Energy Efficient Time Series Anomaly Detection

Rebecca Salles, Benoit Lange, Reza Akbarinia, Florent Masseglia, Esther Pacitti

V. International Conference on Electrical, Computer and Energy Technologies (ICECET 2025)
3-6 July 2025, Paris-France

Energy Efficient Time Series Anomaly Detection

Rebecca Salles
LIRMM, CNRS, Inria, Univ. Montpellier
Montpellier, France
rebeccapsalles@acm.org

Benoit Lange
LIRMM, CNRS, Inria, Univ. Montpellier
Montpellier, France
benoit.lange@inria.fr

Reza Akbarinia
LIRMM, CNRS, Inria, Univ. Montpellier
Montpellier, France
reza.akbarinia@inria.fr

Florent Masseglia
LIRMM, CNRS, Inria, Univ. Montpellier
Montpellier, France
florent.masseglia@inria.fr

Esther Pacitti
LIRMM, CNRS, Inria, Univ. Montpellier
Montpellier, France
esther.pacitti@lirmm.fr

***Abstract*—Anomalous events are commonly observed in real-world temporal data, known as time series. Time series anomaly detection is pervasive for process monitoring in almost every scientific application. The area presents extensive literature and several state-of-the-art methods. Traditionally, choosing a method for a given application is mainly driven by detection accuracy and runtime. However, with the rapid evolution of hardware and connected devices, massive amounts of time series data are produced, and the real-time analysis of such time series brings new demands not only for accurate and scalable solutions, but also for energy consumption management. In this scenario, any improvement in energy efficiency can have a considerable impact on both the environmental footprint and the monetary expenses. However, to the best of our knowledge, there is no existing work on energy efficient time series anomaly detection. This paper fills this gap by addressing for the first time the problem of benchmarking time series anomaly detection methods based on the trade-off between accuracy, runtime, and energy consumption. We introduce a new metric for evaluating relative energy efficiency performance, called saveUp, and provide a novel methodology, inspired by skyline queries, for benchmarking methods based on a more comprehensive set of metrics, including peak power usage and total energy consumption. Experimental results based on large datasets show that our methodology is useful for selecting the methods that provide the best performance with the lowest energy impacts. Moreover, results indicate that speedup and saveUp are not always directly correlated as believed a priori, and sometimes it is best to "take it slow" in favor of green applications.**

***Index Terms*—time series, anomaly detection, energy consumption, efficiency, sustainability**

I. Introduction

Process monitoring in almost every scientific application involves analyzing sequential data that evolve over time, known as time series [1]. This type of data is ubiquitous and used in many domains: environment, healthcare, finance, supply chain, etc. When analyzing most real-world time series data, it is common to observe anomalies, that are specific events that diverge from historical data. An anomaly is characterized by a rare significant change in behavior over a certain time or interval [2]. It may represent, for example, an extreme weather event or a breach of security.

The detection of time series anomalies is a fundamental function in any monitoring application [3]. Consequently, several anomaly detection methods and many papers have been published in this field of research [4]. Because the state-of-the-art is extensive, finding the most suitable method for a specific dataset is not trivial. Currently, method selection is mainly driven by performance metrics: detection accuracy and runtime. Both metrics are useful for evaluating algorithms, but with the rapid evolution of hardware and connected devices, they may not be enough.

In recent years, more and more devices have been developed and deployed every day, producing massive amounts of temporal data at a high rate. This scenario requires anomaly detection methods to be not only accurate, fast, and scalable, but also energy efficient [5]. The efficient use of energy is a pressing issue in the information technology (IT) sector [6] and should not be dismissed by computing applications, including anomaly detection. However, to the best of our knowledge, no existing research focuses on energy efficient time series anomaly detection methods.

As a motivating example, take time series data drawn from financial markets such as Nasdaq[1], with tens of millions of daily transactions. Such time series are produced continuously and need to be managed in real-time (online). In that case, any anomaly detection method that provides an $x\%$ improvement in energy consumption every n second can have a considerable impact over the course of a year in both environmental footprint and monetary savings. With comparable accuracy and/or runtime, energy efficiency should be considered a useful performance evaluation metric, specially for energy-constrained applications.

This paper addresses for the first time the problem of benchmarking time series anomaly detection methods based on the trade-off between not only accuracy and runtime, but also energy consumption. For this, we introduce a new metric, called saveUp, to evaluate relative energy efficiency performance. Furthermore, we provide a novel methodology, inspired by skyline queries, for Accuracy-Time-Energy (ATE) performance benchmarking. It allows the comparison of detection methods based on a more comprehensive set of metrics, including: accuracy, runtime, peak power usage, and total energy consumption.

