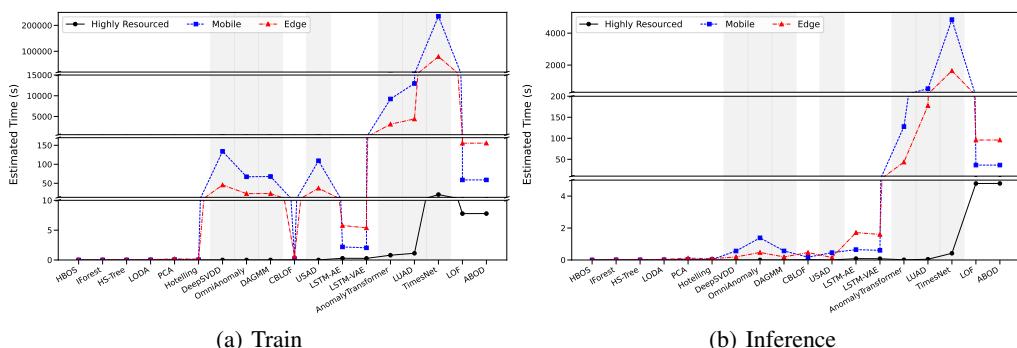
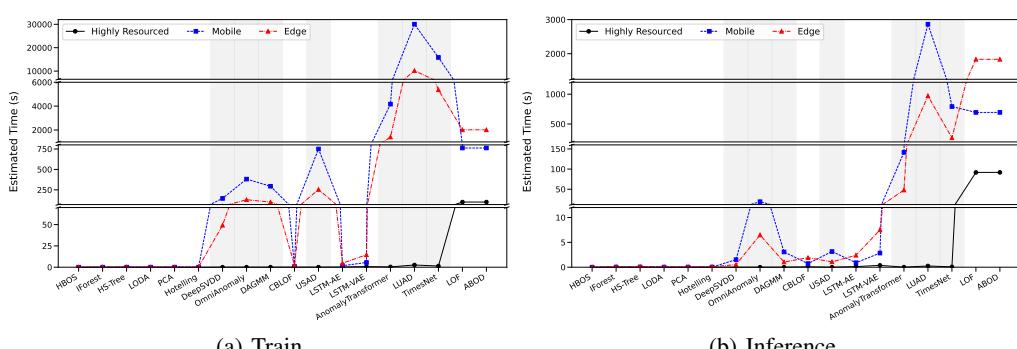
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

Figure 4: AUPRC vs. Total FLOPs (sum of training and inference FLOPs) for each dataset. Orange markers denote the top five models in terms of AUPRC.

1512 E.2 ADDITIONAL RESULTS OF SECTION 4.2
1513

1514 For each dataset, we report the estimated execution time under Highly Resourced, Mobile, and Edge
1515 scenarios. The following figures show dataset specific visualizations that highlight differences in
1516 model feasibility across environments. These results confirm the overall trend that deep learning
1517 models demand substantial resources, while traditional models remain efficient, although the degree
1518 of variation differs across datasets.

1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
Figure 5: Estimated execution time on PSM.1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
Figure 6: Estimated execution time on MSL.1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
Figure 7: Estimated execution time on SMAP.

