BenchCouncil Transactions on Benchmarks, Standards and Evaluations 3 (2023) 100121

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Research article

StreamAD: A cloud platform metrics-oriented benchmark for unsupervised online anomaly detection

Jiahui Xu [a,b,1], Chengxiang Lin [a,b,1], Fengrui Liu [a,b,1], Yang Wang [a,b], Wei Xiong [a,b], Zhenyu Li [a,b], Hongtao Guan [a,b], Gaogang Xie [b,c,*]

[a] Institute of Computing Technology, Chinese Academy of Sciences, China
[b] University of Chinese Academy of Sciences, China
[c] Computer Network Information Center, Chinese Academy of Sciences, China

ARTICLE INFO

Keywords:
Cloud metrics
Online machine learning
Anomaly detection
Streaming data
Time series

ABSTRACT

Cloud platforms, serving as fundamental infrastructure, play a significant role in developing modern applications. In recent years, there has been growing interest among researchers in utilizing machine learning algorithms to rapidly detect and diagnose faults within complex cloud platforms, aiming to improve the quality of service and optimize system performance. There is a need for online anomaly detection on cloud platform metrics to provide timely fault alerts. To assist Site Reliability Engineers (SREs) in selecting suitable anomaly detection algorithms based on specific use cases, we introduce a benchmark called StreamAD. This benchmark offers three-fold contributions: (1) it encompasses eleven unsupervised algorithms with open-source code; (2) it abstracts various common operators for online anomaly detection which enhances the efficiency of algorithm development; (3) it provides extensive comparisons of various algorithms using different evaluation methods; With StreamAD, researchers can efficiently conduct comprehensive evaluations for new algorithms, which can further facilitate research in this area. The code of StreamAD is published at https://github.com/Fengrui-Liu/StreamAD.

1. Introduction

Cloud platform [1] is a type of computing infrastructure that provides hardware and software resources over the internet, such as virtual machines, storage, and networking capabilities. It can facilitate the developers building and deploying software applications.

With the growing market of cloud platform, its scale has become enormous. However, the prosperity of cloud platforms also brings significant challenges to Site Reliability Engineers (SREs) in detecting and diagnosing faults within large-scale cloud platforms. The computing infrastructures providing services need to guarantee Service Level Agreements (SLAs) to customers. Unexpected service downtime can greatly impact stability objectives and lead to substantial financial losses.

Benefiting from the intuitive visualization form of time-series metric data, such as metric dashboards, they are often the primary objects for anomaly detection in cloud platforms. SREs can easily point out whether the collected metrics are as expected. In order to reduce labor and enhance the quality of service, major cloud providers such as Microsoft Azure [2], Google Cloud [3], Amazon Cloud [4] and Alibaba Cloud [5] have adopted machine learning and artificial intelligence technologies to assist SREs in detecting anomalies.

Metrics anomaly detection presents a challenging task due to the following reasons [6,7]:

- Lack of labeled data. Anomalous data is rare compared to normal data, and identifying specific anomalies that warrant attention can be difficult. Practical application scenarios are open-ended, making it difficult to define the anomalies that should be detected. The confirmation of specific anomalies, such as their beginning and duration, requires reliable input from SREs. As a result, obtaining accurate labels is challenging [8]. The manually labeling process is also prone to errors [7], with a wide-ranging discussions regarding the flaws in current public datasets. This issue stems from the subjective judgments made while assigning ground truth labels. The lack of labeled data presents challenges in designing, training, and evaluating models effectively.
- Online detection. Cloud platform metrics are often monitored in real-time, in order to quickly alert when a fault is detected. This helps reduce the mean time to repair (MTTR), which is

* Corresponding author.
E-mail addresses: xujiahui21b@ict.ac.cn (J. Xu), linchengxiang21s@ict.ac.cn (C. Lin), liufengrui18z@ict.ac.cn (F. Liu), wangyang2013@ict.ac.cn (Y. Wang), xiongwei20b@ict.ac.cn (W. Xiong), zyli@ict.ac.cn (Z. Li), guanhongtao@ict.ac.cn (H. Guan), xie@cnic.com (G. Xie).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.tbench.2023.100121
Received 3 May 2023; Received in revised form 3 July 2023; Accepted 5 July 2023
Available online 6 July 2023

critical for maintaining service level agreements. Thus, there is a high demand for algorithms to accurately and efficiently detect metric anomalies with an online manner. An effective online anomaly detection algorithm must continually process incoming data streams and update its model online to ensure accurate and reliable detection. In the situations when metrics experience significant changes in data distribution, known as concept drift [9], algorithms need to adapt to these changes promptly to prevent false alarms from occurring.

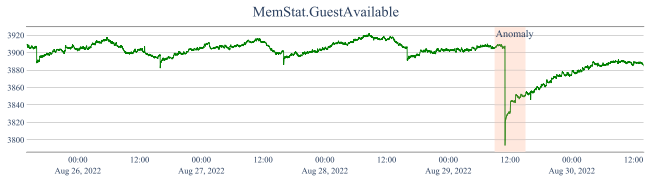
- Data dimension. A cloud platform metric can describe a specific aspect of a cloud platform, such as CPU utilization, network received packets or memory usage. Each metric is represented as an univariate time series, where the series is independent of others, i.e. the data dimension is univariate. However, in the event of a host fault in a cloud platform, multiple metrics may exhibit anomalous behavior. An underlying assumption is that there is internal interaction between different metrics. Thus, they can be used together to detect anomalies utilizing a process known as multivariate detection. However, simply combining the anomaly detection results of each univariate time series performs poorly for multivariate anomaly detection methods [10]. This naive approach fails to capture the inter-dependencies among metrics within a service. Therefore, there is a growing need for dedicated algorithms that can effectively handle multivariate data streams.
- Domain-specific datasets and benchmarks. Although researchers have published several datasets and benchmarks for anomaly detection [11–13], they are not specific to a particular domain. Nevertheless, there are significant differences in data characteristics across various fields. For instance, ECG datasets [14], voice datasets [15], and cloud platform metrics datasets [8] are all in time-series format, they can differ greatly in periodicity, range of values, and other key characteristics. The lack of domain-specific datasets and benchmarks for cloud platform metrics still persists.

As can be seen from the aforementioned challenges, metrics anomaly detection algorithms for real cloud platforms need to be unsupervised, since high-quality labeled data may not always be accessible. Additionally, these algorithms should detect anomalies with an online manner, enabling them to report fault alarms timely.

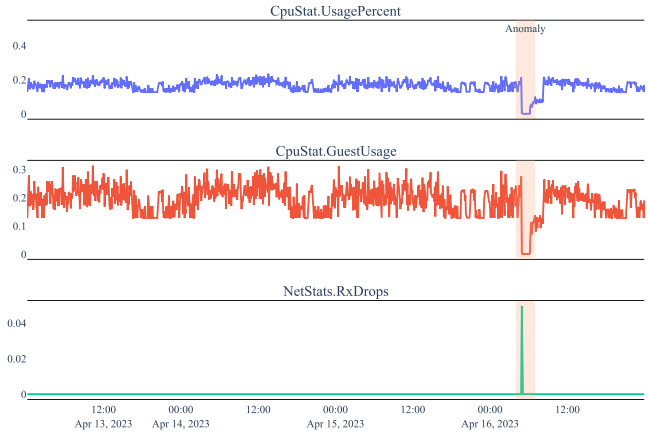
Although researchers have designed and contributed various unsupervised algorithms for online anomaly detection, there is a lack of a comprehensive benchmark to evaluate their effectiveness in cloud platform metrics. To tackle above issue, we propose StreamAD, which is a domain-specific benchmark for unsupervised online anomaly detection of cloud platform metrics. The primary contributions of StreamAD are summarized as follows:

- StreamAD collects eleven unsupervised online anomaly detection algorithms, encapsulating them using a unified and easy-to-use application programming interface (API). It can serve as an out-of-the-box anomaly detection module for quick case validation. All the code is open-source.
- We abstracts various common operators for different online anomaly detection, accompanied by data process methods. It can greatly facilitate researchers using StreamAD to develop new algorithms.
- StreamAD focuses on cloud platform metrics dataset, providing extensive comparisons of various algorithms using different evaluation methods. The results form a benchmark for cloud platform metrics anomaly detection.