[1]https://www.nasdaqtrader.com/Trader.aspx?id=DailyMarketSummary

979-8-3315-3559-9/25/$31.00

We experimentally compare some of the most relevant methods for time series anomaly detection [7], [8] based on three synthetic and real multivariate time series datasets containing anomalous observations, ranging up to 10 million time points and more than one hundred variables [8]. The methods were also run using both single and multi-threading for a thorough analysis of different energy consumption scenarios.

The experimental results show that our methodology is effective in selecting the detection method that better balances accuracy-time-energy performance metrics, providing high accuracy and time efficiency while also presenting the lowest energy-consuming impacts. Moreover, the results indicate that runtime speedup and energy saveUp are not always directly correlated, as we believed a priori. In such cases, when harmless, it may be best to wait longer in favor of green applications.

The remainder of this paper is organized as follows. Section II introduces related work on time series anomaly detection and energy monitoring in software development. Section III describes the proposed metric and methodology for the evaluation of the performance of anomaly detection. Section IV and V present experimental settings, results, and a discussion on the trade-off between accuracy-time-energy in anomaly detection. Finally, Section VI concludes.

II. Related Work

A large number of surveys have been published on anomaly detection. Olteanu et al. [4] provide a systematic meta-survey, analyzing the evolution of anomaly detection over the last 20 years. Most surveys focus on univariate (single-variable) data [9]. Schmidl et al. [8] provide a comprehensive survey of 71 univariate and multivariate state-of-the-art methods for time series anomaly detection and experimentally compares them. Zamanzadeh Darban et al. [10] and Li and Jung [11] survey deep learning techniques for anomaly detection in multivariate time series. Only a subset of the surveys is concerned with efficiency analysis [7], [8], [12]. Moreover, these works limit themselves to comparing runtime-related metrics.

Software performance measures are widely covered in the literature. The most common are based on measuring the wall time, measuring the CPU usage, computing the speedup, and memory use [13], [14]. These methods are based on tools that collect system metrics[2]. One performance criterion, however, is not well-developed, namely, the energy consumption. Not many papers are focused on the energy metric. This metric is not widely used because it requires a specific system to measure consumption and peak power. Moreover, it is complex to understand which part of the system consumes energy. Hardware manufacturers create tools to help users measure the consumption of an application, but this metric needs optimization. For CPU, a solution is based on RAPL metrics [15]. In the context of GPU, the hardware requires very accurate power management, and hardware probes are implemented directly over GPU. From an software point of view, some papers try to evaluate the energy consumption of their core app [16] or evaluate the energy consumption by simulating a real workload [17]. Some tools have been released to integrate into any hardware [3]. More recent tools focus on how to monitor energy consumption in containerized applications [5], [18]–[20]. These tools are extended to monitor cloud infrastructures based mainly on Kubernetes [4], but most of them use proxies to evaluate the energy consumption. In data centers, energy consumption of the system is also widely studied [5], [6], [21], but by focusing on the global energy footprint of the data center rather than applications.

Current works on anomaly detection and energy focus on the problem of identifying anomalies in time series data on energy consumption [22]–[24]. For example, Himeur et al. [25] and Copiaco et al. [26] use artificial intelligence and deep learning to detect anomalous energy consumption in buildings. Both Chou and Telaga [27] and Yin et al. [28] focus on the real-time detection of anomalous energy consumption, while Yin et al. [28] provide energy efficiency optimization analysis considering uncertainty.

To the best of our knowledge, no research work addresses the energy efficiency of time series anomaly detection. This paper fills this gap by providing a new relative energy performance metric and a benchmarking methodology to understand and analyze algorithms regarding, at the same time, accuracy, time, and energy. The next section describes and formalizes our contributions.

III. Accuracy-Time-Energy Trade-Off

In this section, we propose our novel approach, including a new metric and methodology for analyzing the accuracy-time-energy trade-off of anomaly detection methods, aiming to reduce the energy impact of anomaly detection while maintaining high detection performance.

A. SaveUp: Relative Energy Efficiency Metric

The new proposed metric, called saveUp, is inspired by the speedup runtime metric used in parallel and distributed systems. Intuitively, it is a number representing the relative energy performance of two methods processing the same problem. Formally, it is defined by Equation (1), where E is the energy consumption of the method to be analyzed, compared to E_{ref} that is the energy consumption of a given baseline method:

$$saveUp = E_{ref}/E \tag{1}$$

The saveUp metric indicates an improvement in energy performance when it is greater than 1 and a degradation when it is less than 1. In the context of anomaly detection, when the saveUp metric of a detection method A is $s > 1$, it means that A was s times more energy-efficient compared to the baseline method. In the context of parallel and distributed systems, analogous to the speedup metric, saveUp can give the relative