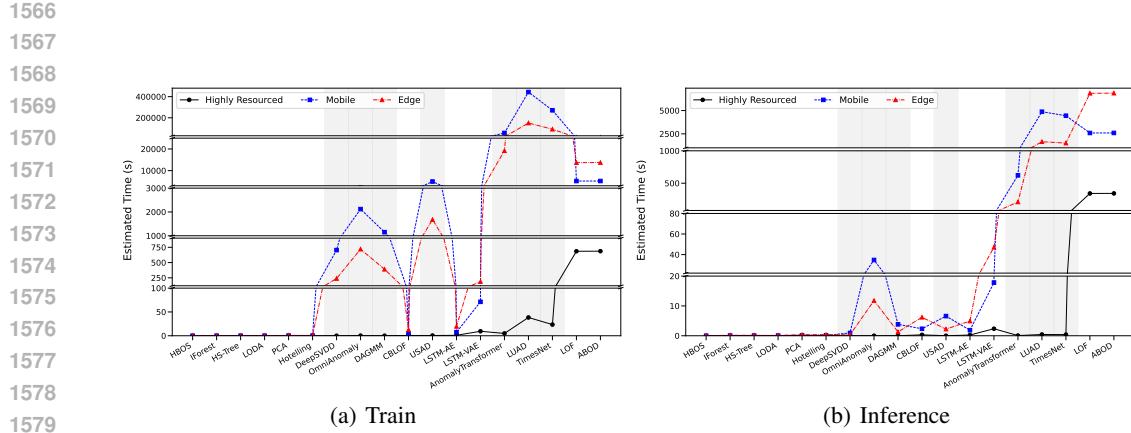


Figure 8: Estimated execution time on SMD.

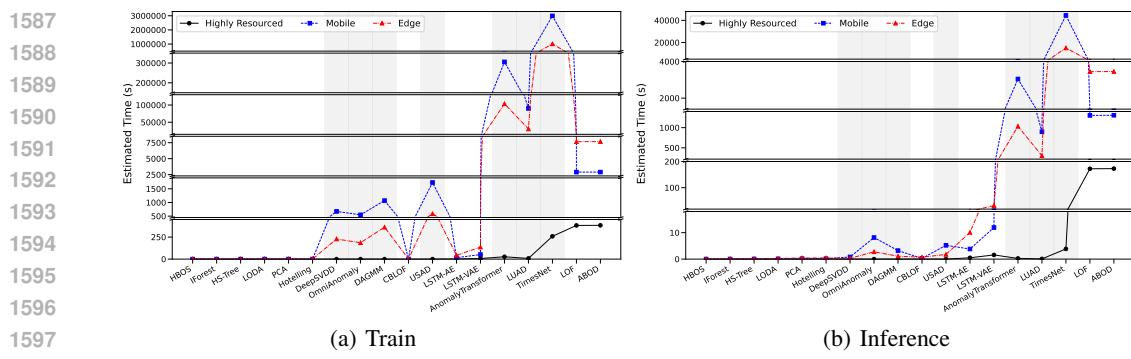


Figure 9: Estimated execution time on SWaT.

