HAL
open science

Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series

Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, Nesime Tatbul

To cite this version:

Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, et al.. Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series. Proceedings of the VLDB Endowment (PVLDB), 2021. hal-03381732

HAL Id: hal-03381732

https://hal.science/hal-03381732v1

Submitted on 17 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series

Vincent Jacob† Fei Song† Arnaud Stiegler† Bijan Rad† Yanlei Diao† Nesime Tatbul*
†Ecole Polytechnique, France *Intel Labs and MIT, USA
{vincent.jacob,fei.song,arnaud.stiegler,bijan.rad,yanlei.diao}@polytechnique.edu,tatbul@csail.mit.edu

ABSTRACT

Access to high-quality data repositories and benchmarks have been instrumental in advancing the state of the art in many experimental research domains. While advanced analytics tasks over time series data have been gaining lots of attention, lack of such community resources severely limits scientific progress. In this paper, we present Exathlon, the first comprehensive public benchmark for explainable anomaly detection over high-dimensional time series data. Exathlon has been systematically constructed based on real data traces from repeated executions of large-scale stream processing jobs on an Apache Spark cluster. Some of these executions were intentionally disturbed by introducing instances of six different types of anomalous events (e.g., misbehaving inputs, resource contention, process failures). For each of the anomaly instances, ground truth labels for the root cause interval as well as those for the extended effect interval are provided, supporting the development and evaluation of a wide range of anomaly detection (AD) and explanation discovery (ED) tasks. We demonstrate the practical utility of Exathlon's dataset, evaluation methodology, and end-to-end data science pipeline design through an experimental study with three state-of-the-art AD and ED techniques.

1 INTRODUCTION

Time series is one of the most ubiquitous types of data in our increasingly digital and connected society. Advanced analytics capabilities such as detecting anomalies and explaining them are crucial in understanding and reacting to temporal phenomena captured by this rich data type. *Anomaly detection* (AD) refers to the task of identifying patterns in data that deviate from a given notion of normal behavior [13]. It finds use in almost every domain where data is plenty, but unusual patterns are the most critical to respond (e.g., cloud telemetry, autonomous driving, financial fraud management). AD over time series data has been of particular interest, not only because time-oriented data is highly prevalent and voluminous, but also more challenging to analyze due to its complex and diverse nature: multivariate time series can consist of 1000s of dimensions; anomalous patterns may be of arbitrary length and shape; there may be intricate cause and effect relationships among these patterns; data is rarely clean. Furthermore, by helping uncover how or why a detected anomaly may have happened, *explanation discovery* (ED) forms a crucial capability for any time series AD system.

Recent advances in data science and machine learning (ML) significantly reinforced the need for developing robust anomaly detection and explanation solutions that can be reliably deployed in production environments [30, 38]. However, progress has been rather slow and limited. While there is extensive research activity going on [12], proposed solutions have been mostly adhoc and far from being generalizable to realistic settings. We believe that one of the critical roadblocks to progress has been the lack of open data repositories and benchmarks to serve as a common ground for reproducible research and experimentation. Indeed, access to such community resources has been instrumental in advancing the state of the art in many other domains (e.g., [4, 19]). Inspired by those efforts, in this paper, we propose *Exathlon, the first comprehensive public benchmark for explainable anomaly detection over high-dimensional time series data.*

Exathlon focuses on the familiar domain of metric monitoring in large-scale computing systems, and provides a benchmarking platform that consists of: (i) a curated anomaly dataset, (ii) a novel benchmarking methodology for AD and ED, and (iii) an end-to-end data science pipeline for implementing and evaluating AD and ED algorithms based on the provided dataset and methodology. More specifically, we make the following contributions in this work:

Dataset. We constructed Exathlon systematically based on real data traces collected from around 100 repeated executions of 10 distributed streaming jobs on a Spark cluster over 2.5 months. Inspired by chaos engineering in industry [5], our traces were obtained by disturbing more than 30 job executions with nearly 100 instances of 6 different classes of anomalous events (e.g., misbehaving inputs, resource contention, process failures). For each of these anomalies, we provide ground truth labels for both the root cause interval and the corresponding effect interval, enabling the use of our dataset in a wide range of AD and ED tasks. Overall, both the normal (*undisturbed*) and anomalous (*disturbed*) traces contain enough variety (including some noise due to Spark's inherent behavior) to capture real-world data characteristics in this domain (Table 1).

Evaluation Methodology. Exathlon evaluates AD and ED algorithms in terms of two orthogonal aspects: functionality and computational performance. For AD, we primarily target *semi-supervised techniques* (i.e., with a model developed/trained using only normal data, possibly with occasional noise, and then tested against anomalous data) for *range-based anomalies* (i.e., contextual and collective anomalies occurring over a time interval instead of only at a single time point [13]) over *high-dimensional time series* (i.e., multivariate with 1000s of dimensions). This decision is informed by our observation of this being the most common and inclusive usage scenario in practice. For ED, we broadly consider both *model-free* (e.g., [3, 58]) and *model-dependent* (e.g., [44]) techniques. AD functionality is evaluated under four well-defined model learning settings, based on four key evaluation criteria – anomaly existence, range detection, early detection, and exactly-once detection – using a novel range-based accuracy framework [51]. Similarly, ED functionality is tested for two capabilities – local explanation and

global explanation – each measured in terms of conciseness, consistency, and accuracy. Computational efficiency and scalability for both AD model training/inference as well as for ED execution can also be evaluated at varying data dimensions and sampling rates. Overall, Exathlon provides a rich and challenging testbed with a well-organized evaluation methodology (Table 2).

Data Science Pipeline. We designed an end-to-end pipeline for explainable time series anomaly detection. This pipeline includes all the data processing steps necessary to turn our raw datasets into AD and ED results together with their benchmark scores. Our design is modular and extensible. This not only makes it easy to implement new AD and ED techniques to benchmark, but also allows creating multiple variants of pipeline steps to experiment with and compare. For example, training data preparation for different AD learning settings or scoring AD results for different criteria levels can be easily configured, run, and compared in our pipeline (Figure 3).

Experimental Study. We provide the first experimental study evaluating a representative set of state-of-the-art AD and ED techniques to illustrate the usage and benefits of Exathlon. Results suggest that our dataset carries useful signals that can be picked up by the tested AD and ED algorithms in a way that can be effectively quantified by our evaluation criteria and metrics. Furthermore, we observe that our benchmarking framework exposes increasing levels of challenges to stress-test these algorithms in a systematic way (§6).

Compared to current public resources for time series AD research [2, 18, 20, 34, 43], a key contribution of Exathlon is that it comprehensively covers one challenging application domain end to end, as opposed to providing multiple smaller and simpler datasets from several independent domains. Furthermore, a public benchmark for time series ED research with ground truth labels is largely lacking today, making it hard to evaluate and compare an increasing number of published papers on this important topic. Thus, we believe Exathlon provides an opportunity for a more in-depth investigation and evaluation of models and algorithms in both time series AD and ED, potentially revealing new insights for accelerating research progress in explainable anomaly detection.

In the rest of the paper, we first briefly summarize related work. After presenting our dataset, evaluation methodology, and pipeline design in more detail, we show the practical utility of Exathlon through an experimental analysis of three state-of-the-art AD and ED algorithms using a selected set of evaluation criteria and settings from our benchmark. Finally, we conclude with an outline of future directions. The dataset, code, and documentation for Exathlon are publicly available at https://github.com/exathlonbenchmark/exathlon.

2 RELATED WORK

Datasets and Benchmarks. Benchmarks to evaluate database (DB) system performance have been around for more than 30 years [16, 17, 26]. In addition to industry-standard benchmarks for relational DB workloads such as TPC-C and TPC-H, new domain-specific benchmarks for emerging workloads have been proposed (e.g., Linear Road Benchmark for stream processing [1], YCSB for scalable key-value stores [15], BigBench for big data analytics [25]). The main focus of these benchmarks has been on computational performance. With recent benchmarks for ML/DL-based advanced data analytics such as ADABench and DAWNBench, there has been a focus shift toward end-to-end ML pipelines and new evaluation metrics such as time to accuracy [14, 42]. Like in DB and systems communities, the ML community has also been publishing datasets and benchmarks to support research in many problem domains from object recognition to natural language processing [4, 19, 40, 53]. Well-known data archives for time series research include: UCI [20], UCR [18], and UEA [2]. These archives provide real-world data collections created for general ML tasks, such as classification and clustering. While the need for systematically constructing AD benchmarks from real data has also been recognized by others [22], public availability of anomaly datasets is still limited [43]. To our knowledge, Numenta Anomaly Benchmark (NAB) is the only public benchmark designed for time series AD [34]. NAB provides 50+ real and artificial datasets, primarily focusing on real-time AD for streaming data. Compared to ours, each of these datasets is much smaller in scale and dimensionality, and does not capture any information to enable ED. NAB also has several technical weaknesses that hinder its use in practice (e.g., ambiguities in its scoring function, missing values in its datasets) [48].

Anomaly Detection (AD). There is a long history of research in AD [13, 27]. The high degree of diversity in data characteristics, anomaly types, and application domains has led to a plethora of AD approaches from simple statistical methods [7] to distance-based [52], density-based [11, 13], and isolation forests [36] to deep learning (DL) methods [12]. It is beyond the scope of this paper to provide a complete survey; we refer the reader to recent survey papers for a full discussion of such methods [12, 13, 27]. In our experimental study, we particularly focus on three DL methods that represent the recent state of the art [39, 47, 56] (detailed in §6). Such DL methods have the potential to handle a variety of anomaly patterns, such as contextual and collective patterns [13], and overcome known limitations of previous density- and distance-based methods that are very sensitive to data dimensions.

Interpretable Machine Learning. Interpretable ML has recently attracted a lot of attention [41]. Relevant techniques generally belong to two broad families: *interpretable models* and *model-agnostic methods*. Interpretable models directly build a human-readable model from the data (e.g., linear or logistic regression, decision trees or rules) [41]. In contrast, model-agnostic methods separate explanations from the ML model, hence offering the flexibility to mix and match ML models with interpretation methods. In the model-agnostic family, several methods obtain interpretable classifiers by perturbing the inputs and observing the response [33, 44, 45, 50]. LIME [44] explains a prediction of any classifier by approximating it locally with an interpretable sparse linear model, and explains the overall model by selecting a set of representative instances with explanations. As such, it is generally considered a method for *local explanations*. We evaluate LIME in our experimental study. Anchors [45] improved upon LIME by replacing its linear model with a logical rule for explaining a data instance. It offers better coverage of data points in a local neighborhood, but does not support time series data. SHAP scores [37], RESP scores [6], and axiomatic attribution [50] are also instance-level explanations that assign a numerical score to each feature, representing their importance in the outcome. In contrast to local explanations, other work aims to explain a model via *global explanations*. Some of them approximate a DL model using a decision tree [24, 55], or by learning a decision

set [32, 33] directly as explainable models. All of these methods suffer from lacking a benchmark dataset and evaluation methodology. The FICO challenge was designed to evaluate such methods using a home loan application dataset, with a known label (high or low risk) for each application, but it relies on manual evaluation of returned explanations by real-world data scientists [23]. As a result, it remains hard to compare different ED methods due to the lack of ground truth explanations and automated evaluation procedures.