[2]https://github.com/influxdata/telegraf

[3]https://github.com/hubblo-org/scaphandre

[4]https://sustainable-computing.io

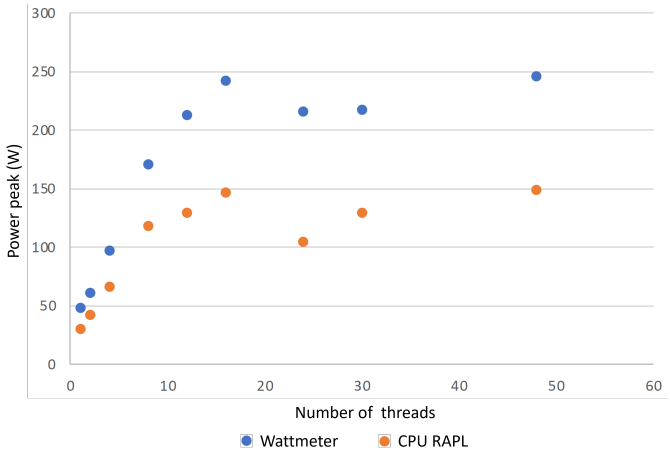

Fig. 1. Rapl estimation vs wattmeter measures.

energy improvement of a parallel method over its centralized version.

B. ATE: Accuracy-Time-Energy Performance Benchmarking

Here, we propose a novel benchmarking methodology called accuracy-time-energy (ATE). It allows the comparison of computational methods based on metrics regarding multiple dimensions: (i) accuracy; (ii) runtime; and (iii) energy consumption. It aims to set a new standard for computer science benchmarking, and is based on the established and well-known skyline queries [29], also referenced by Pareto optimality or the maximum vector problem in multidimensional datasets. In this case, method benchmarking and recommendation take all three dimensions into consideration using the skyline query operator $\succ$ defined in (2). It is equivalent to the problem of finding the maxima of a set of vectors. Given vectors, $v \in \mathbb{R}^n$ and $q \in \mathbb{R}^n$, v dominates q if it is as good as q in every dimension and superior in at least one [30]. The skyline query returns all data (vectors) that are not dominated.

$$v \succ q \iff \forall i \in [n] \mid v[i] \succeq q[i] \wedge \exists j \in [n] \mid v[j] \succ q[j] \quad (2)$$

In our methodology, the skyline operator returns the methods that cannot be dominated by any other method regarding the three accuracy-time-energy dimensions. In the context of anomaly detection, it returns the anomaly detection methods with the best relative accuracy-time-energy performance.

C. Metrics Measurement

This section describes our technique for measuring metrics used for applying the ATE methodology. The capture of metrics is based on three different processes.

The first process is responsible for measuring accuracy. At the end of the execution, each method evaluates the detected anomalies against the labeled anomalies, resulting in accuracy performance metrics, i.e., F1 score, precision, recall, AUC-ROC, etc.

The second process measures the runtime of anomaly detection methods. It uses timestamps at the beginning and end of its run. This information is sent to a time series database to tag events on the server. These tags will be used to extract the metrics required for the analysis.

For energy analysis, different strategies are available. After some evaluation, we decided to measure the direct energy consumption of the whole system rather than using individual CPU probes (RAPL). We observed that RAPL is not suitable for measuring energy consumption when multi-threading is enabled (see Figure 1). The data and implementation code of our three processes are publicly available[5].

IV. Experimental Evaluation

This section focuses on evaluating the saveUp metric and the ATE benchmarking methodology based on different state-of-the-art anomaly detection algorithms and large time series datasets. ATE is expected to help select the best anomaly detection methods based on the trade-off between accuracy, time, and energy impact.

This paper also explores the ability to parallelize algorithms using the built-in parallel implementation. The effect of parallel computing on detection accuracy has been previously analyzed [31], however, the effect on energy consumption is unknown. The experimental settings are described below.

a) Methods: Most anomaly detection methods originate from the outlier detection area, focusing on identifying punctual anomalies [32]. For this comparison, 9 of the most commonly adopted unsupervised state-of-the-art outlier detection algorithms were selected [31], namely: ECOD, COPOD, LODA, HBOS, K-NN, PCA, CBLOF, MCD, and IForest. Algorithm implementations are publicly available in the PyOD framework [33], [34] with the hyperparameters defined in PyOD by default.

b) Datasets: We performed experiments on a generated dataset based on the GutenTAG tool [35]. In this scenario, we increased the number of dimensions from 2, 3, 5, to 10 in order to test with different dimensional time series. Higher dimensions were not considered for this global analysis to avoid timeout issues for some of the adopted methods (e.g., K-NN). For this experiment, the number of data points generated is 3 million. This dataset is used as the main benchmark because it is simpler to scale the number of elements and dimensions.