StreamAD is dedicated to quickly verifying the effectiveness of different algorithms on use cases, enabling the application of machine learning-based algorithms for real-world cloud platform metrics. It also helps researchers in rapidly developing and comparing new algorithms, promoting further research and development in this rapidly evolving research domain.



(a) Univariate metric anomaly in cloud platform



(b) Multivariate metrics anomaly in cloud platform

Fig. 1. Example of anomalies for cloud platform metrics.

2. Background

2.1. Cloud platform metrics

The primary focus of anomaly detection is on the observable data objects within cloud platforms. Observability refers to the ability to monitor and comprehend the operational state of a system's underlying infrastructure, platform, and applications through their external outputs. In a complex cloud platform system, observability assists in describing the system's current status, verifying the proper execution of each component's intended logic, identifying performance bottlenecks, and tracking optimizations for better system management. In the event of anomalies, observable data objects play a crucial role in real-time data collection and visualization of various key metrics. By analyzing these observable data, SREs can swiftly identify and address faults within complex cloud platforms, leading to optimized system performance and enhanced system reliability.

Metrics serve as a fundamental component of observable data objects in cloud platforms. A metric is a numerical value or counter that represents the state of the system, which is atomic and cumulative. Each metric can be regarded as a logical measurement unit, typically representing data statistics updated over time. Although the specific metrics monitored by vary cloud platforms can be different. Take Google Cloud metrics [16] as an example. A typical set of cloud platform metrics includes five categories, including CPU, System, Memory, Block, and Network, which can cover various aspects of the cloud platform.

As each cloud-platform metric can be represented in a time series, it is natural to conduct independent analysis on each metric, namely univariate metric anomaly detection. For instance, a *MemStat.GuestAvaliable* metric from a cloud platform, as shown in Fig. 1(a). The metric is represented as time-series data which is continuously extended as long as it is under continuous observation. This nearly real-time data observation process enables the system status to be monitored online, making it possible to alert the faults in a timely manner.

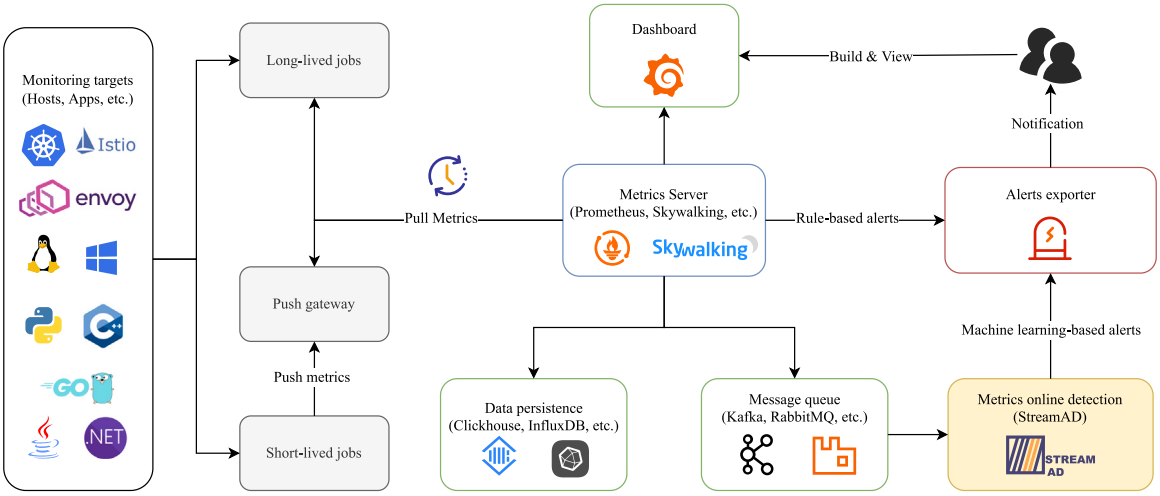


Fig. 2. Architecture example of metrics online detection in a cloud platform.

In addition, some cloud platform faults may be reflected in multiple metrics. Take Fig. 1(b) as an example, a network issue causes a sharp increase in the *NetStats.RxDrops* metric, accompanied with a decrease in the *CpuStat.UsagePercent* and *CpuStat.GuestUsage* metrics. The root cause of this fault is that the host has suffered from receiving packets (*RxPackets*) loss. The deployed services fail to response external requests, resulting in low CPU usage. In this case, multiple metrics exhibit abnormal behaviors during the fault. Under the assumption that there is an internal interaction among different metrics, they can be analyzed together, leading to achieve multivariate metrics anomaly detection. StreamAD covers for both univariate and multivariate metrics detection.

2.2. The role of metric anomaly detection

In real production environments of cloud platforms, as depicted in Fig. 2, different metrics can be collected and reformatted by various agents or probes. These metrics may include data from fundamental host machines, resource management controllers, and applications constructed using different programming languages. Both short-lived and long-lived jobs generate metrics in an online manner and export them to the metrics server using push and pull methods respectively. The metrics server stores data streams into a time series database, achieving data persistence that can be utilized for data retrieval and backtracking. Some popular metrics servers, such as Prometheus [17] and Skywalking [18], have built-in rule modules that allow SREs to implement anomaly detection by pre-setting rules. However, manual operations by setting alert rules struggle to adapt to large-scale cloud platforms, as they heavily rely on expert knowledge and are error-prone. Thus, fully-automated operation pipelines powered by machine learning capabilities become a promising approach for achieving SLA goals.

StreamAD can serve as a logical unit that is dedicated to utilizing machine learning technology for metric monitoring. It is capable of subscribing to message queues, allowing it to receive and analyze streaming data based on algorithmic processing logic. Once the observed data has been analyzed, it will be scored accordingly. Those data with a high anomaly score are then sent to the alert exporter, and then further notify the users in time. SREs can trace the metrics records via a customized dashboard and deal with the faults in the cloud platform.

3. Related work

Anomaly detection is a broad topic that has been applied in different applications, leading to significant research efforts over the years [24–27]. Researchers have devoted substantial effort publishing benchmarks, and we provide a summary of related work in Table 1.

ADBench [11] is a comprehensive anomaly detection benchmark that includes unsupervised, semi-supervised, and fully-supervised algorithms. It analyzes the performance of thirty algorithms under different types of anomalies by simulating different environments. However, this benchmark only focuses on tabular data, which may not be suitable for time-series data in cloud platforms.

TODS [13,19] constructs a full-stack automated machine learning system for anomaly detection. It is a benchmark that identifies four multivariate real-world datasets from different domains and benchmarks nine algorithms on synthetic and real-world datasets. TODS also publishes preprocess and synthetic scripts, as well as algorithm implementations.

NAB [20,21] focuses on scenarios of online anomaly detection in practical applications. Although all algorithms in this benchmark are designed for online anomaly detection and use a scoring algorithm designed for streaming data, it only evaluates univariate anomaly detection algorithms and lacks the discussion of multivariate time series data. Furthermore, this benchmark is not currently maintained and does not cover new online anomaly detection algorithms.