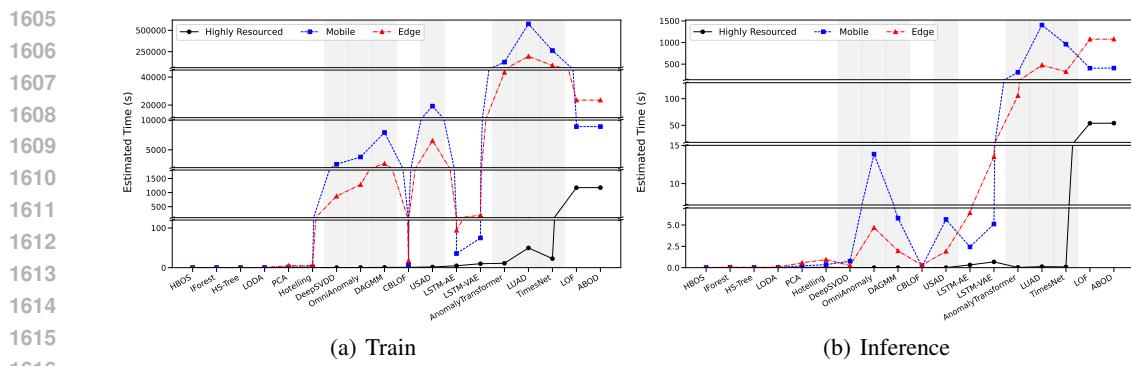
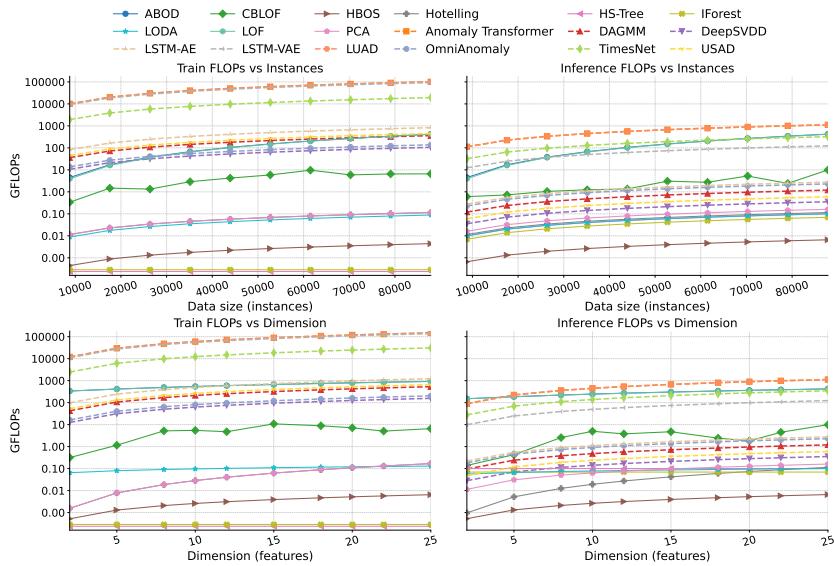
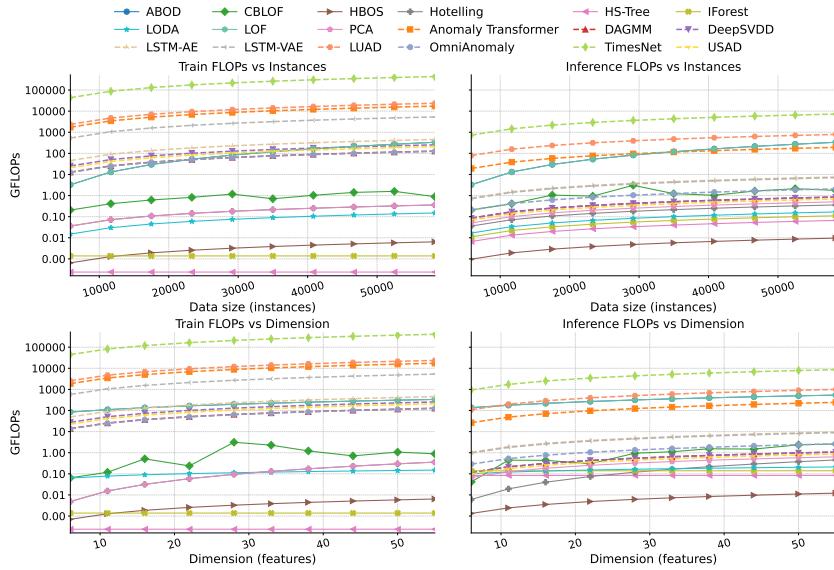


Figure 10: Estimated execution time on WADI.

1620 E.3 ADDITIONAL RESULTS OF SECTION 4.3
1621

1622 We report scalability tests conducted across datasets of varying data size and dimensionality. For
1623 the size scalability experiment, FLOPs were measured while progressively slicing the length of
1624 the dataset, with the maximum test range determined by the smaller of the training and inference
1625 sets to ensure comparability between the two phases. For the dimension scalability experiment,
1626 FLOPs were measured while progressively increasing the feature dimension by sampling features
1627 in ten percent increments of the full dimensionality for each dataset. All models evaluated in this
1628 study are included. The left panel of each figure depicts training FLOPs, while the right panel
1629 presents inference FLOPs, with both y-axes plotted on a logarithmic scale. Overall, the results
1630 reveal consistent scaling patterns across datasets. k -NN based methods exhibit steep growth in
1631 computational cost as data size or dimension increases, while tree-based and projection methods
1632 remain relatively efficient.