Explaining Outliers in Data Streams. There is a handful of work in explaining outliers in data streams. Given normal and abnormal time periods by the user, EXstream finds explanations to best distinguish the abnormal periods from the normal ones [58]. MacroBase helps the user prioritize attention over data streams, with modules for both AD and ED tasks [3]. Its AD module uses simple statistical methods like MAD, which is known to be suitable only for detecting simple point outliers [13]. For a detected anomaly, MacroBase's ED module discovers an explanation in the form of conjunctive predicates, by using a frequent itemset mining framework that takes minimum support and risk ratio as input parameters. We evaluate both of these techniques in our experimental study.

Explaining Outliers in SQL Query Results. Scorpion explains outliers in group-by aggregate queries by searching through various subsets of the tuples that were used to compute the query answers [54]. Given a set of explanation templates by the user, Roy et al.'s approach performs precomputation in a given DB to enable interactive ED [46]. Similarly, given a table, El Gebaly et al.'s work constructs an explanation table and finds patterns affecting a binary value of each tuple [21]. These approaches target traditional DB workloads and are not applicable to our problem. There have been recent industrial efforts on time series anomaly explanation and root cause analysis in DB systems [30, 38]. These approaches require a variety of inputs from the user, e.g., causal hypotheses [30] or labels of root causes [38], whereas our work focuses on semi-supervised learning for explainable AD. Moreover, these systems are largely based on proprietary code and datasets that are not accessible to the research community.

3 DATASET

The Exathlon dataset has been systematically constructed based on real data traces collected from a use case scenario that we implemented on Apache Spark. In this section, we first describe this scenario, followed by the details of how we created the normal and anomalous data traces themselves.

3.1 Use Case: Spark Application Monitoring

Large-scale data analytics applications are deployed on Apache Spark clusters everyday. Monitoring the execution of these jobs to ensure their correct and timely completion via AD can be business-critical. For example, some of the largest e-commerce platforms run Spark jobs on petabytes of data each day to analyze purchase patterns, target offers, and enhance customer experiences [49]. Since results of these jobs affect immediate business decisions such as inventory management and sales strategies, they are often specified with deadlines. Anomalies that occur in job execution prevent analytical jobs from meeting their deadlines and hence cause disruption to critical operations on those e-commerce platforms. We model this widespread and challenging AD use case in our benchmark.

System Setup. Our Spark workload consists of 10 stream processing applications (detailed in our technical report [29]), analyzing user click streams from the WorldCup 1998 website [35]. As in Figure 1(a), *Data Sender* servers send streams at a controlled *input rate* to a Spark cluster of 4 nodes, each with 2 Intel® Xeon® Gold 6130 16-core processors, 768GB of memory, and 64TB disk. Each application has certain workload characteristics (e.g., CPU or I/O intensive) and is executed by Spark in a distributed manner, as in Figure 1(b). Submitted an application, Spark launches a *Driver* process to coordinate the execution. The driver connects to a resource manager (Apache Hadoop YARN), which launches *Executor* processes on a subset of nodes where tasks (units of work on a data partition, e.g., `map` or `reduce`) will be executed in parallel. Given 32 cores, each node can also run tasks from multiple applications concurrently. As common real-world practice, we run 5/10 randomly selected applications at a time. The placement of Driver and Executor processes to cluster nodes is decided by YARN based on data locality, load on nodes, etc. Except for I/O activities, YARN offers container isolation for resource usage of all parallel processes.

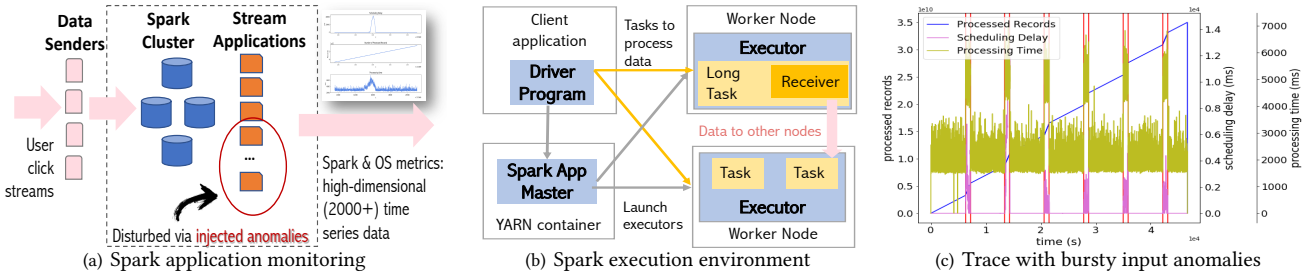
Trace Collection. We ran the 10 Spark streaming applications in our 4-node cluster over a 2.5-month period. The data collected from each run of a Spark streaming application is called a *Trace*. Some of the traces were manually pruned, because they were affected by cluster downtimes or the injected anomalies were not well reflected in the data due to failed attempts. After this manual pruning, we kept 93 traces to constitute the Exathlon dataset.

Metrics Collected. During each application execution, we collected metrics from both the Spark Monitoring and Instrumentation Interface (UI) and underlying operating system (OS). Table 1(a) gives a summary of the metrics collected per trace. The Driver offers 243 Spark UI metrics covering scheduling delay, statistics on the streaming data received and processed, etc. Each executor provides 140 metrics on various time measurements, data sizes, network traffic, as well as memory and I/O activities. As we wanted to keep the number of metrics the same for all traces, we set a fixed limit of 5 for the number of Spark executors (3 active + 2 backup). This way, even if an active executor fails during a run and a backup takes over, the number of metrics collected stays the same, $5 \times 140 = 700$, with null values set for inactive executors. 335 OS metrics for each of the 4 cluster nodes were collected using the `Nmon` command, capturing CPU time, network traffic, memory usage, etc. All in all, each trace consists of a total of 2,283 metrics recorded each second for 7 hours on average, constituting a multi-dimensional time series.

3.2 Undisturbed vs. Disturbed Traces

In generating our traces, we followed an approach similar to chaos engineering (i.e., an approach devised by high-tech companies like Netflix for injecting failures and workload surges into a production system to verify/improve its reliability) [5] and general systems monitoring (e.g., Microsoft's NetMedic [31]). Thus, we first generated *undisturbed traces* to characterize the normal execution behavior of our Spark cluster; we then introduced various anomalous events to generate *disturbed traces*. Table 1(b) provides an overview.

Undisturbed Traces. Uninterrupted executions of 5/10 randomly selected applications at a time, at parameter settings within the capacity limits of our Spark cluster, over a period of 1 month, gave us 59 normal traces of 15.3GB in size. Any instances of occasional



(a) Spark application monitoring (b) Spark execution environment (c) Trace with bursty input anomalies

Figure 1: Spark application monitoring, and metrics observed in anomaly instances (a pair of red vertical bars marks a root cause event)

Metric Type	Spark UI Driver	Spark UI Executor	OS (Nmon)
# of Metrics	243	5 x 140 = 700	4 x 335 = 1340
Total	2,283		

Frequency	1 data item per second
Data Items	2,335,781
Duration	649 hours
Total Size	24.6 GB

(a) Metrics and data size

Trace Type	Anomaly Type	# of Traces	Anomaly Instances	Anomaly Length (RCI + EEI) min, avg, max	Data Items
Undisturbed	N/A	59	N/A	N/A	1.4M
Disturbed	T1: Bursty input	6	29	15m, 22m, 33m	360K
Disturbed	T2: Bursty input until crash	7	7	8m, 35m, 1.5h	31K
Disturbed	T3: Stalled input	4	16	14m, 16m, 16m	187K
Disturbed	T4: CPU contention	6	26	8m, 15m, 27m	181K
Disturbed	T5: Driver failure	11	9	1m, 1m, 1m	128K
Disturbed	T6: Executor failure		10	2m, 23m, 2.8h	
Ground Truth	(`app_id`, `trace_id`, `anomaly_type`, `root_cause_start`, `root_cause_end`, `extended_effect_start`, `extended_effect_end`)				

(b) Undisturbed traces, disturbed traces, and ground truth labels of 97 anomalies

Table 1: The Exathlon dataset

cluster downtime were manually removed from these traces. It is important to note that, although undisturbed, these traces still exhibit occasional variations in metrics due to Spark's inherent system mechanisms (e.g., checkpointing, CPU usage by a DataNode in the distributed file system). Since such variations do appear in almost every trace, we consider them as part of the normal system behavior. In other words, our normal data traces include some "noise", as most real-world datasets typically do.

Disturbed Traces. Disturbed traces were obtained by introducing anomalous events during an execution. Based on discussions with industry contacts from the Spark ecosystem, we came up with 6 types of anomalous events. When designing these, we considered that: (i) they lead to a visible effect in the trace, (ii) they do not lead to an instant crash of the application (since AD would be of little help in this case), (iii) they can be tracked back to their root causes. We briefly describe these anomalies below; please refer to our report for further details.

Bursty Input (Type 1): To mimic input rate spikes, we ran a disruptive event generator (DEG) on the Data Senders to temporarily increase the input rate by a given factor for a duration of 15-30 minutes. We repeated this pattern multiple times during a given trace, creating a total of 29 instances of this anomaly type over 6 different traces. Please see Figure 1(c) for an example.

Bursty Input Until Crash (Type 2): This is a longer version of Type 1 anomalies, where we let the DEG period last forever, crashing the executors due to lack of memory. When an executor crashes, Spark launches a replacement, but the sustained high rates keeps crashing the executors, until Spark eventually decides to kill the whole application. We injected this anomaly into 7 different traces.

Stalled Input (Type 3): This type of anomaly mimics failures of Spark data sources (e.g., Kafka or HDFS). To create it, we ran a DEG that set the input rates to 0 for about 15 minutes, and then periodically repeated this pattern every few hours, giving us a total of 16 anomaly instances across 4 different traces.