Exathlon [22] is a benchmark that focuses on time series data. It provides a new analytical perspective on time series anomaly detection, which is the interpretability of the detection results. It focuses on Spark application monitoring and provides an end-to-end pipeline for explainable time series anomaly detection. However, this benchmark is for offline analysis.

UTSD [23] is a benchmark that focuses on univariate time series and provides a user-friendly visual interface for those series. This benchmark contributes a large number of datasets and their variants. However, the use case of this benchmark directly applies tabular data anomaly detection algorithms to time series data, which has great limitation on modeling the features of time series data.

TSB-UAD [12] is benchmark for univariate time series. It contributes a principled methodology for generating labeled anomaly detection datasets. It also reviews factors affecting the performance of methods. However, this benchmark is also for offline anomaly detection and cannot meet the requirements of online anomaly detection for cloud platforms.

Regarding the benchmark for anomaly detection of cloud platform metrics, it should have the following properties. Firstly, the benchmark should focus on time-series data, including both univariate and multivariate metrics. Secondly, the anomaly detection methods can be updated in a streaming manner, without periodic offline training. Finally, it requires an extensive validation on cloud platform metrics.

Table 1
Comparison of anomaly detection benchmarks across various properties.

Properties/Benchmark	# Algorithms	Time series	Multivariate	Streaming updates	Domain specific
ADBench [11]	30	✗	✓	✗	✗
TODS [13,19]	9	✓	✓	✗	✗
NAB [20,21]	12	✓	✗	✓	✗
Exathlon [22]	3	✓	✓	✗	✓
UTSD [23]	3	✓	✗	✗	✗
TSB-UAD [12]	12	✓	✗	✗	✗
StreamAD (Ours)	11	✓	✓	✓	✓

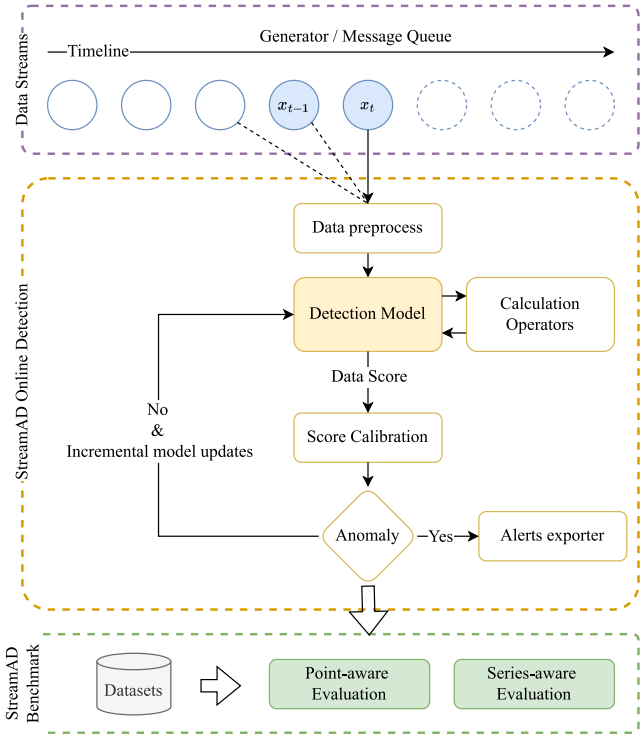


Fig. 3. The framework of StreamAD.

However, comparing the existing anomaly detection benchmarks, we have found that it is challenging for them to meet all required properties. Therefore, we propose StreamAD, which aims to fill this gap and serve as the benchmark for anomaly detection on cloud platform metrics.

4. StreamAD: benchmark details

4.1. Overview

Streaming data refer to an infinite sequence of discrete data points that are continuously generated at a constant rate over time, donated as $\mathcal{X} = \{x_1, x_2, x_3, \ldots, x_t, \ldots\}$, where x_t represents the data point generated at time t. Compared to tabular data and static time series data, streaming data do not have a predetermined length. It refers to a continuous flow of data points arriving in real time. Based on this, online anomaly detection for streaming data can be defined as the process of identifying data points or patterns in a data streams that significantly deviate from the expected behaviors.

StreamAD is proposed for metrics online anomaly detection and its framework is shown in Fig. 3. It can ingest data streams directly from message queues like Kafka or RabbitMQ. Additionally, it provides a data stream generator that can simulate a streaming data environment using the loaded dataset. Each observation in the data stream can be formulated as (t, x_t), which represents an observation

Table 2
Anomaly detection algorithms included in StreamAD.

	Algorithm	Sliding window	Seasonal
Univariate	KNN-CAD [31]	✓	✗
	SPOT [32]	✗	✗
	SR [2]	✗	✓
	Z-Score [33]	✗	✗
	OC-SVM [34]	✓	✗
	MAD [35]	✗	✗
Multivariate	xStream [36]	✓	✗
	RShash [37]	✗	✗
	HSTree [38]	✗	✗
	LODA [39]	✓	✗
	RRCF [40]	✗	✗

x_t with a timestamp t, note that x_t can be univariate or multivariate data. As each streaming data is observed, StreamAD first preprocesses the data. Typical data preprocessing methods include downsampling (aggregating data), scaling (transforming the data to a specific range), and normalization (scaling the individual samples to have unit norm).

The preprocessed data is then forwarded to the anomaly detection models. In StreamAD, there are eleven different detection algorithms as candidate models. The common methods of these algorithms are extracted as calculation operators, such as online statistics and sliding Fourier transformation [28–30]. These operators facilitate our analysis and detection of data streams in a continuous and online manner. The detection model assigns a score to each piece of data to reflect its anomaly degree. However, the output scores by different algorithms are on different scales due to their varying designs. Therefore, StreamAD provides score calibration methods that standardize the anomaly scores into a common scale and outputs them to the alert exporter. Since StreamAD is designed for online anomaly detection, when a data point is scored as normal by the detection model, the model should update itself based on the latest streaming data.

4.2. Anomaly detection algorithms

As detection models in StreamAD, anomaly detection algorithms play a crucial rule. They are the primary research objective in our benchmark. Numerous researchers [2,31,32,36–40] have contributed to the development of various algorithms, leading a prosperous research community. In pursuit of practical cloud platform metrics anomaly detection applications, StreamAD focuses on unsupervised online detection and has integrated eleven widely popular algorithms.

Although these algorithms rely on different techniques, they can be compared from two aspects. One aspect is the observation method of data streams. Some algorithms [31,34,36,39] observe the data stream through a sliding window. This kind of algorithms detect anomalies by comparing the differences between the latest window and the historical windows. While other methods [32,33,35,37,38,40] estimate the data distribution from historical data and examine whether the latest data point belongs to the distribution. The other aspect is the ability of different algorithms to capture the seasonal characteristics of a data stream. When a data pattern appears periodically in a data stream, it is usually regarded as normal. Some methods [2] can capture the seasonal patterns, while most distribution-based methods [32,33,35] cannot. In

addition, these algorithms can be categorized into two types, namely univariate and multivariate, based on the data dimension that they can handle, as outlined in Table 2. The introduction of anomaly detection algorithms for both univariate and multivariate data streams are as follows.