We also selected two real-world multivariate time series datasets preprocessed by the work of Schmidl et al. [8], whic is currently the most comprehensive experimental review of anomaly detection algorithms. In particular, we selected the biggest datasets in both size and dimensions, namely Kitsune [36] and LTDB [37], [38]. The goal of using these datasets is to validate analysis results obtained from the simulated dataset.

c) Hardware setup: All experiments were performed on a machine equipped with a Intel(R) Core(TM) i9-10900X CPU @ 3.70GHz (with 10 physical CPU cores, appearing as 20

[5]https://github.com/RebeccaSalles/ATE

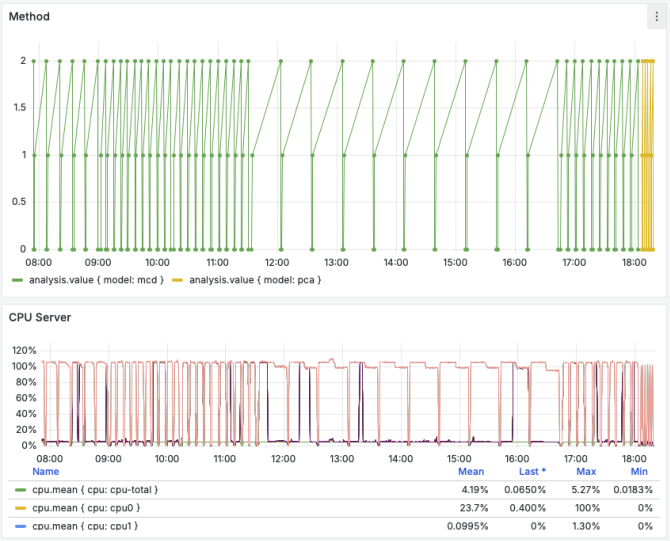

Fig. 2. Example of execution run for one method on the same dataset

logical cores (threads)), with a RAM capacity of 128 GB. Each node runs on Linux Debian 11.11.

The accuracy metric adopted to measure the detection accuracy of all algorithms is the AUC-ROC. For assessing computing efficiency, wall time in seconds (WT) and speedup metrics are measured. Finally, the adopted energy consumption metrics are power peak (PP) in watts (indicative value to evaluate grid stress), cumulative power consumption (CE) in watt per hour (w/h), and the proposed saveUp metric.

All of the metrics are sent to a time series database in order to be centralized. Because the metrics come from different sources (the algorithm, the system, and the wattmeter), different collectors are used. With the generated dataset, runs can be very short for some fast methods. In this context, experiments are run 10 times for each method (see the execution run for one method on the same dataset 2).

For single-threaded analysis, we constrain the runs with the following parameters:

- NUMBA_NUM_THREADS with a parameter of 1,
- OMP_NUM_THREADS with a parameter of 1,
- TF_NUM_INTEROP_THREADS with a parameter of 1,
- MKL_NUM_THREADS with a parameter of 1,
- OPENBLAS_NUM_THREADS with a parameter of 1,
- CUDA_VISIBLE_DEVICES with an empty parameter.

While for multi-threaded analysis, we re-enable the parallel capabilities of each method by removing the single-threaded parameters.

V. Results and Discussion

This section presents the results of the experimental analysis based on our simulated dataset. It is split into three different analyses: single-threaded (Section V-A), multi-threaded (Section V-B), and a global analysis that includes both setups using the skyline query operator (Section V-C). It is necessary to split the analyses into multi-threaded and single-threaded

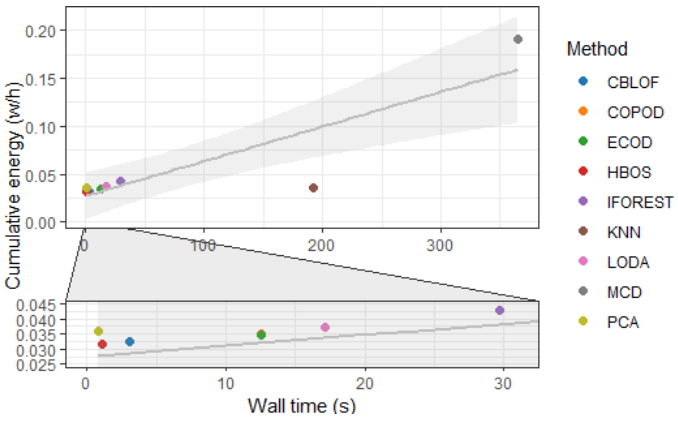

Fig. 3. Association between wall time and cumulative energy for all methods

because their behaviors are quite different and require specific discussions.