CPU Contention (Type 4): The YARN resource manager cannot prevent external programs from using the CPU cores that it has allocated to Spark processes, causing scheduling delays to build up due to CPU contention. We reproduced this anomaly using a DEG that ran Python programs to consume all CPU cores on a given Spark node. We created 26 such anomaly instances over 6 different traces.

Driver Failure (Type 5) and Executor Failure (Type 6): Hardware faults or maintenance operations may cause a node to fail all of a sudden, making all processes (drivers and/or executors) located on that node unreachable. Such processes must be restarted on another node, which causes delays. We created such anomalies by failing driver processes, where the number of processed records drops to 0 until the driver comes back up again in about 20 seconds. We also created anomalies by failing executor processes, which get restarted 10 seconds after the failure, but whose effects on metrics such as processing delay may continue longer. We created 9 driver failures and 10 executor failures over 11 different traces.

Ground Truth Table. For all of these 97 anomaly instances over 34 anomalous traces, we provide ground truth labels with the information shown in Table 1(b). Such labels include both *root cause intervals* (RCIs) and their respective *extended effect intervals* (EEIs). RCIs typically correspond to the time period during which DEG programs are running, whereas the EEIs are the time periods that start immediately after an RCI and end when important system metrics return to normal values or the application is eventually pushed to crash. The EEIs are manually determined using domain knowledge. Additional details can be found in our report.

4 BENCHMARK DESIGN

In this section, we present the evaluation methodology we designed to benchmark anomaly detection (AD) and explanation discovery

	Anomaly Detection (AD) Functionality	Explanation Discovery (ED) Functionality	Computational Performance
Evaluation Criteria	AD1: Anomaly Existence AD2: Range Detection AD3: Early Detection AD4: Exactly-Once Detection	ED1: Local Explanation ED2: Global Explanation	P1: AD Training Scalability P2: AD Inference Efficiency P3: ED Efficiency
Evaluation Metrics	Accuracy: *Range-based Precision, Recall, F-Score, AUPRC*	Conciseness Consistency: *Stability* (ED1), *Concordance* (ED2) Accuracy: *Point-based Precision, Recall, F-Score*	Time, given different *Dimensionality* and *Cardinality* factors

Table 2: The Exathlon evaluation methodology and benchmark design

(ED) algorithms based on the curated, high-dimensional time series dataset described in the previous section. As summarized in Table 2, Exathlon is designed to evaluate AD and ED algorithms in two orthogonal aspects, *functionality* and *computational performance*, using well-defined metrics. In terms of functionality, the evaluation criteria capture that an AD/ED algorithm is exposed to increasingly more challenging requirements as the functionality level is raised from one to the next. In terms of computational performance, Exathlon provides three complementary criteria that can be evaluated by varying dimensionality and size of the dataset.

4.1 Anomaly Detection (AD) Functionality

First and foremost, we designed Exathlon targeting *semi-supervised AD techniques* (i.e., trained only with normal data, possibly with occasional noise, and then tested against anomalous data) for *range-based anomalies* (i.e., contextual and collective anomalies occurring over a time interval instead of only at a single time point) over *high-dimensional* (i.e., multivariate with 1000s of dimensions) time series. This decision is informed by our observation of this being the most common and inclusive usage scenario in practice.

Evaluation Criteria. We identified four key criteria for evaluating AD functionality, listed below from basic towards advanced, where a higher AD level includes the requirements of all preceding levels:

AD1 (*Anomaly Existence*): The first expectation is to flag the existence of an anomaly somewhere within the *anomaly interval* (i.e., the root cause interval (RCI) + the extended effect interval (EEI)).

AD2 (*Range Detection*): The next expectation is to report not only the existence, but also the precise time range of an anomaly. The wider a range of an anomaly that an AD method can detect, the better its understanding of the underlying real-world phenomena.

AD3 (*Early Detection*): The third expectation is to minimize the *detection latency*, i.e., the difference between the time an anomaly is first flagged and the start time of the corresponding RCI.

AD4 (*Exactly-Once Detection*): The last expectation is to report each anomaly instance exactly once. Duplicate detections are undesirable, because they may not only redundantly cause repeated alerts for a single anomalous event, but also confusion if those alerts are for the same anomaly event or not.

Evaluation Metrics. To assess how well an AD algorithm can meet these four functionality levels, we use the customizable accuracy evaluation framework for time series [51]. This framework extends the classical precision/recall from point-based data to range-based data, by introducing a set of tunable parameters. By setting the values of these parameters in a particular way and applying the resulting precision/recall formulas to the output of an AD algorithm, one can assess how well that output measures up to the quality expectations represented by those parameter settings. We leverage

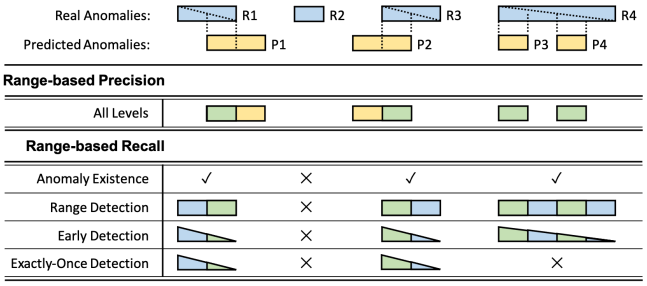


Figure 2: Range-based precision and recall at AD levels 1-4. Precision evaluates prediction quality (green out of yellow for each Pi). Recall evaluates anomaly coverage (green out of blue for each Ri).

this as a mathematical tool to quantify how well an AD algorithm meets AD1-AD4. Furthermore, we chose to do this in a way that every level AD_i builds on and adds to the requirements of the previous level AD_{i-1}. This monotonic design ensures that the AD functionality score that an algorithm gets (Precision, Recall, or other metrics obtained by combining them, e.g., F-Score or Area Under the Precision/Recall Curve (AUPRC)) is always ordered as: $score(AD1) \geq score(AD2) \geq score(AD3) \geq score(AD4)$, which facilitates evaluating and interpreting results in a systematic way. In Exathlon, we preferred this design over the alternative of treating each AD level as orthogonal to enable users to develop/perfect their models for tasks that become increasingly more challenging.

Figure 2 provides a simple example to illustrate how range-based precision and recall are computed for different AD levels. Given real anomaly ranges R1..R4 and predicted anomaly ranges P1..P4 produced by an AD algorithm, we first compute precision/recall for each range and then average them for overall precision/recall. Intuitively, precision focuses on the size of TP ranges (colored green) relative to TP+FP ranges (colored yellow), and recall focuses on the size of TP ranges relative to TP+FN ranges (colored blue). For AD1, Recall(Ri) is 1 if Ri is flagged, 0 otherwise. For AD2, Recall(Ri) is proportional to the relative size of the TP range. For AD3, Recall(Ri) is further weighted by position of the TP range relative to the start of Ri. Finally, at AD4, Recall(Ri) degrades to 0 for any Ri that is not flagged exactly once. Precision(Pi) is computed in an analogous way, except that AD levels about anomaly coverage quality (AD1 and AD3) are not relevant to it; rather, the main focus is on the size and number of the real anomaly ranges that are successfully predicted. In our simple example, it turns out that all AD levels for Precision consider the same Pi subranges.

To achieve precision and recall at different AD levels, we set the tunable parameters of the range-based precision/recall framework with necessary modifications. These details are deferred to our technical report due to space constraints [29]. Further note that

both the semi-supervised AD algorithms we investigate and the precision/recall for time series model we use to assess them focus on binary classification (normal vs. anomalous ranges). On the other hand, our dataset is inherently a multi-class one (normal ranges vs. six types of anomalous ranges). This raises a question about how to evaluate binary predictions under multi-class labels. We take a holistic approach, and evaluate the AD prediction results both globally and grouped by type, whenever this is reasonable and provides useful insights. For example, even though a binary predictor is not able to detect different anomaly types, we can still measure its resulting coverage (i.e., recall) for each type. However, type-wise measurement is not entirely meaningful for precision, since false positives (FPs) are essentially typeless.

Our benchmark also includes a set of four learning settings *LS1-LS4*, ranging from simple to more complex yet realistic ones. These are detailed in our technical report [29].

4.2 Explanation Discovery (ED) Functionality

Once an anomalous instance is flagged by an AD method, the next desirable functionality is to find the best explanation for the anomaly detected, or more precisely, a human-readable formula offering useful information about what has led to the anomaly.

There have been many ED methods in recent work (see §2). These differ in the form of "explanation" provided: some return a logical formula as an explanation [3, 45, 58], others return a decision tree [55], and some others return a numerical score for each feature such as the coefficient in linear regression [44] or the SHAP score [37]. Exathlon does not pose any restrictions on the form of explanation used. Instead, it takes an abstract view of explanations. Formally, we model each trace in the test dataset as a multi-dimensional time series, $[\ \boldsymbol{x}_1 \ldots\ \boldsymbol{x}_t \ldots\ \boldsymbol{x}_n]^T$, where each data item includes m features, $\boldsymbol{x}_t = (x_{t1}, \ldots, x_{tm})$. A detected anomaly is a subsequence of the time series that starts at timestamp t and has duration w, $X_{t,w} = [\ \boldsymbol{x}_t \ldots\ \boldsymbol{x}_{t+w}]^T$. If an AD method can provide only point-based detection, then w is set to 0. We denote the explanation generated for the anomaly $X_{t,w}$ as $F_{t,w}$ and treat it as a function of the features, $\mathbf{A} = (a_1, \ldots, a_m)$, from the data:

$$F_{t,w}(a_1, \ldots, a_m) \models X_{t,w}$$

where $\models$ means that $F_{t,w}$ "explains" the anomaly $X_{t,w}$. In addition, we define an extraction function, $G_{\mathbf{A}}$ over $F_{t,w}$, that returns the set of features used in the explanation (e.g., appearing in a logical formula or having non-zero coefficients in a regression model):

$$G_{\mathbf{A}}\left(F_{t,w}(a_1, \ldots, a_m)\right) = \mathbf{A}_{t,w} \subseteq \mathbf{A}$$

Finally, we define the size of $F_{t,w}$ as the size of its feature set $\mathbf{A}_{t,w}$:

$$|F_{t,w}(a_1, \ldots, a_m)| = |\mathbf{A}_{t,w}|$$

Evaluation Criteria: Subject of Explanation. The key distinction that Exathlon makes is whether an ED method is attempting to explain a single anomaly (*local*) or a broad set of anomalies (*global*).