Univariate data streams anomaly detection refers to the algorithms that identify anomalies in the data streams containing only a single variable. These algorithms focus on analyzing the individual data. The advantages of these algorithms are their simplicity and nature intuition, as well as the great interpretability. KNN-CAD [31] is a sliding window-based method, it constructs a Hankel matrix to characterize the data streams and calculate the distance among observation windows using Mahalanobis distance. Streaming data that results in large distances may be potential anomalies. SPOT [32] and Z-Score [33] respectively assume normal streaming data conform to Pareto distribution and Gaussian distribution. Observations that fall out of the distribution are regarded as anomalies. SR [2] leverages Fourier transform to convert data streams from time domain to frequency domain, simplifying the anomaly detection task to identifying rare frequencies. OC-SVM [34] regards normal streaming data as belonging to the same class. It uses the support vector machine to determine the boundaries of normal data. Those data that cannot be classified as normal are considered anomalies. MAD [35] compares the deviation between newly arrived data and the median value of data stream histories. This detection algorithm has been proven to be effective in InfluxDB community [35].

Multivariate data streams anomaly detection identifies anomalies in data streams containing multiple variables. Different from univariate data streams, it takes into account the relationships, correlations and dependencies among various variables. This kind of anomaly detection algorithms provides us a comprehensive perspective, which enables the detection process to go beyond a specific metric and extend to a component within the cloud platform. For instance, we can simultaneously input the CPU, memory, and network metrics of a virtual machine into the algorithm to obtain an anomaly score.

xStream [36] tackles anomaly detection tasks for feature-evolving streams. As a density-based ensemble anomaly detection algorithm, it approximates the density of a point by counting its nearby neighbors at multiple scales. RShash [37] employs randomized hashing to score data points and features an elegant subspace interpretation. HSTree [38] is a fast one-class anomaly detector for evolving data streams. Utilizing mass [41] as a measure to rank anomalies, it can construct a ranking with small samples, enabling the anomaly detector to learn quickly and adapt to changes in data streams promptly. LODA [39] recognizes that the probability of observed samples valuable in determining their anomalousness. It approximates the joint probability using a collection of one-dimensional histograms, while each constructed on an input space projected onto a randomly generated vector. The use of one-dimensional histograms allows for efficient construction in one pass over the data, with simple query operations needed during classification. RRCF [40] introduces a robust random cut data structure that can serve as a sketch or synopsis of the input stream. This sketch can be efficiently updated in a dynamic data stream environment.

The above discussion presents a quick overview of eleven anomaly detection algorithms for both univariate and multivariate data streams. Some of these algorithms detect anomalies within a sliding observation window, while others incrementally update the detection model based on arriving data. Moreover, various algorithms have different capabilities in capturing seasonal patterns in data streams. Table 2 illustrates that the SR algorithm, which transforms data streams from the temporal domain into the frequency domain, can effectively detect anomalies in data streams that exhibit periodic features.

StreamAD offers a user-friendly API to access the aforementioned algorithms. Fig. 4 provides an usage example for SPOT anomaly detector, which is also the benchmark code snippet. The example code loads a benchmark dataset and simulates a stream environment using a loop. After that, the detector fits and scores each piece of data. The example illustrates how StreamAD assist users in evaluating their own use cases.

```python
from streamad.util import StreamGenerator, UnivariateDS
from streamad.model import SpotDetector

ds = UnivariateDS()
stream = StreamGenerator(ds.data)
model = SpotDetector()

for x in stream.iter_item():
    score = model.fit_score(x)
```

Fig. 4. API example for SPOT anomaly detector.

4.3. Incremental calculation operators

Due to the requirement for online detection of cloud platform metrics, an important property of online calculation is the incremental updating scheme of detection models. It differs significantly from offline calculation. In StreamAD, we extract the common online calculation methods of these algorithms as operators, which can help to enhance the efficiency of algorithm development.

A series of the operators are used for statistics. Take the variance calculation operator [42] as an example. A naive formula for calculating the variance $\sigma^{2,offline}$ of an offline dataset is:

$$\begin{aligned} \mu^{offline} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \sigma^{2,offline} &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu^{offline})^2 \end{aligned} \tag{1}$$

where n is the size of dataset, and $\mu^{offline}$ represents the mean value of the dataset. However, consider the online detection is under an infinite data stream setting, we cannot store all the history of data stream. Thus, we use the Welford's online algorithm [43] to handle the online calculation. For each incoming x, the variance incrementally updates as:

$$\begin{aligned} n &= n + 1 \\ \mu_{i+1}^{online} &= \mu_i + \frac{x - \mu_i^{online}}{n} \\ s_{i+1} &= s_i + (x - \mu_i) \times (x - \mu_{i+1}) \\ \sigma_{i+1}^{2,online} &= \frac{s_{i+1}}{n} \end{aligned} \tag{2}$$

where μ_{i+1}^{online} is the mean value of the first $i+1$ data from a stream, and s is the running sum of squares.

StreamAD has already included seven calculation operators for data statistics, and one operator for sliding Fourier transformation [28–30]. These incremental calculation operators play a vital role in efficiently processing data streams and updating models in real-time. By incorporating these calculation operators, StreamAD provides a comprehensive toolkit for developing and implementing new online anomaly detection algorithms, catering to the dynamic nature of data streams and the evolving requirements of online anomaly detection tasks.

4.4. Selection of datasets

The selection of datasets could have significantly impact on the benchmark results, as datasets from different domains exhibit varying

Table 3
Comparison of anomaly detection datasets across various properties in StreamAD.

Properties/Datasets	# Series	Avg. length	Anomaly ratio	Avg. span (# h)	Dimension	Metrics object
AIOPS_KPI [8]	29	103,588	2.65%	20.88	Univariate	container_cpu, queue, db, ping_time
MICRO [8]	29	228	1.26%	5.96	Univariate	oracle, container, docker, redis, linux
AWSCloudwatch [20]	17	3,984	0.05%	20.06	Univariate	cpu, network, request, grok, rds
GAIA [44]	279	10,156	0.78%	18.21	Univariate	zookeeper, redis, mysql
SMD [45]	28	25,300	4.16%	–	Multivariate	server machine instance

characteristics. Thus, we have chosen 5 public datasets that primarily focus on cloud platforms.

AIOPS_KPI [8] is a large-scale real-world public dataset, consisting of 27 key performance indicators (KPIs) for artificial intelligence for IT operations (AIOps). This dataset is collected from five large internet companies, including Sougo, eBay, Baidu, Tencent and Alibaba. The duration of each KPI data ranges from two to seven months, and each KPI is labeled by experienced SREs in these companies. The KPI patterns in this dataset are various.

MICRO [8] consists of metrics data from a public microservices monitoring dataset. It contains fine-grained metrics, including container, Linux system, Oracle, and Redis. The attributes of the spans on each component are aggregated into KPIs that reflect the overall status of each component. Anomalies in this dataset are simulated by fault injection (e.g., database close, container CPU stress, etc.), with labeled data recorded as fault injection timestamps.

AWSCloudwatch [20] features AWS server metrics collected by the Amazon Cloudwatch service. Example metrics include CPU Utilization, Network Bytes In, and Disk Read Bytes. This is a real-world dataset which shows us the behavior of AWS server.

GAIA [44] is comprised of one-month cloud platform monitoring data, selected from a login-action scenario in a business cloud platform system. The monitoring data includes Zookeeper, Redis and MySQL. This dataset covers different types of time series data, including change point, concept drift, periodic and stationary data. This dataset with rich variety of anomaly types can provide more comprehensively validation scenario for anomaly detection.

SMD [45] is a five-week dataset from a large Internet company, encompassing 28 different server machines. The data for each machine is divided into two equal-length segments for training and testing. It also provides labels indicating whether a point is an anomaly and the dimensions that contribute to each anomaly.

Table 3 presents a comparison of five anomaly detection datasets used in StreamAD, considering their various properties such as the number of instances, average length, anomaly ratio, average span, and metrics objects. The diversity of these selected datasets enables StreamAD to comprehensively evaluate anomaly detection algorithms.