A. Single-threaded run

Average metrics of the single-threaded runs (for 5 dim. time series) are presented in Table I. Regarding accuracy, most methods present high performance results except ECOD, HBOS, COPOD, and K-NN. However, for a more comprehensive analysis, we look at wall time and cumulative energy. Usually, wall time is considered a proxy for relative energy consumption, and it is assumed that reducing one means reducing the other, without further analysis. However, as it is clear from Figure 3, we find that both metrics are not always directly correlated.

A clear example is K-NN, which despite of being one of the slowest methods, consumes as little energy as the best-performing cluster of methods. And even within this cluster (zoom area in Figure 3), we see that CBLOF consumes 33% less energy despite being more than three times slower than PCA. The main possible reason is that some methods may adopt a more optimized use of CPU capabilities, reducing energy consumption. This refers to the efficient utilization of processor resources — such as parallelism, memory access patterns, and vectorized operations — which minimizes idle time and redundant computations, leading to lower power usage. Computational efficiency can be evaluated based on the energy cost per second (cumulative energy divided by wall time). For example, the energy cost of PCA for a 1-second runtime is 155.180, while CBLOF has a 1-second runtime cost of 37.839 (a lower value indicates better use of CPU capabilities). However, a study of how efficiently each algorithm uses CPU resources that impact energy use is considered an area of further research.

Considering all three metrics, using a parallel coordinate analysis (Fig. 4), methods can be aggregated based on their signature. In this case, the ideal combination is high accuracy, short wall time, and low energy consumption. PCA, LODA, CBLOF, and IForest conform to this expectation when compared to others. Three other methods can also be aggregated: ECOD, COPOD, and HBOS, which have lower accuracy but

TABLE I
METRICS RESULTS FOR BOTH SINGLE-THREADED AND MULTI-THREADED EXECUTIONS

Method	Single-threaded				Multi-threaded			
	AUC-ROC	WT (s)	CE (w/h)	PP (w)	diff AUC-ROC	Speedup	SaveUp	PP (w)
PCA	0.99	0.84	0.04	165	0.00	1.00	1.13	165
ECOD	0.60	12.53	0.03	162	0.00	0.99	1.06	167
CBLOF	0.94	3.08	0.03	161	0.00	0.76	0.93	261
HBOS	0.36	1.13	0.03	165	0.00	0.39	1.00	165
IForest	0.90	29.74	0.04	169	0.02	1.01	1.00	168
COPOD	0.49	12.50	0.03	161	0.00	0.98	0.94	167
LODA	0.94	17.15	0.04	168	-0.04	1.00	0.86	346
K-NN	0.11	192.55	0.04	167	0.00	1.00	0.48	252
MCD	0.99	364.86	0.19	169	0.00	0.94	1.00	265

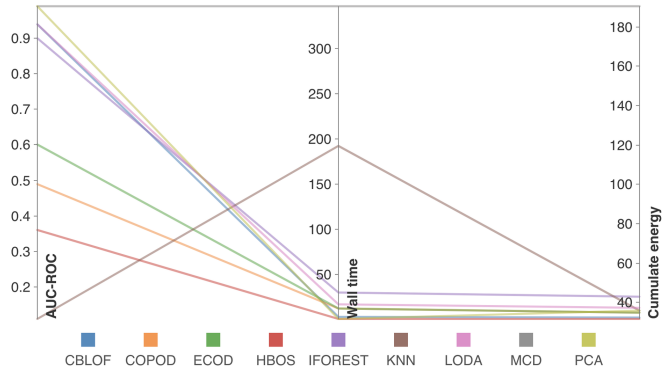

Fig. 4. Parallel coordinates visualization for the three main metrics applied to each method.

Fig. 5. Example of MCD with parallelization enabled over three consecutive runs. Each run is depicted by one of the three repeated curve patterns. The top plot shows power peaks, while the bottom plot shows CPU usage, where each colored line corresponds to one thread. It is possible to see that the parallelism appears only at the end of the algorithm.

similar wall time and energy consumption. Particularly, HBOS is the most efficient method regarding both time and energy. Finally, K-NN and MCD exhibit their own unique patterns.

Based on this analysis, it is possible to select the methods that best fit the requirements of a given anomaly detection application. In the context of a single-threaded runtime, results indicate that the accuracy of K-NN is not worth the wall time, while MCD is by far the slowest and most energy-costly method resulting in an accuracy comparable to significantly more efficient methods. A better selection may be between PCA and CBLOF, which best maximize accuracy and minimize time and energy costs. Ultimately, the selection depends on the application priorities. If accuracy and time take overall precedence, PCA would be preferable. However, CBLOF produces comparable accuracy and lower energy consumption. In that case, when a slight increase in response time has no negative impact, CBLOF is the preferred choice for greener applications.