ED1: *Local Explanation*: This corresponds to explaining one anomaly instance, offering a compact yet meaningful piece of information to help the user understand this particular instance. As mentioned by LIME [44], the explanation should be locally faithful. In our context, it means that the same explanation can hold over immediate "neighbors", which are anomaly instances of the same application and same anomalous type, and around the same time period.

ED2: *Global Explanation*: Alternatively, an ED method may attempt to explain a (potentially large) set of anomalies, called a global explanation. In general, it is not possible to find an identical succinct explanation for many different instances. Hence, a global explanation is usually composed of a set of explanations; e.g., LIME [44] chooses the most representative k instances to explain a model. In our benchmark, it makes most sense to construct a global model for a set of anomalies of the same type, but potentially from different applications or different runs of the same application. This helps us understand for "semantically similar" anomalies, whether an ED method can return explanations that are consistent, or even of predictive power of similar anomalies that arise in the future.

Evaluation Metrics. Exathlon evaluates both local and global explanations for three desired properties:

1. *Conciseness*: This corresponds to the number of features used in the explanation. Following the Occam's razor principle, humans favor smaller, and thus simpler explanations. As different ED methods return explanations of different forms, our benchmark counts the number of features used in the explanation as its conciseness measure. In the ED1 case, that is $|F_{t,w}|$ defined above. In the ED2 case, a global explanation includes a set of explanations, and its conciseness measure is the average of the size of each explanation.

2. *Consistency*: Anomalies of the same type occurring in a similar context should have consistent explanations. We customize this notion for ED1 and ED2, respectively. In both cases, we care only about the set of features employed in the explanation, without considering the numerical or categorical values used.

Stability (ED1) is the customized consistency measure for ED1. It means that the anomalies occurring in a similar context (e.g., for the same application, same run, and same time period) should have similar explanations, subject to a small perturbation of the data. Formally, we introduce a subsampling procedure over an anomaly $X_{t,w}$, which generates a set of samples, $\{X_{t,w}^{(i)}\}$. We denote the corresponding explanations generated for them as $\{F_{t,w}^{(i)}\}$. The extraction function for a set of explanations is defined to be the duplicate-preserving union (like Union All in SQL) of the extraction function of each respective explanation:

$$G_{\mathbf{A}}\left(\{F_{t,w}^{(i)}(a_1, \ldots, a_m)\}\right) = \bigcup_i G_{\mathbf{A}}\left(F_{t,w}^{(i)}\right) = \bigcup_i \mathbf{A}_{t,w}^{(i)} = \mathbf{A}_{t,w}^{\cup}$$

Finally, for each feature $a_j \in \mathbf{A}_{t,w}^{\cup}$, we count its frequency in this feature set and normalize it by the total size of the feature set. The consistency measure is then defined as the entropy of the set of normalized frequencies of such features, a_j, $j = 1, 2, \ldots$:

$$H\left(A_{t,w}^{\cup}\right) = -\sum_j p(a_j) \log_2 p(a_j), \quad a_j \in \mathbf{A}_{t,w}^{\cup}$$

$$p(a_j) = \mathbb{1}_{\mathbf{A}_{t,w}^{\cup}}(a_j)\ \ /\ \left|\mathbf{A}_{t,w}^{\cup}\right|$$

Where $\mathbb{1}_{\mathbf{A}_{t,w}^{\cup}}(a_j)$ is here an indicator function that counts the occurrences of a feature in a multiset. For capturing consistency, our choice of entropy is motivated by information theory that a set of explanations that lack consistency will require using more bits

to encode, hence a larger entropy value. In the ideal case, all explanations, $\{F^{(i)}_{t,w}\}$, are identical, and its entropy takes the minimum value 0 if the size of the explanation is 1 (denoted as H_1), the value 1 if the size is 2 (H_2), or the value 1.58 if the size is 3 (H_3).

Concordance (ED2) is the customized consistency measure for ED2. Here, it means that the anomalies of the same type are expected to have consistent explanations, subject to larger amounts of deviation in data due to different time periods in the same run of a Spark application, different runs of the application, or even different Spark applications. Formally, we are given a set of anomalies, $\{X_{t_i,w_i}\}$. Denote their corresponding explanations as $\{F_{t_i,w_i}\}$. The consistency measure of this set of explanations is computed similarly to ED1, except that we are replacing the subsampled anomalies, $\{X^{(i)}_{t,w}\}$, with the given set of anomalies, $\{X_{t_i,w_i}\}$.

However, one may notice that the conciseness measure also has an impact on consistency. To factor out this impact, we further define *Normalized Consistency* as $\frac{2^{Consistency}}{Conciseness}$, which captures the variability of the explanations conditioned on their average size.

3. *Accuracy*: The last property, which is also the hardest to achieve, is to view an explanation of an anomaly as a predictive model, apply it to other similar instances (defined above for ED1 and ED2, respectively), and then evaluate accuracy of such predictions.

Note that not all explanations can serve as a predictive model. Only those that are a function mapping a given data item to 0/1, $F_{t,w}: \boldsymbol{x_t} \in \mathbb{R}^m \rightarrow \{0,1\}$, can offer predictive power over test data. For example, a logical formula [3, 45, 58] or a decision tree [55] can be used to run prediction on new data items, but feature importance scores or SHAP scores [37] cannot. Even with those ED methods that return a predictive explanation, it is only a point-based predictive model. The literature largely lacks ED methods that can return explanations that characterize a temporal pattern. For this reason, Exathlon evaluates the accuracy of such explanations using point-based precision and recall.

In the case of ED1, we are given a particular anomaly $X_{t,w}$. To measure the accuracy of an ED method, we subsample from $X_{t,w}$, yielding a sample, $X^{(i)}_{t,w}$. We run the ED method to generate an explanation, $F^{(i)}_{t,w}$. Then we run $F^{(i)}_{t,w}$ as a predictive model over a test dataset that includes the remainder of the anomalous data, $X_{t,w} - X^{(i)}_{t,w}$, as well as some normal data that immediately proceeds or follows $X_{t,w}$. For each test point, we obtain a 0/1 prediction and compare it to the ground truth. We repeat this procedure for all test points to compute the final precision, recall, and F-score.

In ED2, we are given a set of anomalies X_{t_i,w_i}. We randomly split this set into a training set and a test set. We can run a suitable ED method to generate a global explanation from the training set, and then use it as a predictive model over the test set. For each anomaly in the test set, we compare the point-wise prediction against the ground truth and compute precision, recall, F-score, similar to ED1.

4.3 Computational Performance

Exathlon can also be used to evaluate computational performance.

Evaluation Criteria and Metrics. ML algorithm performance is typically measured in terms of the total time it takes for model training as well as for using that model for making predictions. For AD, we define P1 and P2 to evaluate training and inference performance, respectively. For ED, the time to discover each explanation, P3, is our third performance metric.

Experimental Parameters. Exathlon offers scalability tests by varying the following two data-related parameters:

Dimensionality M: Our dataset consists of high-dimensional time series data. The 2,283 metrics (features) may be correlated and contain a lot of null values, which are representative of real-world datasets. The benchmark leaves it to each user algorithm as how it copes with the high dimensionality. The relevant techniques may include dimensionality reduction using linear transformation (e.g., PCA), or feature selection by leveraging the correlation structure in the data. Such choices are left to the discretion of each user algorithm, and Exathlon reports on the resulting dimensionality M used in AD and ED tasks.

Cardinality Factor α: Besides high dimensionality, our dataset also has high cardinality, $N = 2{,}335{,}781$ data items, which is significantly higher than the existing Numenta Anomaly Benchmark [34]. If the training time of an algorithm is too long, a user algorithm can choose to reduce the cardinality via resampling, i.e., by taking average of the data items in each l-second interval, which amounts to a cardinality factor $\alpha = 1/l$ and reduced data size of αN.

4.4 Broader Applicability

The Exathlon benchmark is more broadly applicable beyond our particular dataset [29].

AD Benchmark. The Exathlon benchmark considers all techniques that 1) train a model for data normality via learning from undisturbed traces, 2) assign outlier scores to new test records, and 3) derive binary predictions from these scores using a threshold. When meeting the above conditions, our AD metrics can be used with any labeled time series anomaly test datasets similar to ours. The four AD levels can be directly usable if labels are available as ranges (like for real or synthetic datasets from the discord discovery literature, also used in [9], [8]), while one would need to set our evaluation parameters to classical precision and recall if labels are available only as points (like for classification-oriented datasets tuned to data points of imbalanced classes, some of which are used in [52]). Applying such metrics also helps assess the performance and efficiency of any time series AD technique capable of assigning outlier scores and binary predictions to each record of a test sequence.

ED Benchmark. While a user study may be the best way to evaluate the usefulness of explanations, it is not always available and may come at a high cost. Therefore, our benchmark aims to provide *automated* evaluation of ED methods based on intuitive metrics, namely, *conciseness*, *consistency*, and *accuracy*, as well as their various variants in the ED1 and ED2 settings. As further detailed in our technical report [29], our metrics cover many of the metrics used in prior ED works, except for specific metrics that depend on a particular model or algorithm, which Exathlon deliberately avoids as a general benchmark, or require ground truth features or visual inspection by domain experts, which are not always available in complex domains. As the result of sharing metrics with existing works, the ED metrics of Exathlon can be applied to the datasets used in [3, 58], as well as those in [33, 45, 55] for the sake of evaluating the explanation for a classification result, and those in [37, 44] with sufficient preprocessing on the text or image data used.

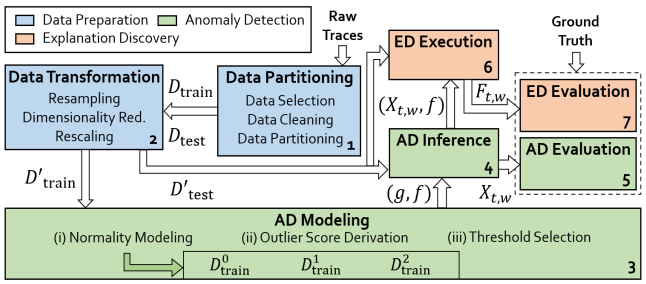


Figure 3: A pipeline for explainable AD on multivariate time series

5 A FULL PIPELINE FOR EXPLAINABLE AD

Besides a curated dataset and an evaluation methodology, Exathlon also provides a full pipeline for explainable AD on high-dimensional time series data. Our pipeline is characterized as follows: (i) It consists of the typical steps in a deep ML pipeline, ranging from data partitioning, feature engineering, dimensionality reduction, to AD and ED. (ii) It implements a variety of AD and ED functionalities and evaluation modules that score them based on the metrics of the benchmark (see §4). (iii) It provides an open, modular architecture that allows different methods to be added and combined through the pipeline. Figure 3 provides an overview.