4.5. Evaluation criteria

For time series anomaly detection evaluation, there are several measures have been proposed to assess the quality of anomaly detection algorithms. In general, these evaluation criteria can be classified into two categories, point-aware evaluation and series-aware evaluation. For these evaluation methods, precision and recall are both considered as evaluation criteria. Precision measures the proportion of relevant instances among the retrieved instances, and recall measures the proportion of relevant instances that were successfully retrieved. StreamAD includes both point-aware and series-aware evaluations to ensure a comprehensive assessment of the detection methods.

4.5.1. Point-aware evaluation

The point-aware evaluation criteria treats time series data as a collection of static data points, considering each point individually, as Fig. 5 shows. To perform the point-wise evaluation, let P and N represent the number of actual positive and negative points, while TP, FP, TN, and FN denote true positive, false positive, true negative, and false negative classifications, respectively. The following metrics are then defined:

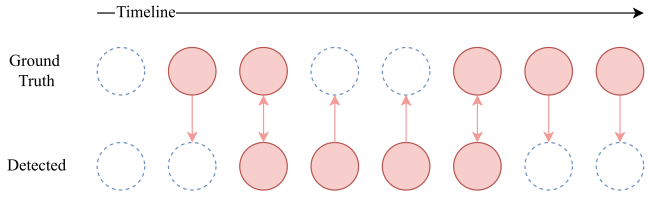


Fig. 5. Example of point-aware evaluation.

$$\begin{aligned} Precision^P &= \frac{TP}{TP+FP} \\ Recall^P &= \frac{TP}{P} = \frac{TP}{TP+FN} \end{aligned} \tag{3}$$

Based on Eq. (3), the point-aware balanced F_1^P score is calculated as the harmonic mean of the $Precision^P$ and $Recall^P$, as:

$$F_1^P = 2 \times \frac{Precision^P \times Recall^P}{Precision^P + Recall^P} \tag{4}$$

Point-aware evaluation criteria are commonly employed in the literature for assessing the performance of anomaly detection algorithms. This approach offers a simple way to compare different methods in terms of their ability to identify individual anomalous data points within a time series.

However, point-aware evaluation exhibits certain limitations. By treating time series data as an uncorrelated set of individual data points, it overlooks the inherent temporal dependencies and relationships within the data, leading to a less accurate understanding of an algorithm's performance in capturing the underlying patterns and dynamics of the time series. Furthermore, point-aware evaluation overemphasizes the ability of algorithms to identify overall labeled anomalies, instead of considering more practical, application-specific concerns like cloud platform metric alerts. In real-world applications, SREs tend to focus on accurately detecting the starting positions of abnormal fragments, which are crucial for timely alerting and effective incident response. Point-aware evaluation may not adequately address this aspect, necessitating alternative evaluation methods that consider the practical requirements and goals of cloud platform monitoring and anomaly detection.

4.5.2. Series-aware evaluation

To alleviate shortcomings of the traditional Precision and Recall measures for time series anomaly detection, researchers [20,46–48] have proposed extensions for series-aware evaluation. The key insight behind series-aware evaluation is that an anomalous segment usually represents a single anomaly event, which may encompass multiple labeled anomaly points. In this context, anomalies at different positions within the segment owing varying weights. By considering anomaly events as continuous segments rather than isolated points, series-aware evaluation provides a more holistic assessment of an algorithm's ability to detect anomalies, accounting for temporal dependencies, and practical alerting considerations in real-world cloud platform applications.

Fig. 6 shows a typical example of series-aware evaluation. In this case, the detected anomalies may partially overlap with the ground

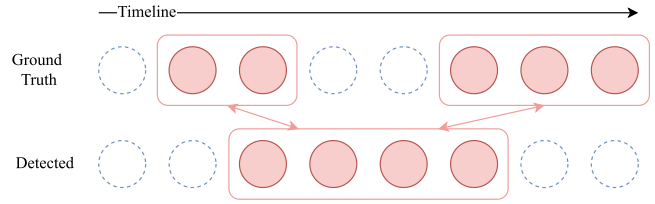


Fig. 6. Example of series-aware evaluation.

truth. Despite the small number of overlapping points lead to a low point-aware evaluation score, the detection still successfully identified the two anomalous sequence fragments, which holds practical value for cloud platform metrics monitoring. Additionally, considering the timeliness of alerts for metric anomalies, algorithms that detect the earlier portion of an anomaly sequence are preferred. Such early detection can help reduce the time required to respond to and address these anomalies. For example, in Fig. 6, the detection results that identify the true anomalies for the second ground truth sequence carry greater practical significance than those for the first sequence due to their earlier identification of the problematic segment. This aspect demonstrates the advantage of series-aware evaluation in assessing anomaly detection algorithms in practical applications.

Based on the above observations, we follow the idea from [46] and set series-aware evaluation criteria in StreamAD. Given a set of ground truth anomaly segments $R = \{R_1, .., R_{N_r}\}$ and a set of detected anomaly segments $P = \{P_1, .., P_{N_p}\}$, the $Precision^S$ and $Recall^S$ are defined as

$$\begin{aligned} Precision^S &= \frac{1}{N_P}\sum_{i=1}^{N_p} C(P_i, R) \times \sum_{j=1}^{N_r} \omega(P_i, P_i \cap R_j, \sigma) \\ Recall^S &= \frac{1}{N_r}\sum_{i=1}^{N_r} [\alpha E(R_i, P) + \\ &\qquad (1-\alpha) C(R_i, P) \times \sum_{j=1}^{N_p} \omega(R_i, R_i \cap P_j, \sigma)] \end{aligned} \tag{5}$$

where $C(\cdot)$ is the cardinality factor, which is used for scaling the rewards earned based on the overlap size and position of detected anomalies. $E(\cdot)$ represents the existing reward function, which encourages to detect every anomaly segments. In addition, α, ω, σ serve as hyperparameters that depend on the specific practical applications. For the evaluation of cloud platform metric monitoring scenarios, these parameters are selected to prioritize early detection, accommodating that the front-end bias is often observed in such contexts.

Based on Eq. (5), the series-aware balanced F_1^S score is calculated as the harmonic mean of the $Precision^S$ and $Recall^S$, as:

$$F_1^S = 2 \times \frac{Precision^S \times Recall^S}{Precision^S + Recall^S} \tag{6}$$

Compared to point-aware evaluation, series-aware evaluation can yield more informative evaluation of anomaly detection algorithms in terms of their real-world performance and utility.

5. Experimental results

In this section, we provide a comprehensive analysis of our benchmark results, addressing various aspects of the anomaly detection algorithms. Firstly, we describe the experimental settings, encompassing the configurations of datasets, hyperparameters, evaluation criteria, and the evaluation platform. Next, we present in-depth results aiming at addressing the following questions:

1. How effective are the anomaly detection algorithms across different datasets?
2. Can the efficiency of the anomaly detection algorithms satisfy the requirements of practical applications?
3. Do the space complexities of detection algorithms remain static?

5.1. Experiment setting

Datsets. As described in Section 4.4, StreamAD includes five public real-world datasets, focusing specifically on cloud platform applications. Due to the online detection paradigm employed by various detection algorithms, we allocate the first one hundred points of each streaming data for algorithm initialization. These detection algorithms are primarily designed for the transductive setting, and outputting anomaly scores for the incoming data.

Hyperparameters. Each anomaly detection algorithm in StreamAD (described in Section 4.2) has its own hyperparameters, such as the observation window for KNN-CAD algorithm and the number of trees for RRCF algorithm. To ensure a fair comparison, we used the default hyperparameter settings from the original papers for all algorithms in StreamAD.