The table also shows that the Power Peak is almost the same for most methods. This is easily explained because for this run the energy consumption is limited to that of a single thread. Thus, each method stresses the system at the same level. The same results can be computed for different numbers of time series dimensions: 2, 3, and 10. The accuracy, wall time, and energy follow the same pattern regardless of the number of dimensions.

B. Multi-threaded run

This section presents results on the parallelized versions of the same algorithms. For this analysis, three different metrics are used to evaluate performance: the difference in accuracy [6], speedup (wall time divided by parallel wall time), and saveUp (Section III-A). The results of these metrics are presented in Tab. I.

These results confirm that, as expected, the difference in accuracy is not significant. Regarding speedup, overall performance is poor because most methods do not benefit from full parallelism and only use it at specific moments of the algorithm run. This behavior is presented in the bottom plot of Figure 5.

Finally, based on saveUp, some methods show a decrease in energy performance, such as LODA and K-NN. Other methods are slightly affected in terms of energy consumption.

[6]The methods are supposed to be unaffected by parallelism [31].

Only ECOD and PCA presented 6% and 13% improvements in energy consumption, respectively, which by themselves might result in relevant mid- to long-term impacts on energy savings. In the case of parallel ECOD, for some energy-constrained applications, it can be beneficial to have a 1% decrease in time (0.99 speedup), in exchange for 6% increase in energy savings.

However, in the context of power limitation, to reduce the energy impact on the electrical grid, it is important to minimize energy peaks. Parallelism increases the peak size as observed in the top plot of Figure 5. So methods that exhibit a higher power peak compared to single-threaded mode demonstrate the ability to run in parallel (Table I). Overall, the reason why some methods do not benefit from parallel execution in terms of energy efficiency could be due to a combination of factors: the poor use of full parallelism (Figure 5), the unoptimized use of CPU resources (as discussed in Section V-A), and the overoccurrence of energy peaks when running multiple threads. Still, the study of the specific conditions that impact the multi-threaded energy efficiency for a particular method is also a subject for further research.

Thus, for most methods, if there is no significant speedup or saveUp it is not worth using multi-threading, and it is better to disable this feature to avoid significantly impacting the electrical grid and the server's power consumption. PCA and IForest were the only methods that maintained their performance without any losses in speedup, saveUp, or power peak. Among them, PCA brings the highest gain regarding energy consumption. However, ultimately, the energy impact of parallelism can only be clearly identified with long-running algorithms.

C. Overall results

This section presents the aggregated results of both single and multi-threaded analysis and the usage of the skyline query operator for accuracy-time-energy performance benchmarking as proposed in Section III-B. Table II summarizes the results. It presents the rankings of the methods based on each metric presented in Table I (higher values represent better-evaluated methods). The sum of these rankings respective to all three dimensions is used to provide an aggregated ranking for both single and multi-threaded runs, respectively. For example, the sum of CBLOF rankings is ranked the highest $(22/27)$ for single-threaded runs. Finally, an overall aggregated ranking respective to all six metrics of both runs is computed. We observe that these aggregated rankings align with our previous analysis of methods in both single and multi-threaded runs.

Moreover, Table II presents the results of the skyline query operator, marking the methods that cannot be dominated by any other method across the three metrics of each run and across all six metrics considering both runs. The skyline query results support the selection of the best methods that fit all of our metrics. It corroborates the aggregated rankings and our previous discussions, indicating the potential of the proposed ATE methodology.

We observe that time and energy can have uncorrelated rankings. For example, for single-threaded results, PCA is ranked the fastest method, but HBOS and CBLOF are ranked the least energy-costly. In this case, CBLOF balances well both dimensions. Similarly, based on rankings for multi-threaded results, PCA also has the best balance of time-energy performance, despite not providing the highest speedup.

Finally, we can use one of these methods as a baseline for computing the saveUp and conducting a comparative analysis of energy efficiency. For this, we adopt PCA, one of the oldest, simplest, and most established methods, presenting average relative energy consumption on single-threaded runs. Table III shows how each method compares to PCA energy consumption. SaveUp indicates the opportunity for redesigning more modern methods like LODA, IForest, and MCD to reduce their energy footprint. This is particularly important for an online anomaly detection scenario, where, for example, the use of HBOS could save up to 472.4 kWh per year compared to PCA, which is equivalent to 15 kg of CO_2 per year if the computation is performed in France.