1652 Figure 11: Scalability results on PSM.
16531664 Figure 12: Scalability results on MSL.
1665
1666
1667
1668
1669
1670
1671
1672
1673

1674

1675

1676

1677

1678

1679

1680

1681

1682

1683

1684

1685

1686

1687

1688

1689

1690

1691

1692

1693

1694

1695

1696

1697

1698

1699

1700

1701

1702

1703

1704

1705

1706

1707

1708

1709

1710

1711

1712

1713

1714

1715

1716

1717

1718

1719

1720

1721

1722

1723

1724

1725

1726

1727

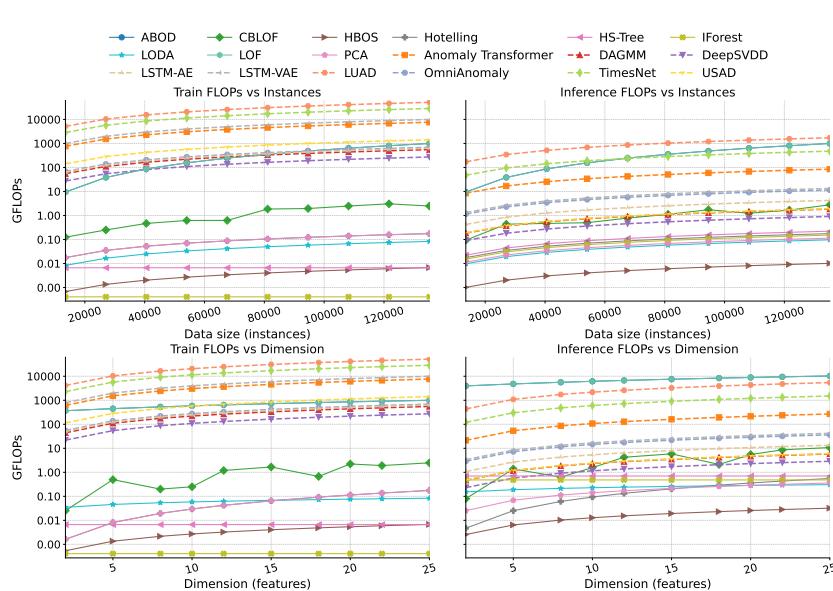


Figure 13: Scalability results on SMAP.

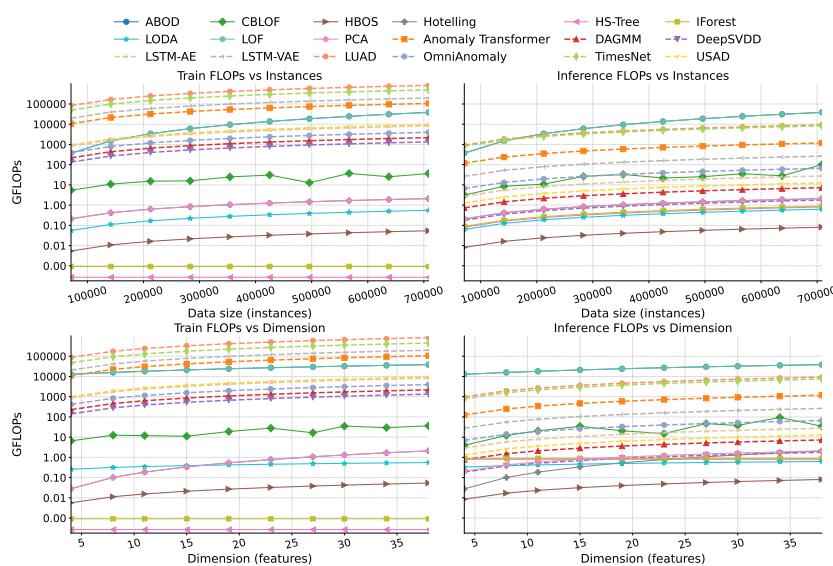


Figure 14: Scalability results on SMD.