Evaluation criteria. The benchmark of StreamAD incorporates both point-aware and series-aware evaluation methods. The point-aware evaluation adheres to the standard definition of evaluation criteria, while the series-aware evaluation accounts for the front-end bias introduced in cloud platform metric evaluation scenarios, as discussed in [46].

Evaluation platform. All experiments are conducted on a server with the following configurations: Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40 GHz, 16 cores, 32 GB RAM. The server runs Debian GNU/Linux 10 (64-bit). All the code is implemented with Python 3.8.

5.2. Benchmark effectiveness evaluation

The effectiveness of an algorithm plays a critical role in identifying anomalies accurately from data streams. As introduced in Section 4.5, we use point-aware and series-aware $Precision^{P/S}$, $Recall^{P/S}$ and $F_1^{P/S}$ as criteria to evaluate the effectiveness of various algorithms.

Table 4 shows the details of evaluation results. We conduct effectiveness experiments on all algorithms on univariate datasets, including AIOPS_KPI, MICRO, AWSCloudWatch, and GAIA. We also test the multivariate algorithms on the SMD dataset. The results indicate that no algorithm consistently exhibits high performance across all datasets, and the effectivenesses varies significantly. For instance, the Z-Score algorithm has the highest F_1^P score on the MICRO dataset but performs poorly on other datasets. Overall, xStream has a great $Recall$, it wins seven times out of ten experiments, which indicates that it can alert most true anomalies. On the other hand, the $Precision$ of SPOT ranks first, it wins six times out of ten experiments, indicating that most of the alerts generated by SPOT are true anomalies.

Additionally, point-aware evaluation and series-aware evaluation lead to significantly different results on the MICRO dataset. This is attributed to the short and concentrated anomaly duration of the MICRO dataset, as its average length is 228 and the anomaly rate is 1.26%. The experiments on this dataset demonstrate the differences between point-aware and series-aware evaluations.

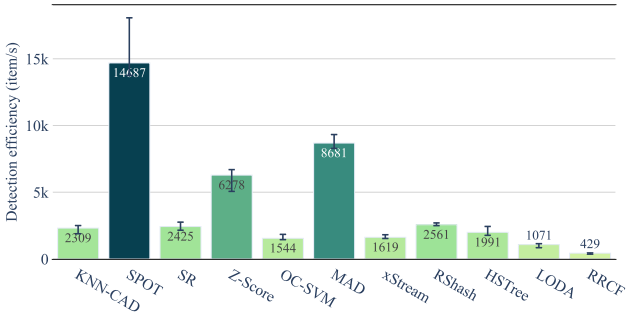
According to the experimental results, it is noteworthy that there is still a considerable scope for effectiveness improvement in these algorithms. The limitations of logical designs of the algorithms impact their performance, and some known flaws [7] in the existing datasets, such as unrealistic anomaly density and mislabeled ground truth, also have an unignorable impact.

In summary, selecting the best algorithm to detect cloud platform metrics anomalies is a challenging task. It is difficult to guarantee that an algorithm can cover all application scenarios. Users still need to try it out based on their specific use cases. As the promising effectiveness from our benchmark results, we recommend using SPOT for univariate data streams and xStream for multivariate data streams as the first attempt method.

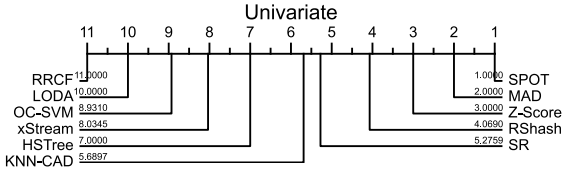
Table 4
Effectiveness comparison of anomaly detection algorithms across various datasets in StreamAD.

Datasets	AIOPS_KPI			MICRO			AWSCloudWatch			GAIA			SMD		
							Point-aware evaluation								
Criteria/Algorithms	P^P	R^P	F_1^P	P^P	R^P	F_1^P	P^P	R^P	F_1^P	P^P	R^P	F_1^P	P^P	R^P	F_1^P
KNN-CAD	0.24	0.24	0.19	0.52	0.62	0.54	0.04	0.66	0.06	0.21	0.40	0.18			
SPOT	**0.44**	0.06	0.08	0.32	0.39	0.33	**0.07**	0.70	**0.12**	**0.37**	0.47	**0.29**			
SR	0.10	0.17	0.11	0.24	0.36	0.27	0.02	0.76	0.04	0.10	0.56	0.11			
Z-Score	0.23	0.15	0.13	**0.62**	0.56	**0.58**	0.04	**0.79**	0.07	0.10	0.63	0.11		–	
OC-SVM	0.13	0.26	0.14	0.50	0.63	0.54	0.02	0.77	0.03	0.13	0.68	0.15			
MAD	0.32	0.24	**0.22**	0.58	0.54	0.56	0.04	0.72	0.06	0.23	0.67	0.23			
xStream	0.05	**0.27**	0.06	0.11	**0.79**	0.17	0.01	**0.79**	0.01	0.01	**0.75**	0.01	0.04	0.21	0.06
RShash	0.18	0.18	0.15	0.06	0.39	0.08	0.02	0.75	0.04	0.14	0.63	0.17	0.06	0.02	0.02
HSTree	0.14	0.16	0.11	0.04	0.17	0.06	0.01	0.50	0.01	0.12	0.72	0.08	**0.19**	**0.10**	**0.11**
LODA	0.07	0.19	0.08	0.45	0.50	0.47	0.02	0.40	0.03	0.09	0.44	0.09	0.09	0.04	0.05
RRCF	0.14	0.13	0.11	0.56	0.39	0.33	0.04	0.73	0.06	0.12	0.55	0.12	0.13	0.05	0.06
							Series-aware evaluation								
Criteria/Algorithms	P^S	R^S	F_1^S	P^S	R^S	F_1^S	P^S	R^S	F_1^S	P^S	R^S	F_1^S	P^S	R^S	F_1^S
KNN-CAD	0.17	**0.33**	0.18	0.88	0.89	0.87	0.02	0.66	0.04	0.21	0.44	0.18			
SPOT	**0.43**	0.09	0.13	0.84	0.86	0.85	**0.06**	0.70	**0.11**	**0.37**	0.49	**0.28**			
SR	0.08	0.31	0.12	0.82	0.84	0.82	0.02	0.76	0.03	0.08	0.62	0.09			
Z-Score	0.18	0.23	0.15	0.89	0.89	0.89	0.04	**0.79**	0.06	0.08	0.73	0.09		–	
OC-SVM	0.07	**0.33**	0.09	0.86	0.91	0.87	0.01	0.77	0.02	0.10	0.76	0.12			
MAD	0.25	0.25	**0.21**	**0.93**	**0.92**	**0.93**	0.03	0.72	0.05	0.18	0.70	0.18			
xStream	0.03	0.23	0.05	0.67	**0.92**	0.70	0.01	**0.79**	0.01	0.04	0.74	0.02	0.04	**0.14**	0.05
RShash	0.15	0.22	0.15	0.70	0.81	0.71	0.02	0.75	0.04	0.14	0.67	0.17	0.05	0.01	0.01
HSTree	0.06	0.07	0.01	0.89	0.90	0.89	0.01	0.50	0.01	0.10	**0.77**	0.07	**0.15**	0.05	0.06
LODA	0.06	0.10	0.06	0.89	0.91	0.90	0.02	0.40	0.03	0.08	0.41	0.06	0.08	0.05	0.06
RRCF	0.10	0.27	0.12	0.88	0.85	0.82	0.03	0.73	0.04	0.10	0.61	0.11	0.10	0.12	**0.09**

* P and R are abbreviations for *Precision* and *Recall*, respectively.
** Not conduct univariate algorithms experiments for multivariate data streams.