Based on the validation datasets Kitsune and LTDB, depending on the time series, the experimentation shows that accuracy is impacted, and the real data are more demanding of all methods. Overall, the ranking of methods gives the same results for the single-threaded run, with PCA, CBLOF, and HBOS being the most accurate/efficient across the three metrics.

VI. Conclusion

This paper addresses for the first time the demand for benchmarking time series anomaly detection methods based on three dimensions: accuracy, time, and energy consumption. We contribute by introducing (i) the saveUp metric to evaluate how each method compares to others regarding energy consumption, and (ii) a new methodology to benchmark methods considering all three dimensions.

In the context of an online scenario, our experimental evaluation, conducted with both single-threaded and multi-threaded implementations, reveals that some state-of-the-art methods are not sustainable. Focusing only on better accuracy without considering power consumption can be neglectful, and runtime analysis is not always a reliable proxy. Our findings highlight the methods that should be redesigned to reduce their footprint and be more suitable for online big data applications.

Further analyses can be conducted to evaluate the energy impacts on a system in production, or based on the use of GPUs. Moreover, the saveUp metric has the potential to be widely adopted to evaluate algorithms across other fields and domain applications.

References

[1] P. Esling and C. Agon, "Time-series data mining," *ACM Computing Surveys*, vol. 45, no. 1, 2012.

[2] H. Wang, M. Bah, and M. Hammad, "Progress in Outlier Detection Techniques: A Survey," *IEEE Access*, vol. 7, pp. 107 964–108 000, 2019.

[3] M. Pimentel, D. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

TABLE II

RANKINGS OF METHODS AND SKYLINE QUERY RESULTS BASED ON PERFORMANCE METRICS FOR BOTH SINGLE- AND MULTI-THREADED RUNS (TABLE I)

Method	Single-threaded					Multi-threaded					All	
	Rankings				Skyline query	Rankings				Skyline query	Rankings	Skyline query
	AUC-ROC	WT (s)	CE (w/h)	Aggregated		diff AUC-ROC	Speedup	SaveUp	Aggregated		Aggregated	
PCA	8	9	4	8	✓	8	8	9	9	✓	9	✓
ECOD	4	5	7	6		8	5	8	7		8	✓
CBLOF	7	7	8	9	✓	8	2	3	3		7	✓
HBOS	2	8	9	7	✓	8	1	7	6		7	✓
IForest	5	3	2	2		9	9	6	8	✓	5	✓
COPOD	3	6	6	5		8	4	4	6		4	✓
LODA	6	4	3	4		1	7	2	2		3	
K-NN	1	2	4	1		8	6	1	4		2	
MCD	9	1	1	3		2	3	5	2		1	

TABLE III

SAVEUP RESULTS WITH PCA AS BASELINE FOR SINGLE-THREADED RUN

Method	HBOS	CBLOF	ECOD	COPOD	K-NN	PCA	LODA	IForest	MCD
SaveUp	1.14	1.11	1.04	1.03	1.00	1.00	0.96	0.84	0.19

[4] M. Olteanu, F. Rossi, and F. Yger, "Meta-survey on outlier and anomaly detection," *Neurocomputing*, vol. 555, p. 126634, 2023.

[5] Ö. E. Demirkol and A. DEMİRKOL, "Energy efficiency with an application container," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 26, no. 2, pp. 1129–1139, 2018.

[6] M. Avgerinou, P. Bertoldi, and L. Castellazzi, "Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency," *Energies*, vol. 10, no. 10, p. 1470, 2017.

[7] A. Ntroumpogiannis, M. Giannoulis, N. Myrtakis, V. Christophides, E. Simon, and I. Tsamardinos, "A meta-level analysis of online anomaly detectors," *The VLDB Journal*, vol. 32, no. 4, pp. 845–886, 2023.

[8] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: A comprehensive evaluation," *Proceedings of the VLDB Endowment*, vol. 15, no. 9, p. 1779 – 1797, 2022.

[9] M. Munir, M. A. Chattha, A. Dengel, and S. Ahmed, "A comparative analysis of traditional and deep learning-based anomaly detection methods for streaming data," in *2019 18th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2019, pp. 561–566.

[10] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Computing Surveys*, vol. 57, no. 1, oct 2024.

[11] G. Li and J. J. Jung, "Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges," *Information Fusion*, vol. 91, pp. 93–102, 2023.

[12] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[13] R. Buyya, "Parmon: a portable and scalable monitoring system for clusters," *Software: Practice and Experience*, vol. 30, no. 7, pp. 723–739, 2000.

[14] D. Narayanan, E. Thereska, and A. Ailamaki, "Continuous resource monitoring for self-predicting dbms," in *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE, 2005, pp. 239–248.