1. Data Partitioning. The first phase takes as input the 93 raw traces, described in §3. It performs simple data cleaning, e.g., replacing missing data with a default value. It then performs data partitioning of the 93 traces. In the default setting, we take all undisturbed traces as training data, D_{train}, and all disturbed traces as test data, D_{test}. Other implementation choices are left to our report [29].

2. Data Transformation. As ML algorithms require data transformations to perform well, our pipeline offers the following steps:
(i) *Resampling* (optional): For the multivariate time series in each trace, the user can choose to resample, by taking the average of data points in each l-second interval. This step reduces the *cardinality factor*, $\alpha = 1/l$, of the time series data, if the training time turns out to be too long for some ML algorithms.
(ii) *Dimensionality reduction*: Since our dataset includes $M = 2,283$ raw features, such high dimensionality may affect both model accuracy, known as the "curse of dimensionality", and training time. To reduce dimensionality, our pipeline offers a PCA-based method (with a parameter that controls different coverage of the data variance and the resulting feature set size), as well as a manually curated feature set with 19 features selected using domain knowledge.
(iii) *Rescaling*: Most ML algorithms require the features to be scaled into a range, e.g., between [0, 1], to better align features whose raw values may differ by orders of magnitude. A unique issue in our problem is that each test trace may represent a new context, e.g., a combination of input rate and concurrency not seen in training data. As a result, rescaling has to take into account this new context. To simplify this setting, we provide the option of performing rescaling per trace, as well as a customized scaling method that rescales test data dynamically as we run an AD model over the data.

3. AD Modeling. The next phase takes the transformed training data and builds an AD model. Most AD methods build a model that describes the normal behavior in the data, called a "normality model", such that any future (test) data that deviates significantly from it will be flagged as an anomaly. Our pipeline offers an open architecture to embrace any AD method that builds such a normality model to detect anomalies. In this paper, we focus on recent DL-based AD methods [39, 47, 56], to explore their potential for handling the complexity of our dataset (high-dimensional, with noise), anomaly patterns (a variety of contextual and collective anomalies), and learning settings (noisy semi-supervised AD modeling).
(i) *Normality modeling*: The first step is to train a normality model based on a DL method of choice. Most DL methods take input data of fixed window size s. Given each of our traces, we create sliding windows of size s and slide 1, and feed them as input to the model. Different DL methods model the data in the window by either trying to forecast the data point following the window (forecasting-based, e.g., LSTM [10]) or reconstructing the window via a succinct internal representation (reconstruction based, e.g., Autoencoder [28, 56] or GANs [47]). To train each specific model, we divide the transformed D_{train} set into internal training (D^0_{train}), validation (D^1_{train}), and test (D^2_{train}) sets. The DL model is trained on D^0_{train}, with early stopping applied based on the model performance on D^1_{train}. Hyperparameter tuning is performed by choosing a configuration that maximizes model performance on D^2_{train}.
(ii) *Outlier score derivation*: We next build an initial AD model, $g : \boldsymbol{x} \in \mathbb{R}^m \rightarrow \mathbb{R}$, which maps each data point to an outlier score. For forecasting models, we compute the difference, d, between the forecast and true values of each data point, and derive the outlier score v based on d; the higher the d value, the higher the v score. For reconstruction-based models, we treat the reconstruction error of each window as the v score for that window, and then derive the v score of each data point by averaging the scores of its enclosed sliding windows. Our pipeline implements the LSTM [10]; Autoencoder (AE) [28]; and BiGAN [47] for AD. Details of these models and the necessary modifications we made to suit our dataset are deferred to our technical report due to space constraints.
(iii) *Threshold selection*: The last step aims to find a threshold on the outlier score to return a 0/1 prediction. It returns a final AD model, $f : \boldsymbol{x} \in \mathbb{R}^m \rightarrow \{0, 1\}$, mapping each data point to 0/1. Exathlon does not offer labeled data for threshold selection. Hence, we provide unsupervised threshold selection fit on D^2_{train}. Among the methods listed in a recent survey [57], we choose three most used automatic techniques: SD, MAD, and IQR, with the possibility of repeating them multiple times to filter large outlier scores.

4. AD Inference. Once the AD model is built, the next phase of the pipeline runs the AD model over each test trace to detect anomalies. In the context of range-based AD, predicted anomalies for a test trace are defined as sequences of positive predictions within that trace, denoted as $X_{t,w}$, which starts at t and has duration w.

5. AD Evaluation. The last AD phase evaluates the AD model for a given set of requirements. We evaluate both a model's ability to separate normal from anomalous data in the outlier score space and its final AD ability based on threshold selection. The separation ability (g) is assessed at the trace, application, and global levels. Global separation is reported as the AUPRC computed on all test data, while the application/trace-level separation is reported by computing an AUPRC for each application/trace, and averaging the

results. The detection ability (f) is assessed by reporting its range-based precision, recall, and F-score, with parameters specified by the AD functionality. Recall is also reported by anomaly type.

6. ED Execution. For each test trace, AD inference reports a set of anomalies, and for each reported anomaly, the ED module returns an explanation for it. Our pipeline supports two families of ED methods. (i) *Model-free ED methods* do not require the access to an ML model. Instead, they only require the anomalous instance, $X_{t,w}$, and a reference dataset, to generate an explanation. Examples include EXstream [58] and MacroBase [3] (§2). Our implementation sets the reference dataset as the subset of data that immediately proceeds the detected anomaly, denoted by $X_{t,-w'}$, and was classified as normal. Then the pair of datasets, $(X_{t,w}, X_{t,-w'})$, are provided to the ED method to generate an explanation, $F_{t,w}$. (ii) *Model-dependent ED methods* take not only the anomalous instance, $X_{t,w}$, but also an AD model, $f : \boldsymbol{x} \in \mathbb{R}^m \rightarrow \{0, 1\}$. Examples include LIME [44], Anchors [45], and SHAP [37] (§2). In our implementation, we provide the AD model used in inference to the ED method.

7. ED Evaluation. After processing each test trace, we obtain a set of anomalies with their corresponding explanations. We then collect the explanations from all the test traces to run the final ED evaluation and compute conciseness, consistency, accuracy, and time metrics. Further details are given in our technical report [29].

6 EXPERIMENTAL STUDY

In this section, we apply our benchmark to a select set of AD and ED methods. While a comprehensive comparison of related AD and ED methods is beyond the scope of this paper, analyzing the select methods allows us to demonstrate the value of our dataset and benchmark. Our analyses include the strengths and limitations of these AD and ED methods, challenges posed by our dataset and evaluation criteria, and some potential directions of future research.

6.1 Experimental Setup

In our experimental setup, we integrated into our pipeline three DL-based AD methods: LSTM [10], AE [28], and BiGAN [47]. We also integrated three recent ED methods: EXstream [58] and MacroBase [3], from the DB community for outlier explanation in data streams, and LIME [44], an influential method from the ML community. Details of these methods can be found in our report [29].

Besides the methods, our pipeline also needs to be configured with the following options: (a) **Data Size and Feature Set (FS)**: Since some of the DL models (e.g., GANs) could not complete training on our cluster using the full dataset, we reduced the data size by setting the cardinality factor, $\alpha = 1/15$. We also used a reduced feature set of 19 features, $m = 19$, produced by either manual selection based on domain knowledge, denoted as $\text{FS}_{\text{custom}}$, or by PCA with the same number (19) of features, denoted as FS_{pca}. For each given training set, we allowed each DL algorithm to train for 1.5 days (including hyperparameter tuning) to obtain an AD model. (b) **Level of AD Evaluation (AD1-4)**, as described in Table 2, with a default setting of AD2 (range detection).

6.2 AD Evaluation Results and Discussion

We begin by applying the LSTM [10], AE [28], and BiGAN [47] methods on our benchmark dataset and report on the AD metrics.

Sep Lvl	Method	Ave	AUPRC for Anomaly Types T1→T6					
Trace	**LSTM**	0.60	0.69	0.81	0.43	0.45	0.77	0.44
	AE	**0.73**	0.83	0.81	0.64	0.76	0.89	0.44
	BiGAN	0.61	0.91	0.76	0.15	0.70	0.64	0.51
App	**LSTM**	0.47	0.57	0.37	0.56	0.38	0.60	0.35
	AE	**0.57**	0.65	0.40	0.63	0.55	0.79	0.43
	BiGAN	0.52	0.81	0.36	0.25	0.54	0.69	0.48
Global	**LSTM**	0.41	0.56	0.32	0.53	0.25	0.53	0.27
	AE	**0.50**	0.60	0.36	0.54	0.47	0.68	0.37
	BiGAN	0.49	0.68	0.32	0.39	0.52	0.65	0.39

Table 3: Separation abilities of AD methods ($\text{FS}_{\text{custom}}$, AD2)

Experiment 1 ($\text{FS}_{\text{custom}}$, AD2). The first experiment compares the three AD methods under a default setting, ($\text{FS}_{\text{custom}}$, AD2). Here, we focus on the model's ability to separate anomalous data from normal data, via the analysis of trace-level, application-level, and global AUPRC results summarized in Table 3.

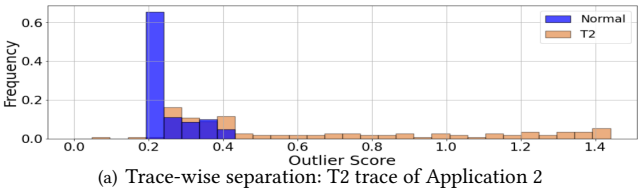
(1) *Trace-level Separation*: We first consider trace-level separation. All three methods achieved decent AUPRC scores for most (or a subset) of anomaly types, with AE achieving the highest score of 0.73. Figure 4(a) shows the distribution of outlier scores assigned by the AE method to the records in the T2 trace of Application 2. In this example, the normal records are separated from anomalous ones for most the data. This shows that our data indeed carries useful signals that can be picked up by the AD method, which allows the AD method to perform better than naive classifiers that randomly assign a normal or abnormal label, or assign each instance to the majority (normal) class – we refer to this remark as **R1**.