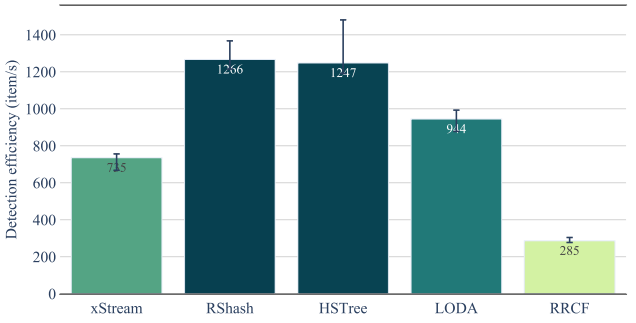


(a) Comparison of throughput rate for various algorithms on univariate data streams (higher value is better)

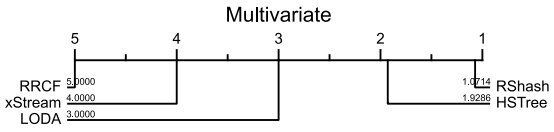


(b) Average rank for various algorithms on univariate data streams (lower rank is better)

Fig. 7. Efficiency comparison for eleven algorithms on univariate data streams.



(a) Comparison of throughput rate for various algorithms on multivariate data streams (higher value is better)



(b) Average rank for various algorithms on multivariate data streams (lower rank is better)

Fig. 8. Efficiency comparison for five algorithms on multivariate data streams.

5.3. Benchmark efficiency evaluation

For online anomaly detection in cloud platform, there are numerous metrics to be monitored. It is crucial to consider the efficiency of the detection methods in terms of the time required to respond to anomalous events, i.e., the execution time. The efficiency comparison provides a valuable assessment for the performance of various detection algorithms, which can help users to find.

Although different experimental environments, especially different computational resources, can greatly affect the implementation efficiency of algorithms, we believe that the horizontal comparison of different algorithms on the same experimental platform still has great significance. It allows us to intuitively compare the efficiency advantages and disadvantages of different algorithms.

In order to evaluate the efficiency of various algorithms across a range of existing datasets, we evaluate the number of detected streaming data per second for each algorithm, which provides us with an insight into the throughput rate. A higher throughput rate is generally

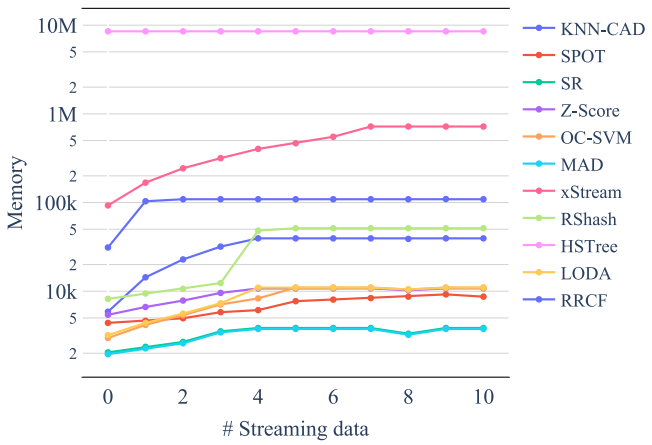


Fig. 9. Memory resource usage during online detection.

indicative of a more efficient algorithm, which is capable of addressing the anomaly detection task in large-scale cloud platforms.

Fig. 7 presents the efficiency comparison results of eleven algorithms for univariate data streams. In Fig. 7(a), we list the average throughout rate with the upper and lower bounds for various algorithms. It can be observed that SPOT outperforms other algorithms in this efficiency experiment, processing 14,687 data points per second. This exceptional performance can be attributed to its underlying assumption that most of the data is normal and does not require model fitting. Conversely, RRCF demonstrates the slowest performance among these algorithms due to its requirement to adjust each basic unit, i.e., random cut tree, when new streaming data arrives. It is a time-consuming process that slows down the overall detection performance. In this efficiency experiment, the fastest algorithm displays a significant performance advantage, processing data over 30 times faster than the slowest algorithm. The efficiency of the other algorithms varies, and their rankings based on performance are summarized in Fig. 7(b).

Additionally, we conducted the efficiency evaluation on multivariate datasets, as illustrated in Fig. 8. Among these algorithms, RShash and HSTree demonstrate similar efficiency, which can process more than one thousand points per second. Compared to Fig. 7(a), we observed that even when employing the same algorithm, processing multivariate data streams is slower than detection in univariate data streams.

Upon analyzing the results of all efficiency experiments, it can be observed that even the worst-performing algorithm, RRCF, is capable of detecting hundreds of points per second. Although a higher throughput rate generally signifies better performance, it is essential to consider the context of the practical application. For cloud platform anomaly detection, each metric requires a corresponding detection model instance, and the typical collection interval is 30 s per point. In this context, all algorithms in StreamAD, including the least efficient ones, can effectively satisfy the requirements regarding detection efficiency, ensuring timely anomaly detection and response in real-world cloud platform monitoring scenarios.

5.4. Memory limitation evaluation

As cloud platform metrics should be detected in real-time, the anomaly detection algorithm needs to be deployed and run for a long time. It is essential to ensure that the memory resources required by the algorithm do not increase continuously with data streaming, i.e., the memory resources should have a static limitation.

To evaluate the memory usage of various algorithms, we record the memory usage for the first one hundred data points under a univariate data stream. The results in Fig. 9 demonstrate that all the algorithms in StreamAD do not enlarge the occupied space after their initialization. Among all these algorithms, the tree-based method HSTree consumes the highest amount of memory. This can be attributed to the initialization of a tree by the algorithm, which arranges the historical stream data within it, and the size of the tree impacts the memory consumption of HSTree. We also find that the MAD algorithm has the lowest memory requirement. This is because MAD only needs to keep statistical information of the historical data streams without retaining all the records, which makes it more lightweight than others.

Thus, we believe that they can all comply with the memory limitation requirements for online applications. These results encourage the practical application of anomaly detection algorithms on cloud platforms, without a worry about the memory consumption.

6. Future work

With the development of the online anomaly detection community, the construction of benchmarks is a long-term process. In our future work, we are going to follow the state-of-the-art work, and integrate them into StreamAD. In addition, we plan to evaluate benchmarks from more perspectives, such as the impact of hyperparameters on the detection performance of different algorithms, and the interpretability of various algorithms. We hope that these future work can provide a more comprehensive view for benchmark evaluation.

7. Conclusion

In this work, we propose StreamAD, a cloud metrics-oriented benchmark for unsupervised online anomaly detection. StreamAD comprises eleven anomaly detection algorithms and conducts comprehensive experiments on five existing public datasets. The benchmark includes comparisons for the effectiveness, efficiency and memory resource consumption for various algorithms. StreamAD is open-source, and it provides a user-friendly API to help SREs evaluate anomaly detection applications in their specific use cases. Researchers can even develop new algorithms with StreamAD, which can facilitate further research in this area.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by National Key R&D Program of China: 2020YFB1805603, and CAS-Austria Joint Project (171111KYSB20200001).

References

[1] E. Bisong, An overview of Google cloud platform services, in: Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners, A Press, 2019, pp. 7–10.

[2] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, Q. Zhang, Time-series anomaly detection service at microsoft, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, 2019, pp. 3009–3017.

[3] D.T. Shipmon, J.M. Gurevitch, P.M. Piselli, S.T. Edwards, Time series anomaly detection; Detection of anomalous drops with limited features and sparse examples in noisy highly periodic data, 2017, arXiv arXiv:1708.03665.