[15] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "Rapl in action: Experiences in using rapl for power measurements," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 3, no. 2, pp. 1–26, 2018.

[16] G. Luo, B. Guo, Y. Shen, H. Liao, and L. Ren, "Analysis and optimization of embedded software energy consumption on the source code and algorithm level," in *2009 Fourth International Conference on Embedded and Multimedia Computing*. IEEE, 2009, pp. 1–5.

[17] V. Berry, A. Castelltort, B. Lange, J. Teriihoania, C. Tibermacine, and C. Trubiani, "Is it worth migrating a monolith to microservices? an experience report on performance, availability and energy usage," in *2024 IEEE International Conference on Web Services (ICWS)*. IEEE, 2024, pp. 944–954.

[18] S. Kreten, A. Guldner, and S. Naumann, "An analysis of the energy consumption behavior of scaled, containerized web apps," *Sustainability*, vol. 10, no. 8, p. 2710, 2018.

[19] E. A. Santos, C. McLean, C. Solinas, and A. Hindle, "How does docker affect energy consumption? evaluating workloads in and out of docker containers," *Journal of Systems and Software*, vol. 146, pp. 14–25, 2018.

[20] Z. Li, H. Wei, C. Lian, and S. Qin, "Docker-based energy management system development and deployment methods," in *2020 4th International Conference on Power and Energy Engineering (ICPEE)*. IEEE, 2020, pp. 1–5.

[21] I. M. Murwantara and P. Yugopuspito, "Evaluating energy consumption in a different virtualization within a cloud system," in *2018 4th International Conference on Science and Technology (ICST)*. IEEE, 2018, pp. 1–6.

[22] Y. Zhang, W. Chen, and J. Black, "Anomaly detection in premise energy consumption data," in *2011 IEEE power and energy society general meeting*. IEEE, 2011, pp. 1–8.

[23] C. Chahla, H. Snoussi, L. Merghem, and M. Esseghir, "A deep learning approach for anomaly detection and prediction in power consumption data," *Energy Efficiency*, vol. 13, no. 8, pp. 1633–1651, 2020.

[24] S. F. Luna-Romero, X. Serrano-Guerrero, M. A. de Souza, and G. Escrivá-Escrivà, "Enhancing anomaly detection in electrical consumption profiles through computational intelligence," *Energy Reports*, vol. 11, pp. 951–962, 2024.

[25] Y. Himeur, K. Ghanem, A. Alsalemi, F. Bensaali, and A. Amira, "Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives," *Applied Energy*, vol. 287, p. 116601, 2021.

[26] A. Copiaco, Y. Himeur, A. Amira, W. Mansoor, F. Fadli, S. Atalla, and S. S. Sohail, "An innovative deep anomaly detection of building energy consumption using energy time-series images," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105775, 2023.

[27] J.-S. Chou and A. S. Telaga, "Real-time detection of anomalous power consumption," *Renewable and Sustainable Energy Reviews*, vol. 33, pp. 400–411, 2014.

[28] S. Yin, H. Yang, K. Xu, C. Zhu, S. Zhang, and G. Liu, "Dynamic real–time abnormal energy consumption detection and energy efficiency optimization analysis considering uncertainty," *Applied Energy*, vol. 307, p. 118314, 2022.

[29] E. Tiakas, A. N. Papadopoulos, and Y. Manolopoulos, "Skyline queries: An introduction," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*. IEEE, 2015, pp. 1–6.

[30] M. E. Schüle, A. Kulikov, A. Kemper, and T. Neumann, "Artful skyline computation for in-memory database systems," in *New Trends in Databases and Information Systems: ADBIS 2020 Short Papers, Lyon, France, August 25–27, 2020, Proceedings 24*. Springer, 2020, pp. 3–12.

[31] R. Salles, B. Lange, R. Akbarinia, F. Masseglia, E. Ogasawara, and E. Pacitti, "Scalable and accurate online multivariate anomaly detection," *Information Systems*, p. 102524, 2025.

[32] A. Blázquez-García, A. Conde, U. Mori, and J. Lozano, "A Review on Outlier/Anomaly Detection in Time Series Data," *ACM Computing Surveys*, vol. 54, no. 3, 2021.

[33] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.

[34] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 32 142–32 159.

[35] P. Wenig, S. Schmidl, and T. Papenbrock, "Timeeval: A benchmarking toolkit for time series anomaly detection algorithms," *Proceedings of the VLDB Endowment (PVLDB)*, vol. 15, no. 12, pp. 3678 – 3681, 2022.

[36] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.

[37] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[38] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *IEEE engineering in medicine and biology magazine*, vol. 20, no. 3, pp. 45–50, 2001.