(2) *Application-level and Global Separation*: Moving from the trace- to application- to global level separation, the AUPRC scores gradually decrease. This is because the separation of normal from anomalous instances in outlier score becomes increasingly harder as we broaden the contexts in which data is generated. Figure 4(b) shows the distributions of outlier scores assigned by the AE method to all disturbed traces of Application 2.[1] At the application level, the outlier scores assigned to normal instances spread further, and start to mix with the outlier scores assigned to T2 anomaly instances, hence decreasing the model's separation ability. At the global level, this trend is aggravated, as we can see in Figure 4(c).

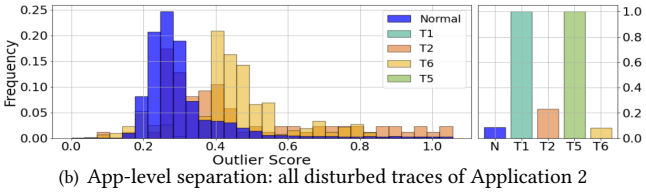
To understand why, Figures 4(e) and 4(f) show the outlier scores of the T1 and T2 traces of Application 2. The outlier scores assigned to some normal points in the T1 trace are in fact higher than the T2 anomalies, due to two reasons: (a) *Different contexts*: T1 and T2 traces were generated under different input rates, with the rate increase in T1 events around 2.5 times higher than in the T2 events. (b) *Noisy training data*: The normal data in T1 is "noisy". In fact, the normal records in the T1 trace that obtained higher outlier scores than the T2 anomaly exactly match the high processing delay due to Spark checkpointing activities. This indicates that the model has failed to capture these activities as normal behavior.

The above analyses show that our dataset carries a great deal of *variability* across traces (e.g., different input rates, concurrency among programs), and a small amount of *noise*. Such variability and noise make our dataset challenging for the three DL-based AD methods tested in this study (**R2**).

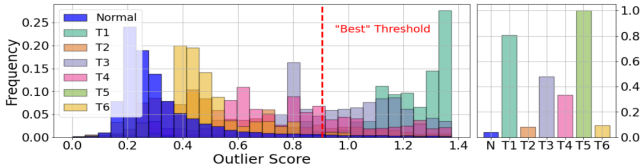
[1]For readability, outlier scores greater than 3 times the IQR were grouped together and shown separately in the right plot, which shows the proportion of records with outlier scores beyond 3∗IQR for each anomaly type.



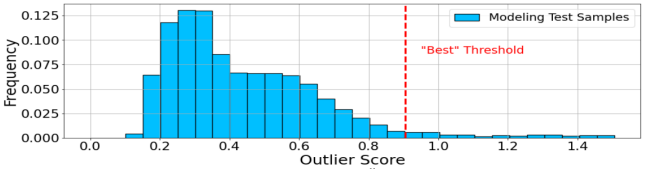
(a) Trace-wise separation: T2 trace of Application 2



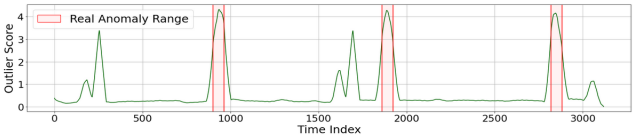
(b) App-level separation: all disturbed traces of Application 2



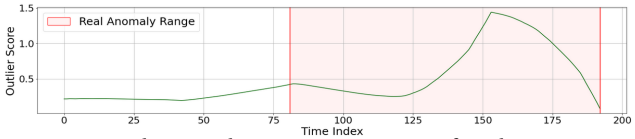
(c) Global-level separation: all disturbed traces with "best" AD2 threshold



(d) Modeling test samples with "best" AD2 threshold



(e) Record-wise outlier scores on a T1 trace of Application 2



(f) Record-wise outlier scores on a T2 trace of Application 2

Figure 4: Outlier score distributions and record-wise outlier scores using the AE method (FS_{custom})

AD1	F1	Prec	Rcl	Rcl for Anomaly Types T1→T6					
LSTM	0.77	0.67	0.96	1.00	1.00	1.00	1.00	1.00	0.67
AE	0.59	0.54	0.76	1.00	0.88	1.00	0.57	1.00	0.31
BiGAN	0.28	0.90	0.19	0.59	0.00	0.00	0.10	0.17	0.06
AD2	F1	Prec	Rcl	Rcl for Anomaly Types T1→T6					
LSTM	0.38	0.67	0.29	0.42	0.11	0.55	0.16	0.60	0.10
AE	0.52	0.54	0.60	0.97	0.15	0.67	0.40	1.00	0.15
BiGAN	0.17	0.90	0.10	0.30	0.00	0.00	0.06	0.17	0.00
AD3	F1	Prec	Rcl	Rcl for Anomaly Types T1→T6					
LSTM	0.29	0.67	0.20	0.27	0.04	0.37	0.10	0.58	0.08
AE	0.51	0.54	0.57	0.96	0.06	0.62	0.37	1.00	0.14
BiGAN	0.14	0.90	0.08	0.23	0.00	0.00	0.05	0.17	0.00
AD4	F1	Prec	Rcl	Rcl for Anomaly Types T1→T6					
LSTM	0.13	0.67	0.08	0.00	0.00	0.07	0.00	0.58	0.06
AE	0.49	0.52	0.56	0.94	0.06	0.58	0.37	1.00	0.14
BiGAN	0.14	0.86	0.08	0.23	0.00	0.00	0.05	0.17	0.00

Table 4: Median anomaly detection results (FS_{custom}, AD1:4)

(3) *Anomaly Type Comparison*: For different anomaly types, Table 3 shows that at the global level, the best separated types are T1, T3 and T5 for LSTM and AE, and T1, T4 and T5 for BiGAN. The good performance for T1 (bursty input) and T5 (driver failure) across all methods are largely due to the fact these types have very visible impacts on many of the features output by FS_{custom}, e.g., features relating to the input rate, application delays and memory usage for bursty input, and virtually all features for driver failure. However, most methods offer poor separation for T6 (executor failure) anomalies, due to the limited impact such anomalies have on the FS_{custom} features, where the 6 executor features are *averaged across active executor spots*. As such, the impact of an executor going down is only visible during the (short) period of time for which it shuts down and is potentially replaced. The above discussion shows that the variety of our anomaly types present signals of different strength levels in the data. They offer challenges for designing, as well as opportunities for analyzing, different AD methods, and feature engineering in the AD method will play a key role in preserving the signals for each anomaly type. (**R3**).

(4) *Method Comparison*: Regarding the separation ability, the best performing method is AE, followed by BiGAN, then LSTM for all levels. AE (and BiGAN) typically produce *smooth* point-wise outlier scores, by taking averages over overlapping windows. The outlier scores produced by the LSTM, however, often exhibit discontinuous *spikes*. For the task of range detection (AD2), such frequent mixes of high and low values make it hard to produce continuous ranges of high outlier scores, penalizing recall when the outlier threshold is set high or precision when the threshold is set low. Hence, we observe differences among AD methods as follows: for range detection (AD2), AE works the best while LSTM is the worse, mostly because the non-smooth outlier scores of LSTM make it hard to handle range anomalies (**R4**).

Experiment 2 (FS_{custom}, AD2). We next examine how the separation abilities translate into actual AD performance via threshold selection. Detection metrics for AD2 (range detection) are reported in the second section of Table 4. To generate the results, we ran each of the STD, IQR and MAD thresholding techniques, leading to different AD performance results for each AD method, for which we report the median performance in Table 4.

Among the three methods, AE provided the best median F1-score, due to its best separation ability reported in the previous experiment. However, this F1-score of AE is not very high (0.52). This is due to the difficulty in choosing a single threshold T on the outlier score for all traces and anomaly types in an *unsupervised* setting, where we select T by using part of the training data, D^2_{train}. Figure 4(d) shows the distribution of the outlier scores assigned to the D^2_{train} samples (the 3% largest were cut for readability), along with the best threshold found on them. This threshold is then used to flag anomalies in the test (disturbed) traces, as shown in Figure 4(c). We see that the anomalies whose scores lie left to T will be missed, penalizing recall, and the normal records whose scores lie right to T will lead to false positives, hurting precision. The above discussion shows that besides data characteristics, our benchmark

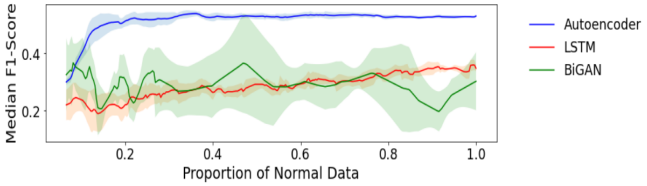


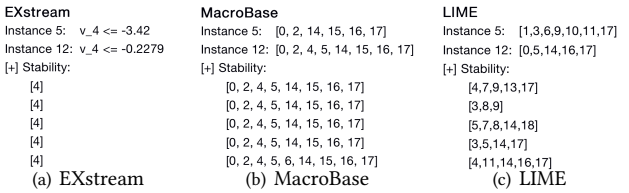
Figure 5: Median F1-score as training data increases for the three methods (FS_{custom}, AD2)

poses another challenge on AD methods due to the requirement of unsupervised threshold selection (**R5**).

Experiment 3 (FS_{custom}, AD2). To better understand the reasons behind the low F1-scores, we study the effect of the amount of training data on each method in Figure 5. The amount of training data was varied by starting from the largest undisturbed trace, and then randomly adding one undisturbed trace at a time until reaching the full set of undisturbed traces (except for the BiGAN method for which multiple traces could be added at once due to its longer training time). For each method, the above process was repeated 5 times. The average performance is reported in Figure 5 using solid lines, while the shaded areas correspond to the confidence region with width of one standard deviation.

We can see that the three methods behave quite differently as training data increases. First, the AE method benefits a lot from the first few traces that it obtains for training, but quickly reaches a performance plateau afterwards. This seems to indicate that what is holding back the AE performance is not simply the lack of training data, but rather the actual challenges posed by our benchmark (see remarks R2, R3, R5). On the other hand, the LSTM method and the BiGAN method to some extent seem to require more data to perform well. While LSTM exhibits roughly a linear trend, BiGAN appears less stable, which probably arises from the fact that GANs typically require more manual and calibrated tuning in order to converge to a good solution. For both these methods, adding more data could be beneficial, along with more tuning and experimentation to try to improve performance. Overall, this experiment suggests that the F1-score observed for AE could be primarily due its technical limitations for handling complex data, extracting most informative features for different anomaly types, and unsupervised threshold selection, while LSTM and BiGAN can further benefit from more training data and extensive hyperparameter tuning (**R6**).