[4] D. Sun, M. Fu, L. Zhu, G. Li, Q. Lu, Non-intrusive anomaly detection with streaming performance metrics and logs for DevOps in public clouds: A case study in AWS, IEEE Trans. Emerg. Top. Comput. 4 (2016) 278–289.

[5] J. Gao, X. Song, Q. Wen, P. Wang, L. Sun, H. Xu, RobustTAD: Robust time series anomaly detection via decomposition and convolutional neural networks, 2021, arXiv, arXiv:2002.09545.

[6] Q. Cheng, D. Sahoo, A. Saha, W. Yang, C. Liu, G. Woo, M. Singh, S. Saverese, S.C.H. Hoi, AI for IT operations (AIOps) on cloud platforms: Reviews, opportunities and challenges, 2023, arXiv, arXiv:2304.04661.
[7] R. Wu, E.J. Keogh, Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress, IEEE Trans. Knowl. Data Eng. 35 (2023) 2421–2429.
[8] Z. Li, N. Zhao, S. Zhang, Y. Sun, P. Chen, X. Wen, M. Ma, D. Pei, Constructing large-scale real-world benchmark datasets for AIOps, 2022, arXiv, arXiv:2208.03938.
[9] M. Ma, S. Zhang, D. Pei, X. Huang, H. Dai, Robust and rapid adaption for concept drift in software system anomaly detection, in: 2018 IEEE 29th International Symposium on Software Reliability Engineering, ISSRE, 2018, pp. 13–24.
[10] Z. Li, Y. Zhao, J. Han, Y. Su, R. Jiao, X. Wen, D. Pei, Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, 2021, pp. 3220–3230.
[11] S. Han, X. Hu, H. Huang, M. Jiang, Y. Zhao, ADBench: Anomaly detection benchmark, Adv. Neural Inf. Process. Syst. 35 (2022) 32142–32159.
[12] J. Paparrizos, Y. Kang, P. Boniol, R.S. Tsay, T. Palpanas, M.J. Franklin, TSB-UAD: An end-to-end benchmark suite for univariate time-series anomaly detection, Proc. VLDB Endowment 15 (2022) 1697–1711.
[13] K.-H. Lai, D. Zha, G. Wang, J. Xu, Y. Zhao, D. Kumar, Y. Chen, P. Zumkhawaka, M. Wan, D. Martinez, X. Hu, TODS: An automated time series outlier detection system, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 16060–16062.
[14] M. Pelc, Y. Khoma, V. Khoma, ECG signal as robust and reliable biometric marker: Datasets and algorithms comparison, Sensors 19 (2019) 2350.
[15] J. Wilkins, P. Seetharaman, A. Wahl, B. Pardo, Vocalset: a singing voice dataset, 2018.
[16] Google Cloud metrics, https://cloud.google.com/monitoring/api/metrics_gcp.
[17] B. Rabenstein, J. Volz, Prometheus: A Next-Generation Monitoring System (Talk), USENIX Association, 2015.
[18] Apache SkyWalking, https://skywalking.apache.org/.
[19] K.-H. Lai, D. Zha, J. Xu, Y. Zhao, G. Wang, X. Hu, Revisiting time series outlier detection: Definitions and benchmarks, in: Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1), 2022.
[20] A. Lavin, S. Ahmad, Evaluating real-time anomaly detection algorithms– the numenta anomaly benchmark, in: 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA, 2015, pp. 38–44.
[21] N. Singh, C. Olinsky, Demystifying numenta anomaly benchmark, in: 2017 International Joint Conference on Neural Networks, IJCNN, 2017, pp. 1570–1577.
[22] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, N. Tatbul, Exathlon: A benchmark for explainable anomaly detection over time series, Proc. VLDB Endowment 14 (2021) 2613–2626.
[23] D. Muhr, M. Affenzeller, Outlier/anomaly detection of univariate time series: A dataset collection and benchmark, in: Big Data Analytics and Knowledge Discovery, Springer International Publishing, 2022, pp. 163–169.
[24] S. Agrawal, J. Agrawal, Survey on anomaly detection using data mining techniques, Procedia Comput. Sci. 60 (2015) 708–713.
[25] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (2009) 1–58.
[26] A. Boukerche, L. Zheng, O. Alfandi, Outlier detection: Methods, models, and classification, ACM Comput. Surv. 53 (2021) 1–37.
[27] M. Gupta, J. Gao, C.C. Aggarwal, J. Han, Outlier detection for temporal data: A survey, IEEE Trans. Knowl. Data Eng. 26 (2014) 2250–2267.
[28] E. Jacobsen, R. Lyons, The sliding DFT, IEEE Signal Process. Mag. 20 (2003) 74–80.
[29] E. Jacobsen, R. Lyons, An update to the sliding DFT, IEEE Signal Process. Mag. 21 (2004) 110–111.
[30] A. Chauhan, K.M. Singh, Recursive sliding DFT algorithms: A review, Digit. Signal Process. 127 (2022) 103560.
[31] E. Burnaev, V. Ishimtsev, Conformalized density- and distance-based anomaly detection in time-series data, 2016, arXiv, arXiv:1608.04585.
[32] A. Siffer, P.-A. Fouque, A. Termier, C. Largouet, Anomaly detection in streams with extreme value theory, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, 2017, pp. 1067–1075.
[33] Standard score, Wikipedia (2023).
[34] Y. Zhang, N. Meratnia, P. Havinga, Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks, in: 2009 International Conference on Advanced Information Networking and Applications Workshops, 2009, pp. 990–995.
[35] A. Dotis-Georgiou, Anomaly detection with median absolute deviation, in: InfluxData.
[36] E. Manzoor, H. Lamba, L. Akoglu, Xstream: Outlier detection in feature-evolving data streams, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, 2018, pp. 1963–1972.
[37] S. Sathe, C.C. Aggarwal, Subspace outlier detection in linear time with randomized hashing, in: 2016 IEEE 16th International Conference on Data Mining, ICDM, 2016, pp. 459–468.
[38] S.C. Tan, K.M. Ting, T.F. Liu, Fast Anomaly Detection for Streaming Data.
[39] T. Pevný, Loda: Lightweight on-line detector of anomalies, Mach. Learn. 102 (2016) 275–304.
[40] S. Guha, N. Mishra, G. Roy, O. Schrijvers, Robust Random Cut Forest Based Anomaly Detection On Streams.
[41] K.M. Ting, G.-T. Zhou, F.T. Liu, J.S.C. Tan, Mass estimation and its applications, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, 2010, pp. 989–998.
[42] E. Schubert, M. Gertz, Numerically stable parallel computation of (Co-)variance, in: Proceedings of the 30th International Conference on Scientific and Statistical Database Management, ACM, 2018, pp. 1–12.
[43] B.P. Welford, Note on a method for calculating corrected sums of squares and products, Technometrics 4 (1962) 419–420.
[44] GAIA-DataSet/Companion_Data at Main · CloudWise-OpenSource/GAIA-DataSet, GitHub.
[45] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2019, pp. 2828–2837.
[46] N. Tatbul, T.J. Lee, S. Zdonik, M. Alam, J. Gottschlich, Precision and recall for time series, in: Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018.
[47] W.-S. Hwang, J.-H. Yun, J. Kim, H.C. Kim, Time-series aware precision and recall for anomaly detection: Considering variety of detection result and addressing ambiguous labeling, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, ACM, 2019, pp. 2241–2244.
[48] W.-S. Hwang, J.-H. Yun, J. Kim, B.G. Min, "Do you know existing accuracy metrics overrate time-series anomaly detections?", in: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, ACM, 2022, pp. 403–412.