1728

1729

1730

1731

1732

1733

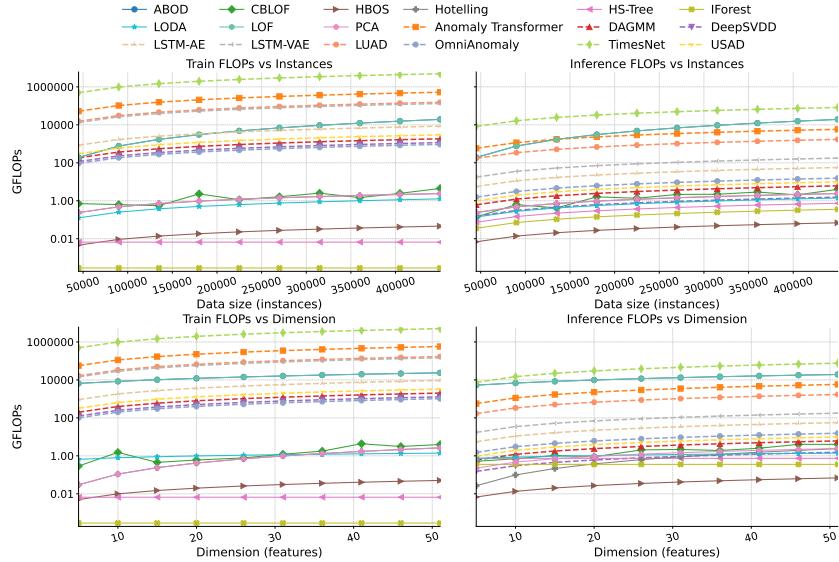


Figure 15: Scalability results on SWaT.

1741

1742

1743

1744

1745

1746

1747

1748

1749

1750

1751

1752

1753

1754

1755

1756

1757

1758

1759

1760

1761

1762

1763

1764

1765

1766

1767

1768

1769

1770

1771

1772

1773

1774

1775

1776

1777

1778

1779

1780

1781

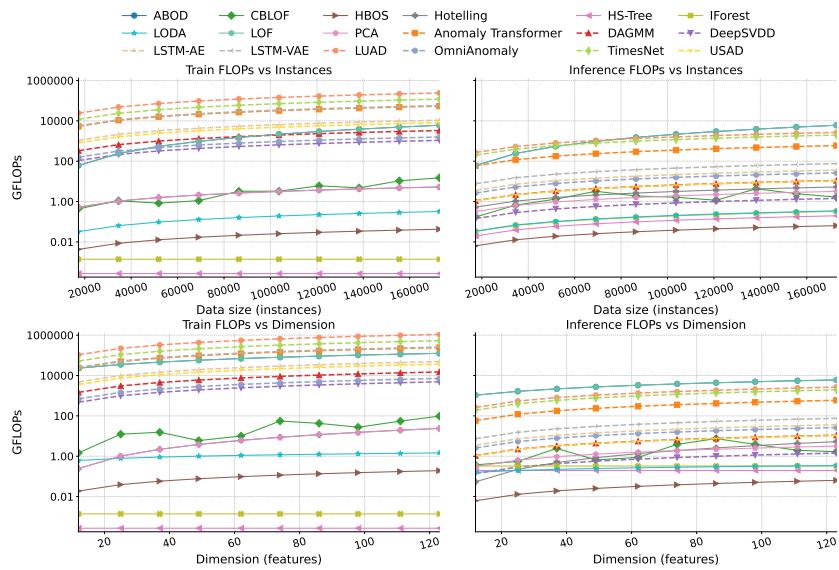


Figure 16: Scalability results on WADI.

1782
1783

E.4 LLM USAGE STATEMENT

1784 The large language model (LLM) was used solely to improve the clarity and readability of the
1785 manuscript. Specifically, they helped polish the writing, refine grammar, and improve phrasing. The
1786 use of LLM was limited to language editing, and LLM did not contribute to the ideation of the
1787 research, experimental design, implementation, analysis, or interpretation of the results.

1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835