Experiment 4 (FS_{custom}). We next evaluate the AD methods under different AD levels, AD1-4, of our benchmark. Results are reported in Table 4. (AD1) Given our range-based anomalies, a good recall score is easier to reach under AD1. We observe a general increase in performance for all methods. LSTM becomes the best method, because its spikes inside a real anomaly range are now sufficient for getting a good recall score, while using a high threshold to ensure good precision. (AD3) As AD3 awards less recall scores for late detection, AE maintains its performance, indicating that its reported range anomaly is not concentrated at the end of the true range. For LSTM, the performance drops because the early detections it makes are more scattered and hence weigh less in recall. (AD4) In AD4, reporting the same anomaly multiple times reduces the recall score. AE and BiGAN can maintain their performance while LSTM degrades significantly. Again, the tendency of



(a) EXstream (b) MacroBase (c) LIME

Figure 6: Example explanations given by EXstream, MacroBase, and LIME for two instances of stalled input (T3) anomaly

LSTM to produce outlier scores in discontinuous spikes makes it more likely to report multiple anomalies where only one is needed. Hence, we see that the different AD levels in our benchmark indeed pose varying levels of challenges to the AD methods (**R7**).

6.3 ED Evaluation Results and Discussion

Next, we report results of running MacroBase [3], EXstream [58], LIME [44] on our benchmark dataset to generate anomaly explanations. For each anomaly, MacroBase and EXstream tried to explain its separation from a reference dataset, while LIME tried to explain the reason behind the high record outlier scores assigned by an AD model (AE in our case). Since LIME only explained the predictions of window size s of our AE model, if an anomaly was larger than s, we created multiple windows for LIME to explain.

Table 5 summarizes *conciseness*, *consistency*, *normalized consistency*, *accuracy*, and *running time* for local (ED1) or global (ED2) explanations using the three ED methods. We also show example explanations in Figure 6, which are the explanations returned for two instances of stalled input anomaly (T3). Only feature indices are reported here (for feature names, see [29]). Figure 6(a) reports the complete explanations returned by EXstream, while showing the features appearing in the explanations for the others due to limited space. For each method, we also show the features returned when explaining 5 different samples of anomaly instance #12 (stability).

Explanations. The explanations shown highlight the impact of feature correlation. Although MacroBase and EXstream output different features (Figure 6(a) and 6(b)), they might both be correct. For example, features 4, 5 and 14 are related to processed records, received records, and CPU time [29]. For a human user, it makes sense for these three features to be used for a stalled input anomaly. EXstream picks up only one most important feature among the correlated ones. MacroBase returns all important features no matter whether they are correlated or not. LIME is known to have inconsistency issues [41], which is also illustrated here.

Algorithm Analyses. We discuss the results for each algorithm.

MacroBase. (1) MacroBase generated explanations of 3.16 features on avg. across anomaly types. For some anomaly types, e.g., T3, it generated longer explanations (6-7 features). The algorithm does not consider compactness, outputting longer explanations in presence of correlated features. (2) The explanations it provided were not very locally stable (ED1 consistency), with an entropy score outside the ideal range of $H_1 = 0$ and $H_3 = 1.58$. This relates to a correlation between conciseness and stability: as Table 5 shows for different anomaly types, longer explanations tend to be less stable. For global consistency, its concordance value further degrades, using inconsistent features for explanations of the same anomaly type.

	MacroBase											EXstream											LIME						
	Concise		Consistency		Norm.Cons		Prec		Rel		Time (sec)	Concise		Consistency		Norm.Cons		Prec		Rel		Time (sec)	Concise		Consistency		Norm.Cons		Time (sec)
	ED1	ED2	ED1	ED2	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2	ED1	ED2	ED1	ED2	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2
T1	2.36	2.10	1.41	2.12	1.23	2.06	0.96	0.81	0.92	0.84	0.87	2.25	2.17	1.59	3.36	1.56	4.74	0.88	0.82	0.75	0.45	0.0162	3.58	5.86	3.04	3.82	2.33	2.41	259
T2	1.94	1.71	0.98	1.33	1.11	1.46	0.97	0.79	0.97	0.90	1.05	3.89	3.71	2.13	3.59	1.37	3.25	0.71	0.87	0.53	0.51	0.0087	3.74	9.29	3.42	4.16	2.88	1.93	237
T3	6.08	4.83	2.98	3.07	1.37	1.74	0.97	0.88	0.91	0.51	8.75	3.35	3.75	2.13	3.67	1.57	3.40	0.82	0.78	0.65	0.18	0.0156	3.83	6.58	3.28	3.93	2.59	2.32	254
T4	1.51	1.55	1.11	2.90	1.59	4.82	0.73	0.52	0.77	0.42	0.14	3.41	3.80	1.80	3.78	1.27	3.62	0.61	0.51	0.33	0.11	0.0106	4.03	6.10	3.34	3.97	2.56	2.57	241
T5	4.63	1.71	2.79	2.79	1.77	4.04	0.13	0.29	0.11	0.40	0.96	1.66	1.71	1.08	2.52	1.47	3.35	0.34	0.40	0.34	0.41	0.0067	4.57	4.33	3.53	3.72	2.55	3.04	242
T6	2.42	1.75	1.55	2.22	1.29	2.66	0.80	0.17	0.82	0.50	0.24	2.75	2.88	1.20	3.50	1.30	3.94	0.72	0.30	0.59	0.27	0.0088	4.20	4.43	3.44	3.65	2.66	2.84	239
Ave	3.16	2.28	1.80	2.40	1.39	2.80	0.76	0.58	0.75	0.59	2.00	2.88	3.00	1.66	3.41	1.42	3.72	0.68	0.61	0.53	0.32	0.0111	3.99	6.10	3.34	3.88	2.60	2.52	245

Table 5: Results of ED methods, MacroBase, EXstream, and LIME, in terms of conciseness, consistency, accuracy, and running time

This issue was alleviated by measuring normalized consistency. In the example of Fig. 6(b), the stability of MacroBase for instance #12 is 3.09, while its normalized stability is 1.03, almost perfect. The latter value is in accordance with human intuition, since across the 5 runs, the reported features were almost the same. (3) Its ED1 accuracy is good for some anomaly types (e.g., T1-3), but poor for some others (e.g., T5). As expected, its accuracy degraded from ED1 to ED2. (4) Its execution time is around 1 sec, except for T3.

EXstream. (1) EXstream provided more concise explanations, using multiple techniques to prune marginally related features. (2) It achieved good explanation stability, with an average entropy score of $H_3 = 1.58$, but worse explanation concordance, be it normalized or not. This is likely due to our exclusion of the false positive filtering step, requiring additional labels. Thus, some features standing out as different during anomalous periods might not be related to the anomaly but to normal changes between two contiguous periods. Such features are likely to be more distinct between instances of different contexts than for perturbations of the same instance, hence the greater effect on concordance. (3) Its ED1 accuracy is good for some anomaly types (e.g., T1-3), but not for others (e.g., T3-6), especially in recall. Its accuracy degraded for ED2, as expected. (4) Its execution time is very short, being a streaming algorithm.

LIME. (1) LIME generated longer explanations, e.g., for anomaly types T1-T3. (2) Its consistency scores were better maintained from ED1 to ED2, with normalized metrics alleviating the observed correlations between size and consistency. Accuracy measures do not apply to LIME, since it could not be compared to the others for prediction. (4) Its execution time is very long, up to 245 sec on avg.

Comparison. We next compare the three methods, first in *conciseness* and *stability*. MacroBase lacks a mechanism for minimizing the size of an explanation, while LIME relies on sparse linear regression to select few features. Neither was as effective as EXstream, which eagerly prunes marginally related features through its heuristic to a non-monotone submodular optimization problem. Stability being positively correlated to conciseness, the longer explanations of MacroBase and LIME are also less stable. For global explanations, *concordance* was harder to achieve than stability for all methods, i.e., a user is likely to see explanations built on different features for anomalies of the same type, which is undesirable. This lack of concordance indicates a direction for future ED research. For local *accuracy*, the logical formulas derived by MacroBase and EXstream on a subset of each instance could be evaluated for AD on their remaining part and neighboring normal data. MacroBase was more accurate, paying the cost of evaluating a large number of feature combinations, while EXstream suffered in recall, "overfitting" each anomalous instance. LIME, returning only feature importance scores, could not be evaluated for accuracy. In the global setting, accuracy degraded for all methods. By hard-coding context-dependent constants in predicates, the explanations of MacroBase and EXstream did not generalize well to other contexts (e.g., different input rates). This phenomenon is intrinsic to point-based explanations. In order to free explanations from such context-dependent predicates, transitioning from point-based to temporal explanations, capturing causal and context-free relationships between events, could be a direction for future research. In *efficiency*, EXstream was the fastest, taking $\tilde{0}.01$ sec to generate explanations, against 0.2-9 sec for MacroBase and > 4 min for LIME. As such, LIME is unsuitable for stream processing use cases, with its high latency preventing timely corrective actions, e.g., avoiding an application crash or denial of service.

7 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we presented Exathlon – a novel public benchmark for explainable AD, and demonstrated its utility through an experimental analysis of selected AD and ED algorithms from recent literature. Our AD results show that Exathlon's dataset is valuable for evaluating AD algorithms due to rich signals and diverse anomaly types included in the data. Yet more importantly, our results reveal the limitations of these AD methods for semi-supervised learning under noisy training data and mixed anomaly types. On the ED front, the literature lacked comparative analysis tools and studies. Our benchmark fills this gap by providing a common framework for analyzing the strengths and limitations of diverse ED methods in their conciseness, consistency, accuracy, and efficiency. These results call for new research to advance the current state of the art of AD and ED, as well as integrated solutions to anomaly and explanation discovery. For a true integration, ED methods should first become capable of discovering range-based explanations, which is also a key step towards automated root cause analysis (a.k.a., "why explanations"). Exathlon's dataset and extensible design are well-positioned to support research progress towards these goals in the long term. Going forward, we envision Exathlon to develop into a collaborative community platform for fostering reproducible research and experimentation in the area. We intend to actively maintain and extend this platform, as well as welcoming feedback and contributions from the AD and ED communities.

ACKNOWLEDGMENTS

This work was supported by the European Research Council (ERC) Horizon 2020 research and innovation programme (grant n725561).

REFERENCES

[1] Arvind Arasu, Mitch Cherniack, Eddie F. Galvez, David Maier, Anurag Maskey, Esther Ryvkina, Michael Stonebraker, and Richard Tibbetts. 2004. Linear Road: A Stream Data Management Benchmark. In *VLDB Conference*. 480–491.

[2] Anthony J. Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn J. Keogh. 2018. The UEA Multivariate Time Series Classification Archive, 2018. *CoRR* abs/1811.00075 (2018).

arXiv:1811.00075 http://arxiv.org/abs/1811.00075 Accessed: 2021-07-27.
[3] Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. 2017. MacroBase: Prioritizing Attention in Fast Data. In *ACM International Conference on Management of Data (SIGMOD)*. 541–556.
[4] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. 2019. ObjectNet: A Large-Scale Bias-controlled Dataset for Pushing the Limits of Object Recognition Models. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 9453–9463.
[5] Ali Basiri, Niosha Behnam, Ruud de Rooij, Lorin Hochstein, Luke Kosewski, Justin Reynolds, and Casey Rosenthal. 2016. Chaos Engineering. *IEEE Software* 33, 3 (2016), 35–41.
[6] Leopoldo E. Bertossi, Jordan Li, Maximilian Schleich, Dan Suciu, and Zografoula Vagena. 2020. Causality-based Explanation of Classification Outcomes. In *Fourth Workshop on Data Management for End-To-End Machine Learning (DEEM)*. 6:1–6:10.
[7] Ana Maria Bianco, Marta Garcia Ben, Eunie Jr. Martinez, and Victor J. Yohai. 2001. Outlier Detection in Regression Models with ARIMA Errors using Robust Estimates. *Journal of Forecasting* 20, 8 (2001), 565–579.
[8] Paul Boniol, Michele Linardi, Federico Roncallo, and Themis Palpanas. 2020. Automated Anomaly Detection in Large Sequences. In *IEEE 36th International Conference on Data Engineering (ICDE)*. 1834–1837.
[9] Paul Boniol and Themis Palpanas. 2020. Series2Graph: Graph-Based Subsequence Anomaly Detection for Time Series. *Proceedings of the VLDB Endowment (PVLDB)* 13, 12 (2020), 1821–1834.
[10] Loïc Bontemps, Van Loi Cao, James McDermott, and Nhien-An Le-Khac. 2016. Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks. In *International Conference on Future Data and Security Engineering (FDSE)*, Vol. 10018. 141–152.
[11] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-based Local Outliers. In *ACM International Conference on Management of Data (SIGMOD)*. 93–104.
[12] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep Learning for Anomaly Detection: A Survey. *CoRR* abs/1901.03407 (2019). arXiv:1901.03407 http://arxiv.org/abs/1901.03407 Accessed: 2021-07-27.
[13] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Computing Surveys* 41, 3 (2009), 15:1–15:58.
[14] Cody Coleman, Daniel Kang, Deepak Narayanan, Luigi Nardi, Tian Zhao, Jian Zhang, Peter Bailis, Kunle Olukotun, Christopher Ré, and Matei Zaharia. 2019. Analysis of DAWNBench, a Time-to-Accuracy Machine Learning Performance Benchmark. *ACM SIGOPS Operating Systems Review* 53, 1 (2019), 14–25.
[15] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking Cloud Serving Systems with YCSB. In *ACM Symposium on Cloud Computing (SoCC)*. 143–154.
[16] The Standard Performance Evaluation Corporation. [n.d.]. SPEC Benchmarks. https://www.spec.org/ Accessed: 2021-07-27.
[17] The Transaction Processing Council. [n.d.]. TPC Benchmarks. http://www.tpc.org/ Accessed: 2021-07-27.
[18] Hoang Anh Dau, Anthony J. Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn J. Keogh. 2018. The UCR Time Series Archive. *CoRR* abs/1810.07758 (2018). arXiv:1810.07758 http://arxiv.org/abs/1810.07758 Accessed: 2021-07-27.
[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 248–255.
[20] Dheeru Dua and Casey Graff. [n.d.]. The UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/ Accessed: 2021-07-27.
[21] Kareem El Gebaly, Parag Agrawal, Lukasz Golab, Flip Korn, and Divesh Srivastava. 2014. Interpretable and Informative Explanations of Outcomes. *Proceedings of the VLDB Endowment (PVLDB)* 8, 1 (2014), 61–72.
[22] Andrew F. Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. 2013. Systematic Construction of Anomaly Detection Benchmarks from Real Data. In *ACM SIGKDD Workshop on Outlier Detection and Description (ODD)*. 16–21.
[23] FICO. 2018. Explainable Machine Learning Challenge. https://community.fico.com/s/explainable-machine-learning-challenge Accessed: 2021-07-27.
[24] Nicholas Frosst and Geoffrey E. Hinton. 2017. Distilling a Neural Network Into a Soft Decision Tree. In *International Workshop on Comprehensibility and Explanation in AI and ML*.
[25] Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, Meikel Poess, Alain Crolotte, and Hans-Arno Jacobsen. 2013. BigBench: Towards an Industry Standard Benchmark for Big Data Analytics. In *ACM SIGMOD International Conference on Management of Data*. 1197–1208.
[26] Jim Gray. 1993. *The Benchmark Handbook for Database and Transaction Systems*. Morgan Kaufmann.
[27] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. 2014. Outlier Detection for Temporal Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 26, 9 (2014), 2250–2267.
[28] Geoffrey Hinton and Ruslan Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 (2006), 504 – 507.
[29] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul. 2021. Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series. *CoRR* abs/2010.05073 (2021). arXiv:2010.05073 http://arxiv.org/abs/2010.05073 Accessed: 2021-07-27.
[30] Vimalkumar Jeyakumar, Omid Madani, Ali Parandeh, Ashutosh Kulshreshtha, Weifei Zeng, and Navindra Yadav. 2019. ExplainIt! - A Declarative Root-cause Analysis Engine for Time Series Data. In *ACM International Conference on Management of Data (SIGMOD)*. 333–348.
[31] Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitu Padhye, and Victor Bahl. 2009. Detailed Diagnosis in Enterprise Networks. In *ACM SIGCOMM Conference*. 243–254.
[32] Martin Kopp, Tomás Pevný, and Martin Holena. 2020. Anomaly Explanation with Random Forests. *Expert Systems with Applications* 149 (2020), 113187.
[33] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *ACM SIGKDD Conference*. 1675–1684.
[34] Alexander Lavin and Subutai Ahmad. 2015. Evaluating Real-Time Anomaly Detection Algorithms - The Numenta Anomaly Benchmark. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*. 38–44.
[35] Boduo Li, Edward Mazur, Yanlei Diao, Andrew McGregor, and Prashant J. Shenoy. 2012. SCALLA: A Platform for Scalable One-Pass Analytics Using MapReduce. *ACM Transactions on Database Systems* 37, 4 (2012), 27.
[36] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data* 6, 1 (2012), 3:1–3:39.
[37] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Annual Conference on Neural Information Processing Systems (NIPS)*. 4765–4774.
[38] Minghua Ma, Zheng Yin, Shenglin Zhang, Sheng Wang, Christopher Zheng, Xinhao Jiang, Hanwen Hu, Cheng Luo, Yilin Li, Nengjun Qiu, Feifei Li, Changcheng Chen, and Dan Pei. 2020. Diagnosing Root Causes of Intermittent Slow Queries in Large-Scale Cloud Databases. *Proceedings of the VLDB Endowment (PVLDB)* 13, 8 (2020), 1176–1189.
[39] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2015. Long Short Term Memory Networks for Anomaly Detection in Time Series. In *European Symposium on Artificial Neural Networks (ESANN)*. 89–94.
[40] George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* 38, 11 (1995), 39–41.
[41] Christoph Molnar. 2021. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. https://christophm.github.io/interpretable-ml-book/ Accessed: 2021-07-27.
[42] Tilmann Rabl, Christoph Brücke, Philipp Härtling, Stella Stars, Rodrigo Escobar Palacios, Hamesh Patel, Satyam Srivastava, Christoph Boden, Jens Meiners, and Sebastian Schelter. 2019. ADABench - Towards an Industry Standard Benchmark for Advanced Analytics. In *TPC Technology Conference on Performance Evaluation and Benchmarking (TPCTC)*. 47–63.
[43] Shebuti Rayana. 2016. Outlier Detection DataSets (ODDS) Library. http://odds.cs.stonybrook.edu/ Accessed: 2021-07-27.
[44] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *ACM SIGKDD Conference*, 1135–1144.
[45] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. *AAAI Conference*.
[46] Sudeepa Roy, Laurel Orr, and Dan Suciu. 2015. Explaining Query Answers with Explanation-Ready Databases. *Proceedings of the VLDB Endowment (PVLDB)* 9, 4 (2015), 348–359.
[47] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. 2017. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In *International Conference on Information Processing in Medical Imaging (IPMI)*. 146–157.
[48] Nidhi Singh and Craig Olinsky. 2017. Demystifying Numenta Anomaly Benchmark. In *International Joint Conference on Neural Networks (IJCNN)*. 1570–1577.
[49] Spark-uses [n.d.]. How are Big Companies using Apache Spark. https://medium.com/@tao_66792/how-are-big-companies-using-apache-spark-413743dbbbae Accessed: 2021-07-27.
[50] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. *ICML*, 3319–3328.
[51] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. 2018. Precision and Recall for Time Series. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 1924–1934.
[52] Luan Tran, Liyue Fan, and Cyrus Shahabi. 2015. Distance Based Outlier Detection for Data Streams. *Proceedings of the VLDB Endowment (PVLDB)* 9, 4 (2015), 1089–1100.

[53] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations (ICLR)*.

[54] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining Away Outliers in Aggregate Queries. *Proceedings of the VLDB Endowment (PVLDB)* 6, 8 (2013), 553–564.

[55] Mike Wu, Michael C. Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. 2018. Beyond Sparsity: Tree Regularization of Deep Models for Interpretability. *AAAI Conference*.

[56] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, and Honglin Qiao. 2018. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In *International World Wide Web Conference (WWW)*. 187–196.

[57] Jiawei Yang, Susanto Rahardja, and Pasi Fränti. 2019. Outlier Detection: How to Threshold Outlier Scores?. In *International Conference on Artificial Intelligence, Information Processing and Cloud Computing (AIIPCC)*. 37:1–37:6.

[58] Haopeng Zhang, Yanlei Diao, and Alexandra Meliou. 2017. EXstream: Explaining Anomalies in Event Stream Monitoring. In *International Conference on Extending Database Technology (EDBT)*. 156–167.